Q. 1 - 10

Q.1. チャネル入れ替え

画像を読み込み、RGBをBGRの順に入れ替えよ。

画像の赤成分を取り出すには、以下のコードで可能。 cv2.imread()関数ではチャネルがBGRの順になることに注意！ これで変数redにimori.jpgの赤成分のみが入る。

```
import cv2
img = cv2.imread("imori.jpg")
red = img[:, :, 2].copy()
```

入力 (imori.jpg)	出力 (answers_image/answer_1.jpg)

答え

- Python >> answers_py/answer_1.py
- C++ >> answers_cpp/answer_1.cpp

Q.2. グレースケール化

画像をグレースケールにせよ。 グレースケールとは、画像の輝度表現方法の一種であり下式で計算される。

Y = 0.2126 R + 0.7152 G + 0.0722 B

入力 (imori.jpg)	出力 (answers_image/answer_2.jpg)

答え

- Python >> answers_py/answer_2.py
- C++ >> answers_cpp/answer_2.cpp

Q.3. 二値化

画像を二値化せよ。 二値化とは、画像を黒と白の二値で表現する方法である。 ここでは、グレースケールにおいて閾値を128に設定し、下式で二値化する。

```
y = { 0 (if y < 128)
      255 (else)
```

入力 (imori.jpg)	出力 (answers_image/answer_3.jpg)

答え

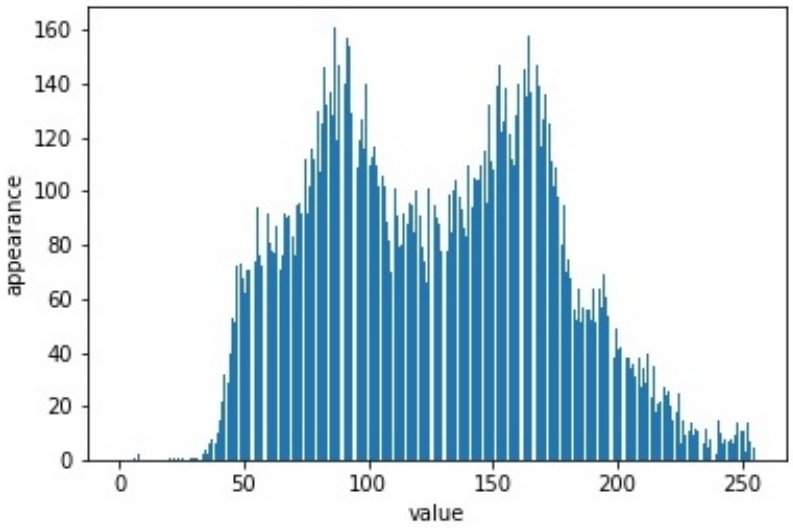
- Python >> answers_py/answer_3.py
- C++ >> answers_cpp/answer_3.cpp

Q.4. 大津の二値化

大津の二値化を実装せよ。 大津の二値化とは判別分析法と呼ばれ、二値化における分離の閾値を自動決定する手法である。 これは**クラス内分散**と**クラス間分散**の比から計算される。

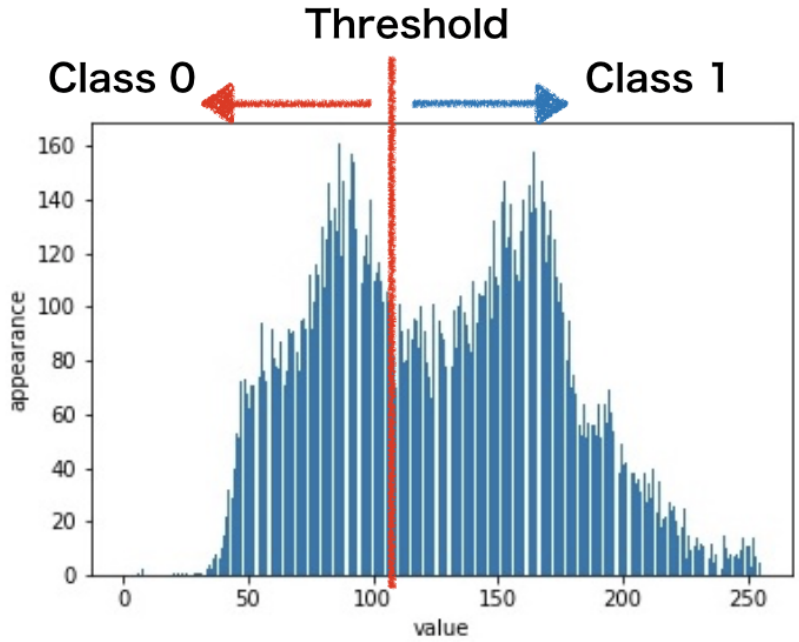
グレースケールの輝度値（ピクセルの値）のヒストグラムはこうなる。

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('assets/imori.jpg')
gray = 0.2126 * img[..., 2] + 0.7152 * img[..., 1] + 0.0722 * img[..., 0]
plt.hist(gray.ravel(), bins=255, rwidth=0.8, range=(0, 255))
plt.xlabel('value')
plt.ylabel('appearance')
plt.show()
```



二値化はある値を境界にして、０か１にする方法だけど、

- 閾値t未満をクラス0, t以上をクラス1とする。
- w0, w1 ... 閾値tにより分離された各クラスの画素数の割合 (w0 + w1 = 1を満たす)
- S0^2, S1^2 ... 各クラスの画素値の分散
- M0, M1 ... 各クラスの画素値の平均値



とすると、

クラス内分散 $S_w^2 = w_0 S_o^2 + w_1 S_1^2$

クラス間分散 $S_b^2 = w_0(M_0 - M_t)^2 + w_1(M_1 - M_t)^2$

$= w_0 w_1 (M_0 - M_t)^2$

画像全体の画素の分散 $S_t^2 = S_w^2 + S_b^2 = (const)$

となり、分離度Xは次式で定義される。

分離度 $$X = \frac{S_b^2}{S_b^2} = \frac{S_b^2}{S_t^2 - S_b^2}$$

となるので、

$$argmax_t\, X = argmax_t\, S_b^2$$

となる。すなわち、Sb^2 = w0 * w1 * (M0 - M1) ^2 が最大となる、閾値tを二値化の閾値とすれば良い。

入力 (imori.jpg)	出力 (th = 127) (answers_image/answer_4.jpg)

答え

- Python >> answers_py/answer_4.py
- C++ >> answers_cpp/answer_4.cpp

Q.5. HSV変換

HSV変換を実装して、色相Hを反転せよ。

HSV変換とは、**Hue(色相)**、**Saturation(彩度)**、**Value(明度)** で色を表現する手法である。

- Hue ... 色合いを0~360度で表現し、赤や青など色の種類を示す。 (0 <= H < 360) 色相は次の色に対応する。

```
赤 黄色  緑  水色  青  紫   赤
0  60  120  180 240 300 360
```

- Saturation ... 色の鮮やかさ。Saturationが低いと灰色さが顕著になり、くすんだ色となる。 (0<= S < 1)
- Value ... 色の明るさ。Valueが高いほど白に近く、Valueが低いほど黒に近くなる。 (0 <= V < 1)

RGB -> HSV変換は以下の式で定義される。

R,G,Bが[0, 1]の範囲にあるとする。

```
Max = max(R,G,B)
Min = min(R,G,B)

H =  { 0                                (if Min=Max)
```

```
        60 x (G-R) / (Max-Min) + 60  (if Min=B)
        60 x (B-G) / (Max-Min) + 180 (if Min=R)
        60 x (R-B) / (Max-Min) + 300 (if Min=G)

V = Max

S = Max - Min
```

HSV -> RGB変換は以下の式で定義される。

```
C = S

H' = H / 60

X = C (1 - |H' mod 2 - 1|)

(R,G,B) = (V - C) (1,1,1) + { (0, 0, 0)  (if H is undefined)
                              (C, X, 0)  (if 0 <= H' < 1)
                              (X, C, 0)  (if 1 <= H' < 2)
                              (0, C, X)  (if 2 <= H' < 3)
                              (0, X, C)  (if 3 <= H' < 4)
                              (X, 0, C)  (if 4 <= H' < 5)
                              (C, 0, X)  (if 5 <= H' < 6)
```

ここでは色相Hを反転(180を加算)し、RGBに直し画像を表示せよ。

入力 (imori.jpg)	出力 (answers_image/answer_5.jpg)
	

答え

- Python >> answers_py/answer_5.py
- C++ >> answers_cpp/answer_5.cpp

Q.6. 減色処理

ここでは画像の値を256^3から4^3、すなわちR,G,B in {32, 96, 160, 224}の各4値に減色せよ。 これは量子化操作である。 各値に関して、以下の様に定義する。

```
val = {  32  (  0 <= val <  64)
         96  ( 64 <= val < 128)
```

```
          160  (128 <= val < 192)
          224  (192 <= val < 256)
```

入力 (imori.jpg)	出力 (answers_image/answer_6.jpg)

答え

- Python >> answers_py/answer_6.py
- C++ >> answers_cpp/answer_6.cpp

Q.7. 平均プーリング

ここでは画像をグリッド分割(ある固定長の領域に分ける)し、かく領域内(セル)の平均値でその領域内の値を埋める。このようにグリッド分割し、その領域内の代表値を求める操作は**Pooling(プーリング)** と呼ばれる。 これらプーリング操作は**CNN(Convolutional Neural Network)** において重要な役割を持つ。

これは次式で定義される。

```
v = 1/|R| * Sum_{i in R} v_i
```

ここではimori.jpgは128x128なので、8x8にグリッド分割し、平均プーリングせよ。

入力 (imori.jpg)	出力 (answers_image/answer_7.jpg)

答え

- Python >> answers_py/answer_7.py
- C++ >> answers_cpp/answer_7.cpp

Q.8. Maxプーリング

ここでは平均値でなく最大値でプーリングせよ。

入力 (imori.jpg) **出力 (answers_image/answer_8.jpg)**

答え

- Python >> answers_py/answer_8.py
- C++ >> answers_cpp/answer_8.cpp

Q.9. ガウシアンフィルタ

ガウシアンフィルタ(3x3、標準偏差1.3)を実装し、*imori_noise.jpg*のノイズを除去せよ。

ガウシアンフィルタとは画像の**平滑化**（滑らかにする）を行うフィルタの一種であり、**ノイズ除去**にも使われる。

ノイズ除去には他にも、メディアンフィルタ(Q.10)、平滑化フィルタ(Q.11)、LoGフィルタ(Q.19)などがある。

ガウシアンフィルタは注目画素の周辺画素を、ガウス分布による重み付けで平滑化し、次式で定義される。 このような重みは**カーネル**や**フィルタ**と呼ばれる。

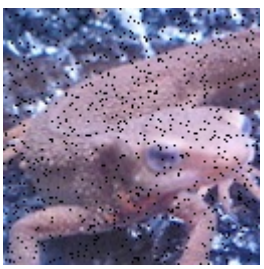
ただし、画像の端はこのままではフィルタリングできないため、画素が足りない部分は0で埋める。これを**0パディング**と呼ぶ。 かつ、重みは正規化する。(sum g = 1)

重みはガウス分布から次式になる。

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

```
重み g(x,y,s) = 1/ (2 * pi * sigma * sigma) * exp( -(x^2 + y^2) / (2*s^2))
標準偏差s = 1.3による8近傍ガウシアンフィルタは
            1 2 1
K =  1/16 [ 2 4 2 ]
            1 2 1
```

入力 (imori_noise.jpg) **出力 (answers_image/answer_9.jpg)**



答え

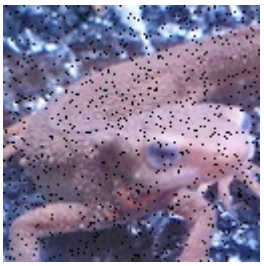
- Python >> answers_py/answer_9.py
- C++ >> answers_cpp/answer_9.cpp

Q.10 メディアンフィルタ

メディアンフィルタ(3x3)を実装し、*imori_noise.jpg*のノイズを除去せよ。

メディアンフィルタとは画像の平滑化を行うフィルタの一種である。

これは注目画素の3x3の領域内の、メディアン値(中央値)を出力するフィルタである。 これもゼロパディングせよ。

入力 (imori_noise.jpg)	出力 (answers_image/answer_10.jpg)
	

答え

- Python >> answers_py/answer_10.py
- C++ >> answers_cpp/answer_10.cpp