情報処理（14週目）
Pythonを用いた画像処理

王 忠奎（wangzk@fc.ritsumei.ac.jp）
立命館大学　ロボティクス学科
2024.07.15

講義の流れ

- グレースケール画像への変更と平滑化処理
- エッジ検出と画像の切り抜き
- 画像の回転と反転
- 画像の二値化と色空間の変換
- 輪郭抽出
- 顔と目の認識
- レポート

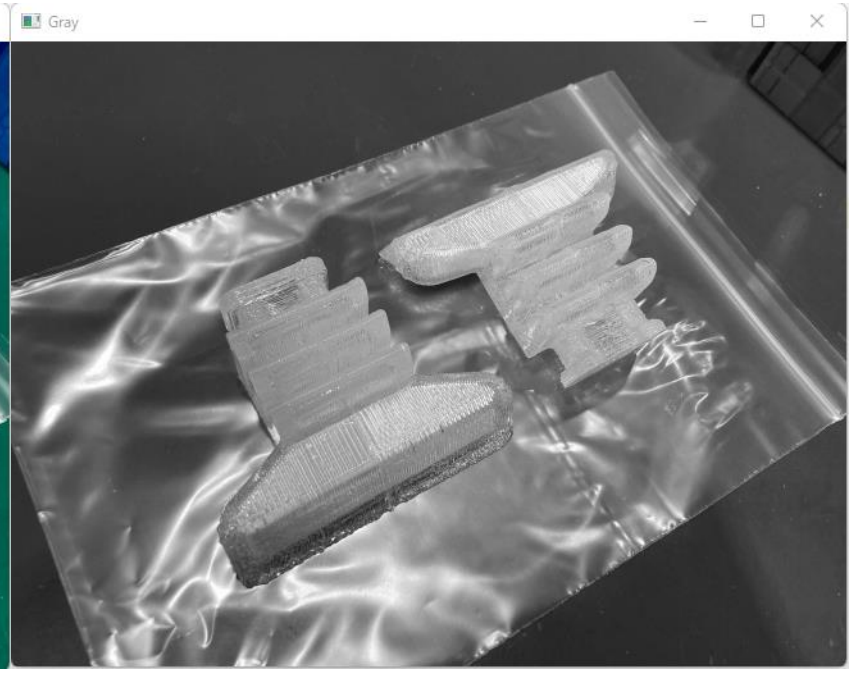
グレースケール画像への変更と平滑化処理

```
import cv2 as cv

img = cv.imread('image1_rescaled.jpg')  # 画像の読み込み
cv.imshow('Original', img)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)  # グレースケールへの変更
cv.imshow('Gray', gray)

blur = cv.GaussianBlur(img, (3,3), cv.BORDER_DEFAULT)  # 平滑化
cv.imshow('Blur', blur)

cv.waitKey(0)
```

カーネル

グレースケール画像への変更と平滑化処理

元画像



グレースケール元画像

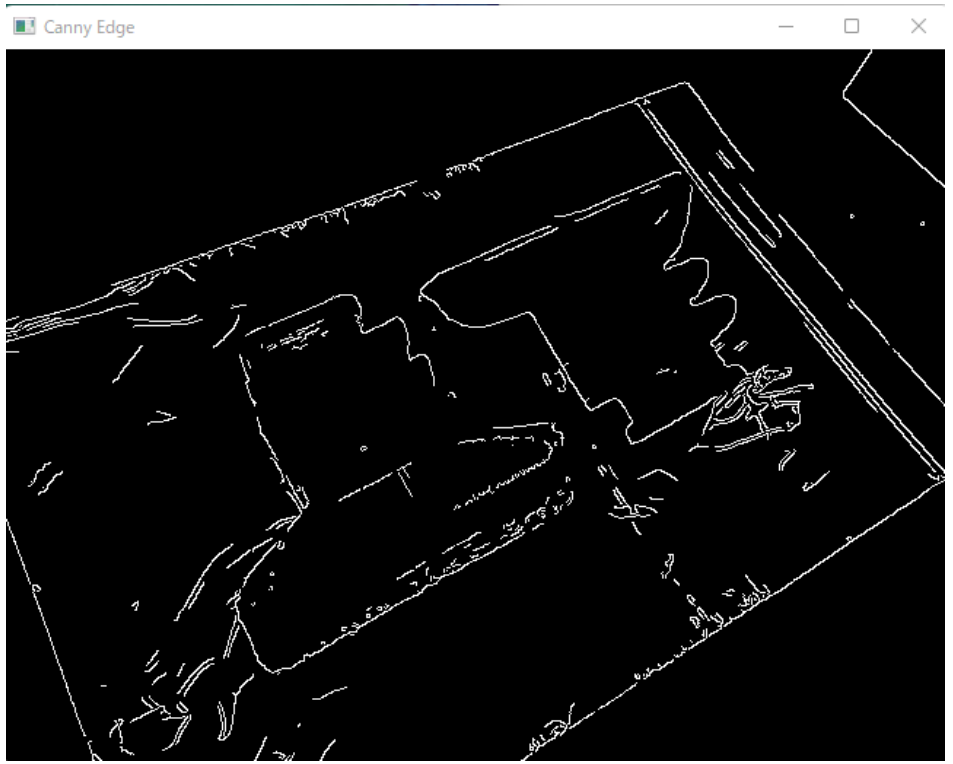
平滑化画像

エッジ検出と画像の切り抜き

```
import cv2 as cv

img = cv.imread('image1_rescaled.jpg')  # 画像の読み込み
cv.imshow('Original', img)

canny = cv.Canny(img, 200, 300)  # エッジ検出
cv.imshow('Canny Edge', canny)

cropped = img[50:200, 200:400]  # 画像の切り抜き
cv.imshow('cropped', cropped)

cv.waitKey(0)
```

エッジ検出と画像の切り抜き

元画像

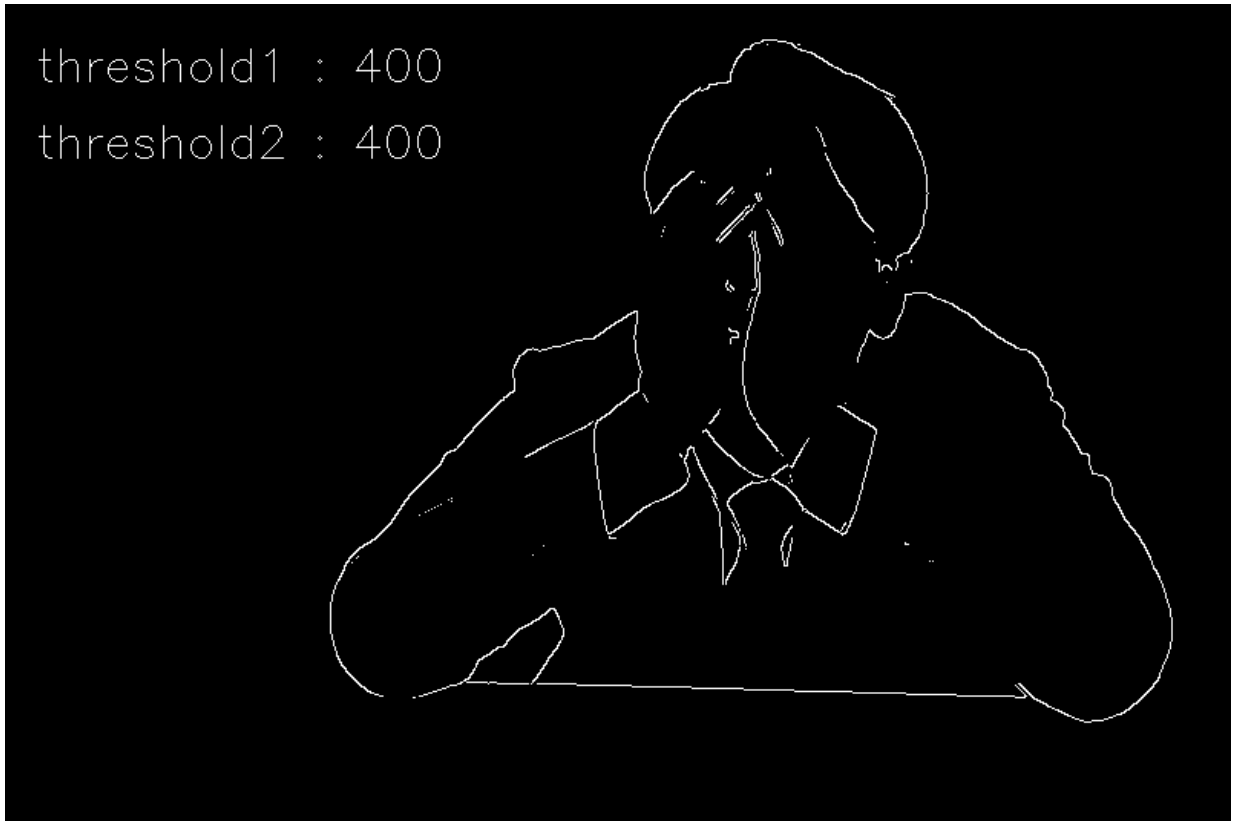


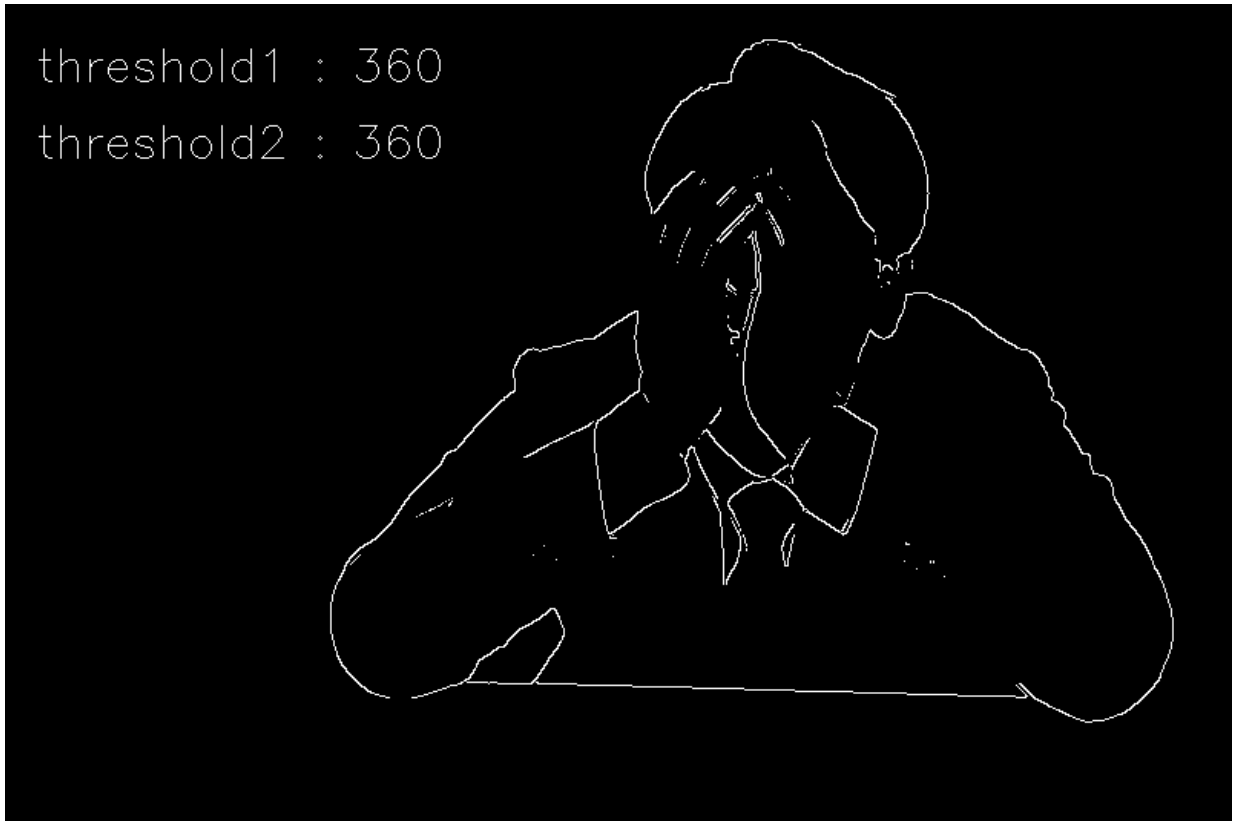
エッジ検出

画像の切り抜き

エッジ検出閾値

cv2.Canny(img, threshold1, threshold2)





1.threhsold1 を threshold2 と同じ値となるようにしておく。
2.threshold2 を検出されてほしいところにエッジが現れるように調整する。
3.threshold1 を使用してエッジを育てる

https://qiita.com/Takarasawa_/items/1556bf8e0513dca34a19

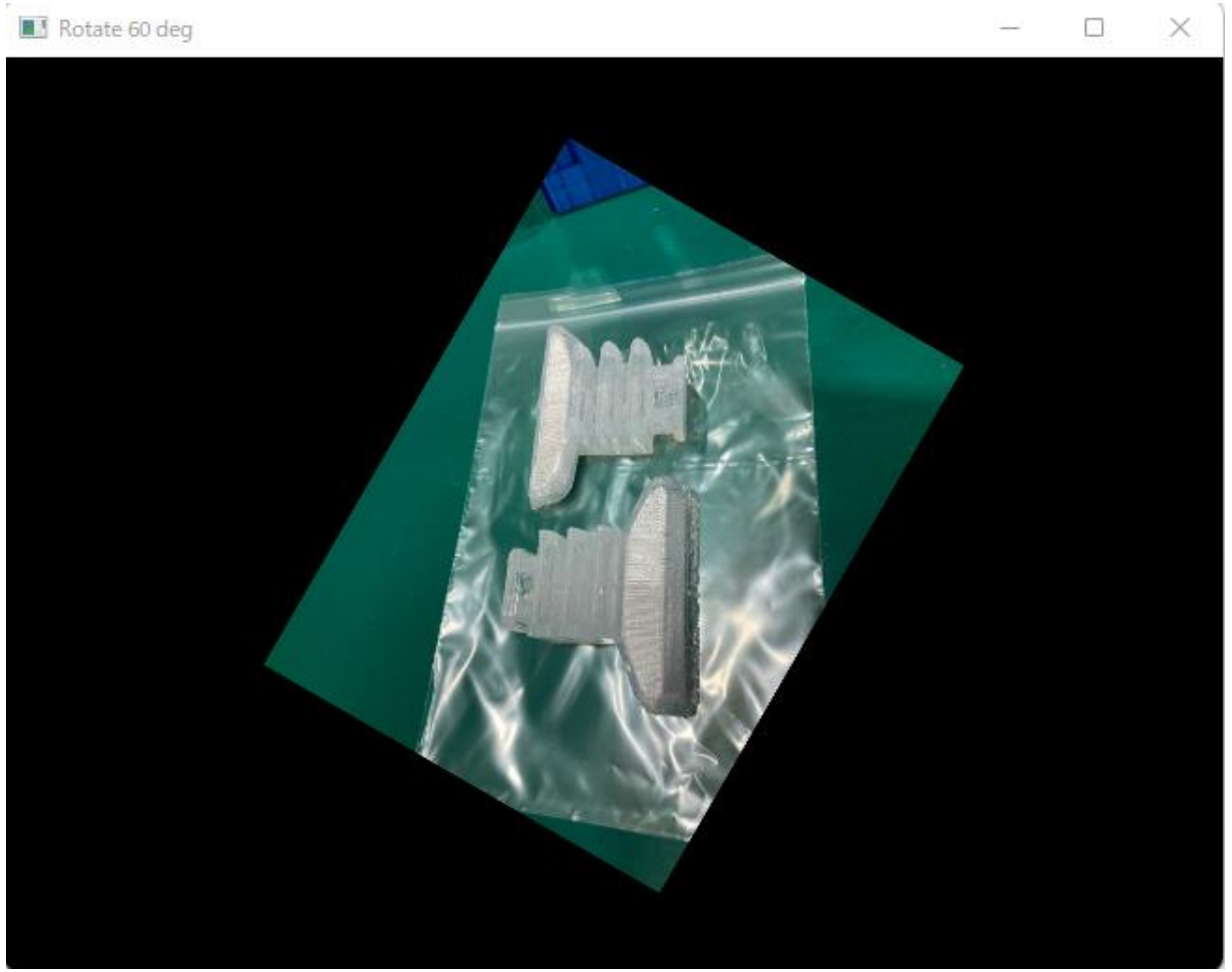
画像の回転

```
import cv2 as cv

img = cv.imread('image1_rescaled.jpg')  # 画像の読み込み
cv.imshow('Original', img)

height, width = img.shape[:2]
rotPoint = (width/2, height/2)

rotMat = cv.getRotationMatrix2D(rotPoint, 60, 0.5)  # 回転行列の取得
dimensions = (width, height)

img_rotate = cv.warpAffine(img, rotMat, dimensions)  # 画像の回転
cv.imshow('Rotate 60 deg', img_rotate)

cv.waitKey(0)
```

スケーリング

画像の回転

元画像



回転画像

画像の上下左右反転

- `flipcode = 0`：上下反転
- `flipcode > 0`：左右反転
- `flipcode < 0`：上下左右反転

```
import cv2 as cv

img = cv.imread('image1_rescaled.jpg')  # 画像の読み込み
cv.imshow('Original', img)

img_flip_v = cv.flip(img, 0)  # 上下反転
cv.imshow('Flip vertically', img_flip_v)

img_flip_h = cv.flip(img, 1)  # 左右反転
cv.imshow('Flip horizontally', img_flip_h)

img_flip_vh = cv.flip(img, -1)  # 上下と左右反転
cv.imshow('Flip vertically and horizontally', img_flip_vh)

cv.waitKey(0)
```

画像の上下左右反転

元
画像

上下
反転

左右
反転

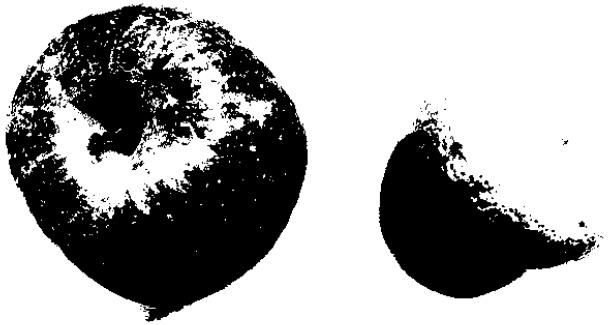
上下
左右
反転

画像の二値化

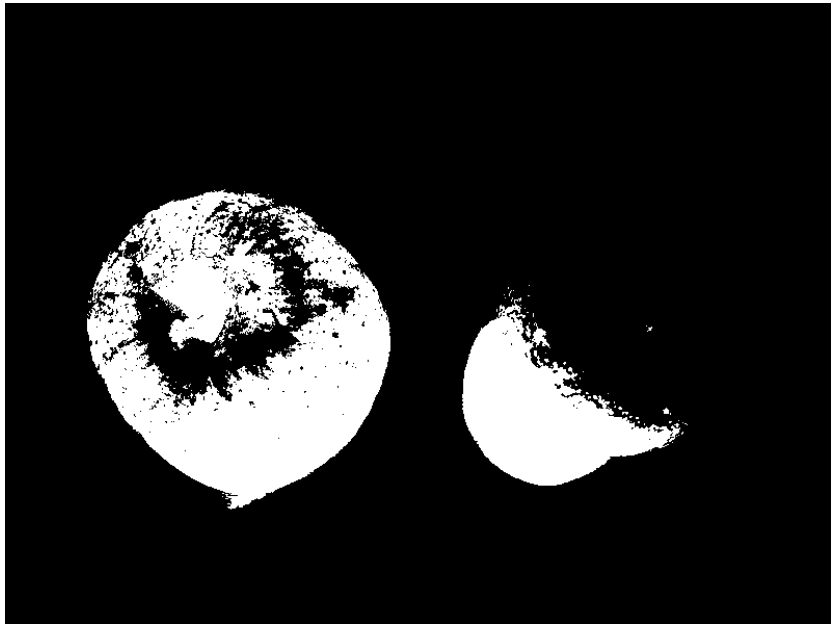
```
import cv2 as cv

img = cv.imread('objects_rescaled.jpg')
cv.imshow('Objects', img)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)  # グレー画像に変更
ret, thresh = cv.threshold(gray, 80, 255, cv.THRESH_BINARY)  # 二値化
cv.imshow('Simple threshold', thresh)

ret_inv, thresh_inv = cv.threshold(gray, 80, 255, cv.THRESH_BINARY_INV)  # 二値化
cv.imshow('Simple threshold inverse', thresh_inv)

cv.waitKey(0)
```

画像の二値化

元画像



80以上白色



80以上黒色

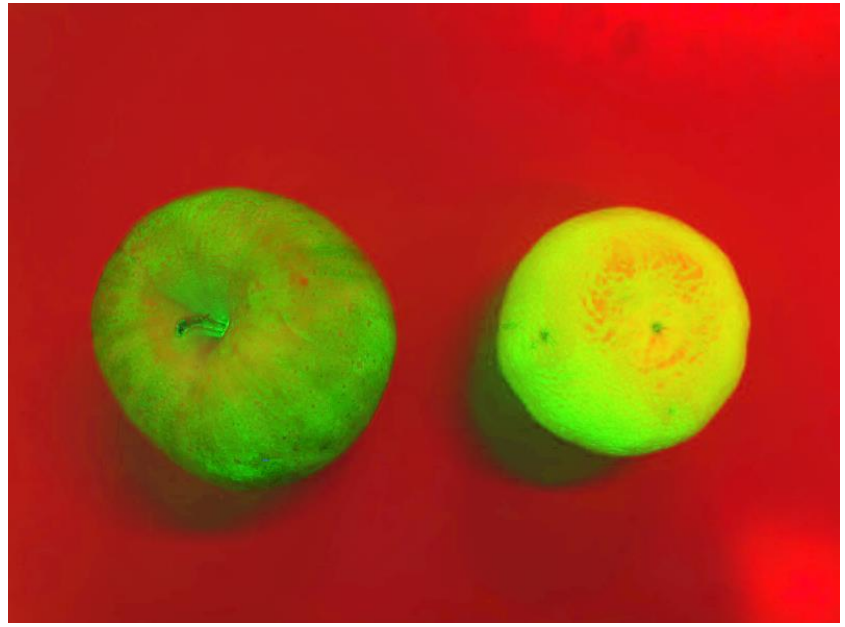
色空間の変換

```
import cv2 as cv

img = cv.imread('objects_rescaled.jpg')  # 画像の読み込み
cv.imshow('Original', img)

RGB = cv.cvtColor(img, cv.COLOR_BGR2RGB)  # BGRからRGBへの変換
cv.imshow('RGB image', RGB)

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)  # BGRからHSVへの変換
cv.imshow('HSV image', hsv)

cv.waitKey(0)
```

色空間の変換

元画像

RGB画像



HSV画像

輪郭抽出－－方法１

```
import cv2 as cv

img = cv.imread('objects_rescaled.jpg')
cv.imshow('Objects', img)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Gray', gray)

ret, thresh = cv.threshold(gray, 100, 255, cv.THRESH_BINARY)
cv.imshow('Thresh', thresh)

contours, hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)

img_contour = cv.drawContours(img, contours, -1, (0, 255, 0), 5)
cv.imshow('Image with contour', img_contour)

cv.waitKey(0)
```

同じ階層で取得

すべての点を取得

輪郭番号　線の色　線の幅

findContours関数

contours, hierarchies = cv.findContours(image, mode, method)

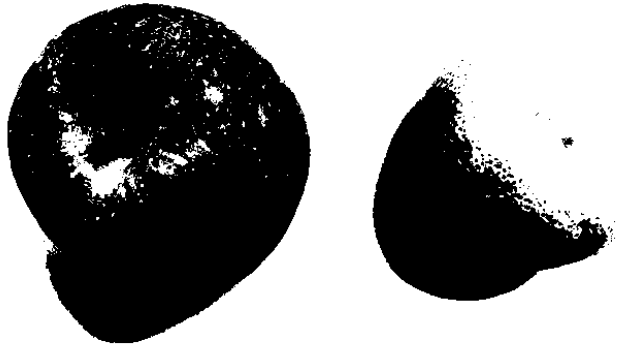
image	輪郭抽出する画像データ
mode	輪郭構造の取得方法
method	輪郭座標の取得方法
(戻り値)contours	輪郭座標
(戻り値)hierarchy	輪郭の階層情報

mode: RETR_EXTERNAL,
RETR_LIST,
RETR_CCOMP,
RETR_TREE,
RETR_FLOODFILL

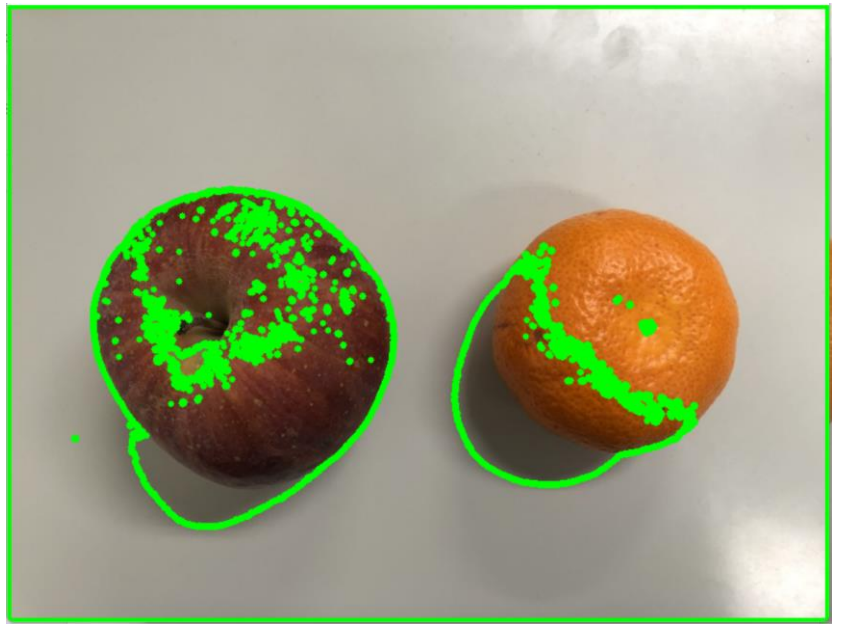
method: CHAIN_APPROX_NONE,
CHAIN_APPROX_SIMPLE,
CHAIN_APPROX_TC89_L1,
CHAIN_APPROX_TC89_KCOS

輪郭抽出結果１

元画像



二値化画像



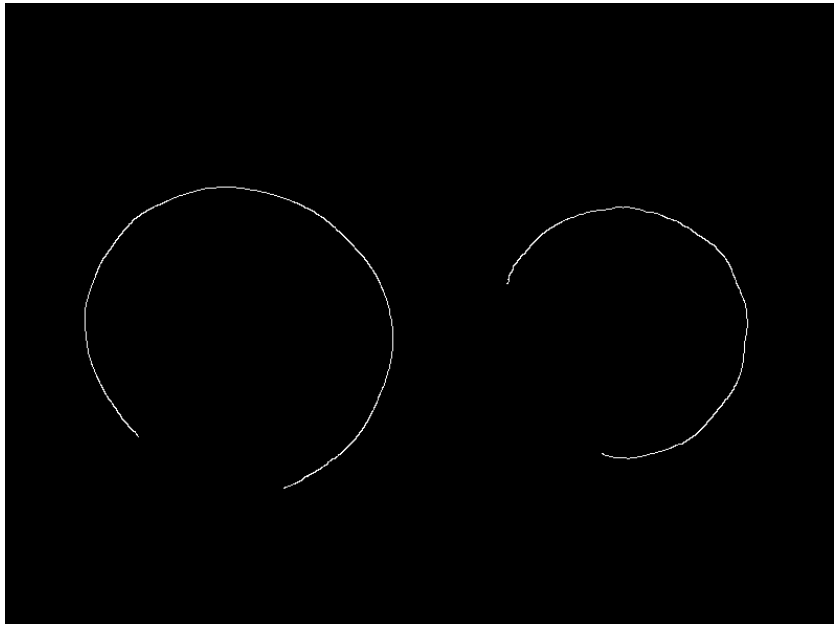
輪郭抽出

輪郭抽出－－方法２

```
import cv2 as cv

img = cv.imread('objects_rescaled.jpg') # 画像の読み込み
cv.imshow('Original', img)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY) # グレースケールに変換
blur = cv.GaussianBlur(gray, (3,3), cv.BORDER_DEFAULT) # 平滑化

canny = cv.Canny(blur, 40, 300) # エッジ検出
cv.imshow('Canny Edges', canny)

contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)
img_contour = cv.drawContours(img, contours, -1, (0, 255, 0), 5)
cv.imshow('Image with contour', img_contour)

cv.waitKey(0)
```

輪郭抽出結果２

元画像



エッジ検出

輪郭抽出

顔と目の認識（カスケード識別器）

分類機のダウンロード：
https://github.com/opencv/opencv/tree/master/data/haarcascades

下記のコードの中身をコピーして、xmlファイルを作成してください。

haarcascade_frontalface_default.xml
haarcascade_eye.xml

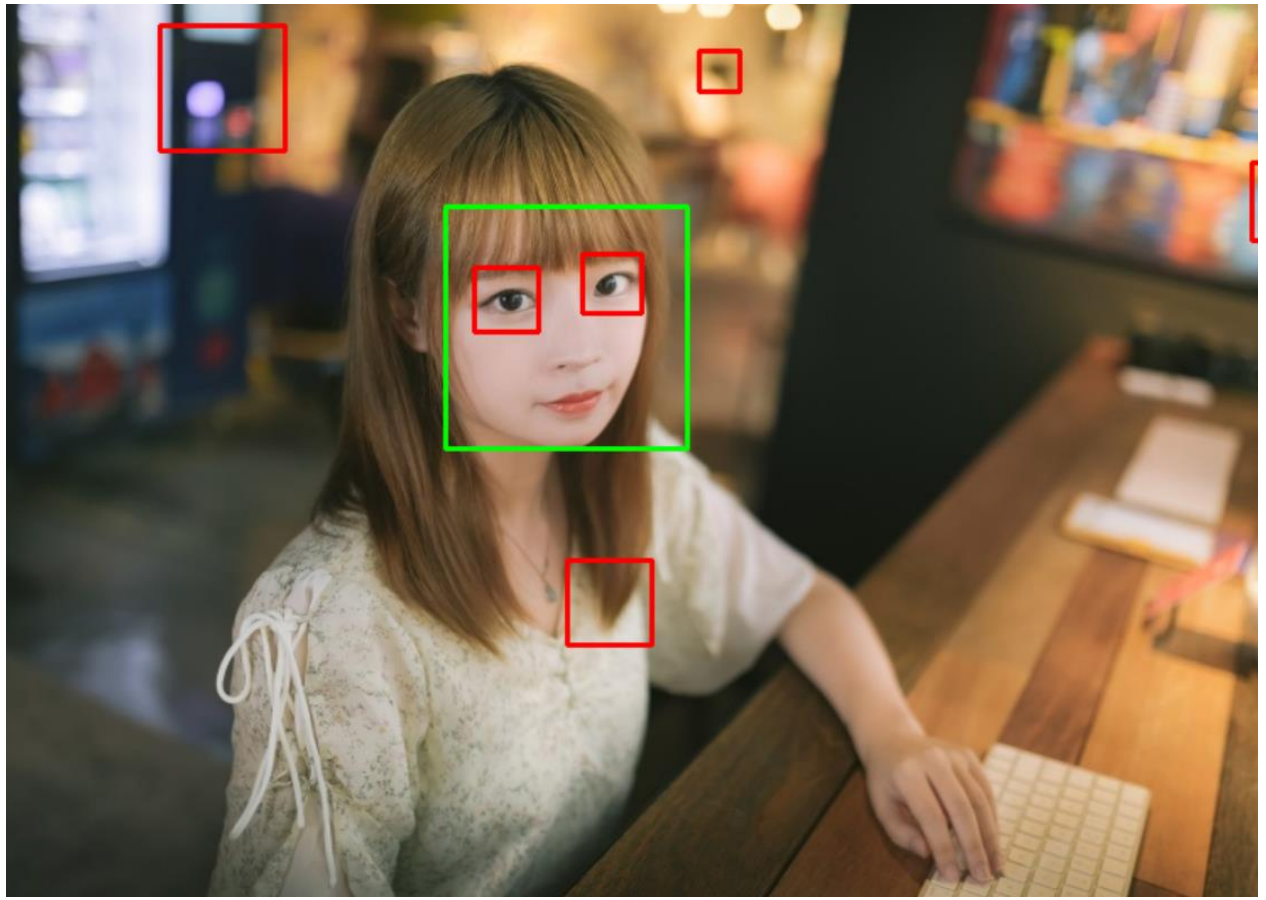
直接ダウンロードした場合、エラーが発生する可能性があります

写真のダウンロード：
https://www.pakutaso.com/20191158315post-24140.html

```python
import cv2 as cv

img = cv.imread('image5_resized.jpg')
cv.imshow('Original image', img)

face_cascade = cv.CascadeClassifier('haarcascade_frontalface_default.xml')  # 顔分類機導入
eye_cascade = cv.CascadeClassifier('haarcascade_eye.xml')  # 目分類機の導入

faces = face_cascade.detectMultiScale(img, scaleFactor=1.1, minNeighbors=3)  # 顔の認識
eyes = eye_cascade.detectMultiScale(img, scaleFactor=1.1, minNeighbors=5)  # 目の認識

for (x,y,w,h) in faces:
    cv.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)  # 緑色で顔の描画

for (ex,ey,ew,eh) in eyes:
    cv.rectangle(img, (ex,ey), (ex+ew,ey+eh), (0,0,255), 2)  # 赤色で目の描画

cv.imshow('Detected faces and eyes', img)

cv.waitKey(0)
```

顔と目の認識結果

元画像



顔と目の認識結果

14週目レポート

- 写真の1枚を用意して、適切なサイズに変更し、輪郭を抽出してみよう。また、抽出した輪郭を楕円でフィッティングし、表示してみよう。楕円での輪郭フィッティングと描画は下記のメソッドを利用してください。

```
ellipse = cv.fitEllipse(i)  # 楕円でフィッティング、iは輪郭
img_ellipse = cv.ellipse(img, ellipse, (0, 0, 255), 2)  # 画像imgに楕円の描画
```

- for文を利用してすべての輪郭をフィッティングし、描画してみよう。
- 輪郭の点数が 5 点より少ない場合、エラーが出る可能性がありますので、if文でエラーを避けてみよう。
- 写真とプログラムをmanaba+Rで提出してください