

人工知能 プロジェクト実習 AI-2 (第6回)

1

コンピュータサイエンス学部

青木 輝勝

Email: aokitms@stf.teu.ac.jp

本日の内容

- 第6回：画像のコントラスト改善
 - 6.1 前回の復習と補足
 - 6.2 代表的なコントラスト改善手法
 - 6.2.1 ヒストグラム均一化
 - 6.2.2 ガンマ補正
 - 6.2.3 その他の手法
 - 6.3 本日の課題

本日の内容

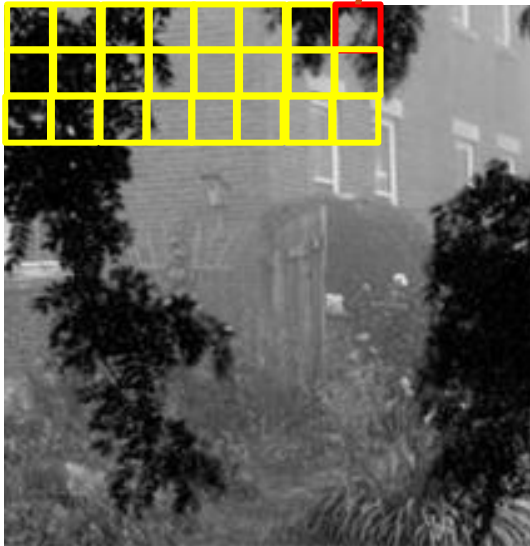
- 第6回：画像のコントラスト改善
 - 6.1 前回の復習と補足
 - 6.2 代表的なコントラスト改善手法
 - 6.2.1 ヒストグラム均一化
 - 6.2.2 ガンマ補正
 - 6.2.3 その他の手法
 - 6.3 本日の課題

DCPのアルゴリズム (3/8)

③ $L' = \min_{\Omega} \min_{c \in \{R, G, B\}} I(x, y)^c$ を満たす画像 L' を生成する.
 $= K'$ * Ω は $m \times n$ の画像パッチを表す.

(1) 画像 K' に対し, 画像パッチ Ω (例: 8×8 , 16×16 など) 毎に, Ω 内で最小となる画素値を求める.

画像 K'



22	21	19	16
46	44	15	17
47	45	27	19
51	54	52	54

最小値 = 15

DCPのアルゴリズム (4/8)

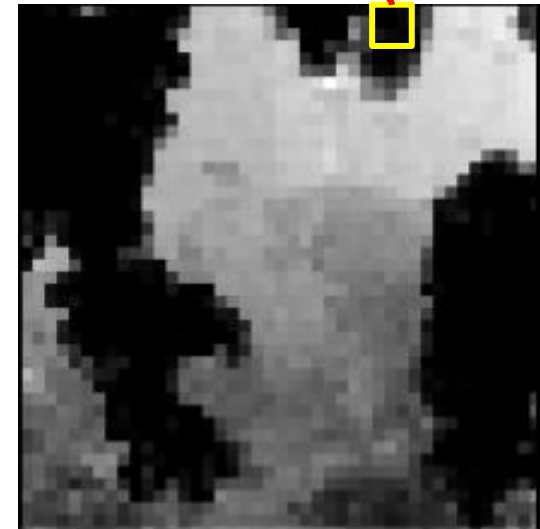
③ $L' = \min_{\Omega} \min_{c \in \{R, G, B\}} I(x, y)^c$ を満たす画像 L' を生成する.
 $= K'$ * Ω は $m \times n$ の画像パッチを表す.

(2) 画像パッチ Ω 毎に, Ω 内のすべての画素を最小値で埋める.

22	21	19	16
46	44	15	17
47	45	27	19
51	54	52	54



15	15	15	15
15	15	15	15
15	15	15	15
15	15	15	15

画像 L'

処理③のアルゴリズム(1) (1/2)

■ 4重for文

```
for y in range(0, height, 8):  
    for x in range(0, width, 8):
```

x, y は必ずパッチの
左上の画素を指す

```
        for j in range(0, 8):  
            for i in range(0, 8):
```

パッチ内の全画素
をスキャンする

```
                if((x+i < width) and (y+j < height)):  
                    最小値を求める処理
```

```
            for n in range(0, 8):  
                for m in range(0, 8):
```

x+i は width
より大きくなる
場合がある
(y+jも同様).

```
                    if((x+i < width) and (y+j < height)):  
                        最小値を各画素に埋める処理
```

処理③のアルゴリズム(1) (2/2)

- 最小値を求める処理 / 最小値を埋める処理

```
for y in range(0, height, 8):  
    for x in range(0, width, 8):
```

```
        min = 255
```

```
        for j in range(0, 8):  
            for i in range(0, 8):
```

```
                if((x+i < width) and (y+j < height)):
```

```
                    if(K_img[y+j, x+i] < min):
```

```
                        min = K_img[y+j, x+i]
```

```
        for n in range(0, 8):  
            for m in range(0, 8):
```

```
                if((x+i < width) and (y+j < height)):
```

```
                    L_img[y+j, x+i] = min
```

最小値を
求める処理

最小値を
埋める処理

処理③のアルゴリズム(2)(3)

- pythonらしいエレガントな方法

```
for j in range(0, height, 8):  
    for i in range(0, width, 8):  
        L_img[j : j+8, i : i+8] = K_img[j : j+8, i : i+8].min()
```

- append()を用いる方法 (→あまりお勧めしない)

```
for y in range(0, height, 8):  
    for x in range(0, width, 8):  
  
        a = [ ]  
        for j in range(0, 8):  
            for i in range(0, 8):  
  
                if((x+i < width) and (y+j < height)):  
                    p = K_img[y+j, x+i]  
                    a.append(p)  
  
        min_value = min(a)
```


全般的な注意事項 (1/2)

- Jupyter Notebookを使っているので, 可能な限りプログラムは処理毎に別のセルを使うこと.
- 処理中のデータが**1チャンネル画像(グレースケール画像)**なのか**3チャンネル画像(カラー画像)**なのかを常に意識すること.
- 画像でないデータ形式は可能な限り使わないほうが良い.
 - 例) 空のリストを用意し, `append()` でデータを追加してゆく etc.
- `for`文の中で, `range(0, height - 3, 8)` や `range(7)` など論理不明な記述が見られるのは残念.

全般的な注意事項 (2/2)

- 画素値が0～255の範囲にあることの確認(画素値のオーバーフローの確認)が正しくないプログラムが散見される.
 - 下記プログラム中の $a*b+c$ は深い意味はなく「何らかの演算」を表す).

正しくない処理

```
for y in range(0, height):  
    for x in range(0, width):  
  
        output_img[y,x,0] = a*b+c  
  
        if(output_img[y,x,0] > 255):  
            output_img[y,x,0] = 255  
        elif(output_img[y,x,0] < 0):  
            output_img[y,x,0] = 0
```

この処理の時点で、オーバーフロー
(例: $257 \rightarrow 1$)が生じてしまっている.

正しい処理

```
for y in range(0, height):  
    for x in range(0, width):  
  
        p = a*b+c  
  
        if(p > 255.0):  
            output_img[y,x,0] = 255  
        elif(p < 0.0):  
            output_img[y,x,0] = 0  
        else:  
            output_img[y,x,0] = int(p)
```

本日の内容

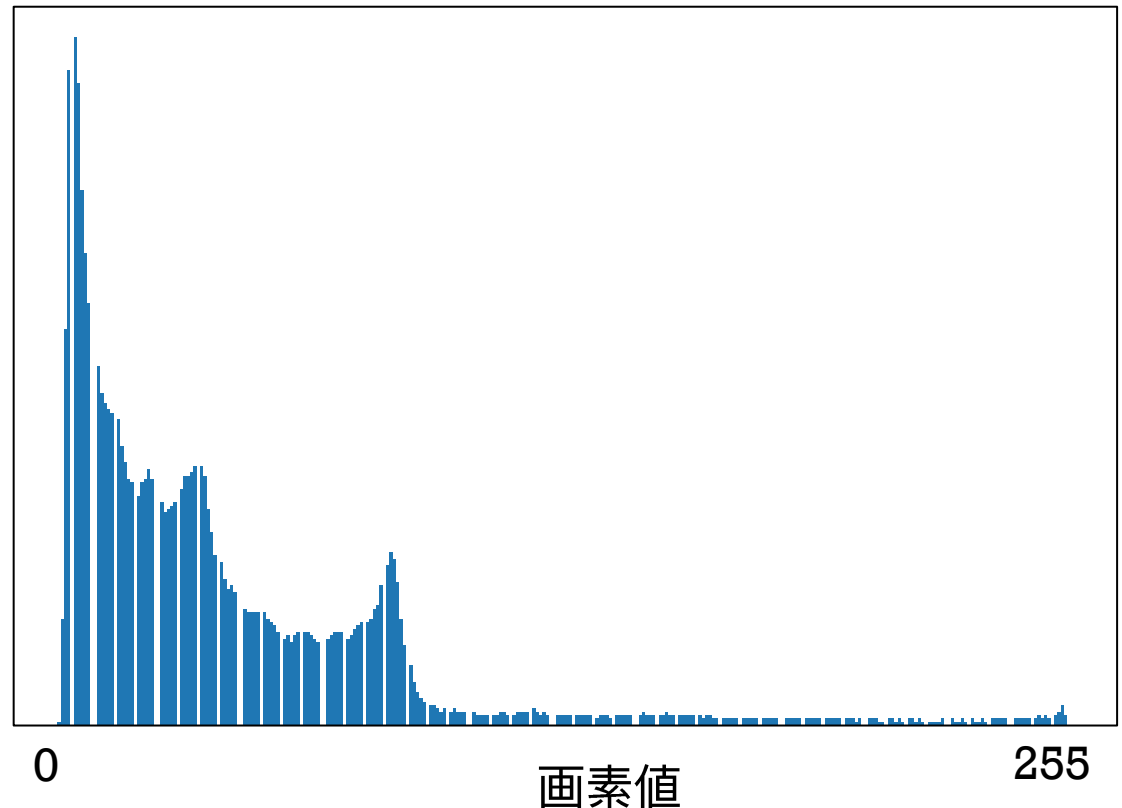
- 第6回：画像のコントラスト改善
 - 6.1 前回の復習と補足
 - 6.2 代表的なコントラスト改善手法
 - 6.2.1 ヒストグラム均一化
 - 6.2.2 ガンマ補正
 - 6.2.3 その他の手法
 - 6.3 本日の課題

ヒストグラム均一化とは (1/2)

- (画像の)ヒストグラム
 - 画像内で各値(0~255)を持つ画素数をグラフ化したもの。

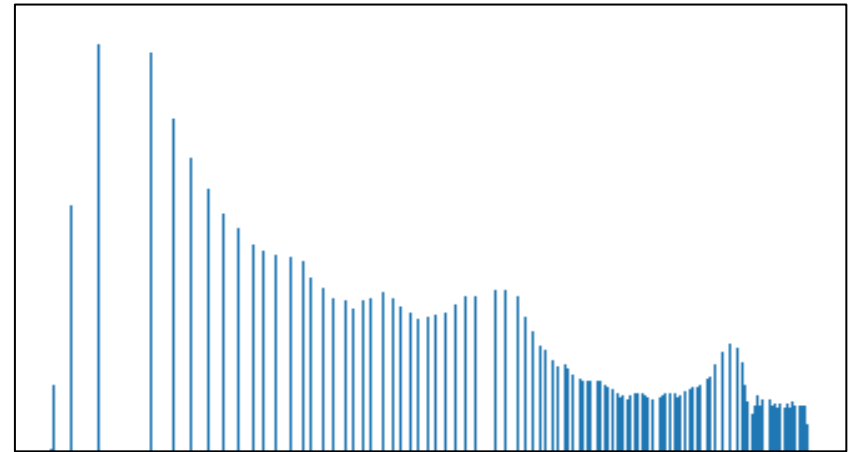
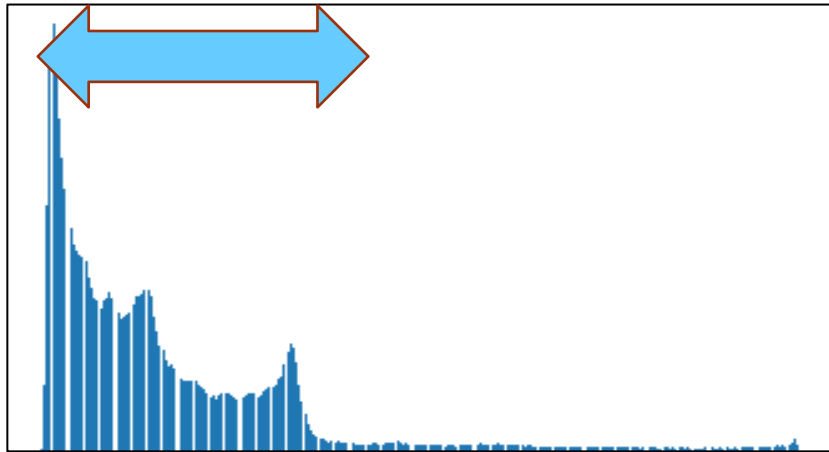


画素数



ヒストグラム均一化とは (2/2)

- ヒストグラム均一化
 - ヒストグラムを最適に横に伸ばす.



簡単そうに見えるが、実際は様々な問題があり、現在なお多くの研究が行われている[1].



[1] Sergei Yelmanov and Yuriy Romanyshyn, "Image Contrast Enhancement Using a Modified Histogram Equalization", IEEE International Conference on Data Stream Mining & Processing (DSMP), Aug., 2018.

ヒストグラムの生成

- OpenCVの `calcHist()` という関数を用いると良い.

```
hist = cv2.calcHist([画像名], [チャンネル名],  
                    マスク画像, [ビン数], [画素値の範囲])
```

画像名：ヒストグラムを計算する画像の指定

チャンネル名：グレイ画像の場合には[0], カラー画像の場合にはどのチャンネルのヒストグラムを取得するかによって[0][1][2]のいずれか.

マスク画像：マスク画像を指定すると, そのマスク内のヒストグラムを算出する. 多くの場合 None (マスク画像なし).

ビン数：ビン(ヒストグラムの縦線1本)の数. 多くの場合 256.

画素値の範囲：多くの場合 [0, 256]

出力の `hist`：ビン毎の出力値を格納した1次元配列 (=画素値0~255に対する画素数が格納されており, `hist[i]`で画素値 `i` に対する画素数が取り出せる)

ヒストグラムの表示

- matplotlib.pyplot の機能を使うのが便利.

- 折れ線グラフ

```
plt.plot(hist)
```

- 棒グラフ

```
plt.bar(data_x, data_y)
```

data_x : x座標を格納した1次元配列

data_y : y座標を格納した1次元配列

ヒストグラムの生成・表示の プログラム例

(前略)

```
img = cv2.imread("/content/drive/MyDrive/ColabNotebooks/testing.jpg")
```

```
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

#ヒストグラムの取得

```
hist = cv2.calcHist([gray_img], [0], None, [256], [0,256])
```

#ヒストグラムを棒グラフとして表示

```
data_x = np.zeros(256)
```

```
data_y = np.zeros(256)
```

```
for i in range(0, 256):
```

```
    data_x[i] = i
```

```
    data_y[i] = hist[i]
```

```
plt.bar(data_x, data_y)
```

* cv2.calcHist() 内のパラメータの [] を書き忘れないこと.

ヒストグラム均一化の実現

- ヒストグラム均一化には様々な手法があり, 現在もなお開発が続けられているが, 最もシンプルな手法についてはOpenCVで関数が用意されている.

```
he_img = cv2.equalizeHist(input_img)
```

input_img : 入力画像

he_img : ヒストグラム均一化された出力画像



input_img



he_img

本日の内容

- 第6回：画像のコントラスト改善
 - 6.1 前回の復習と補足
 - 6.2 代表的なコントラスト改善手法
 - 6.2.1 ヒストグラム均一化
 - 6.2.2 ガンマ補正
 - 6.2.3 その他の手法
 - 6.3 本日の課題

ガンマ補正とは(1/2)

■ ガンマ補正

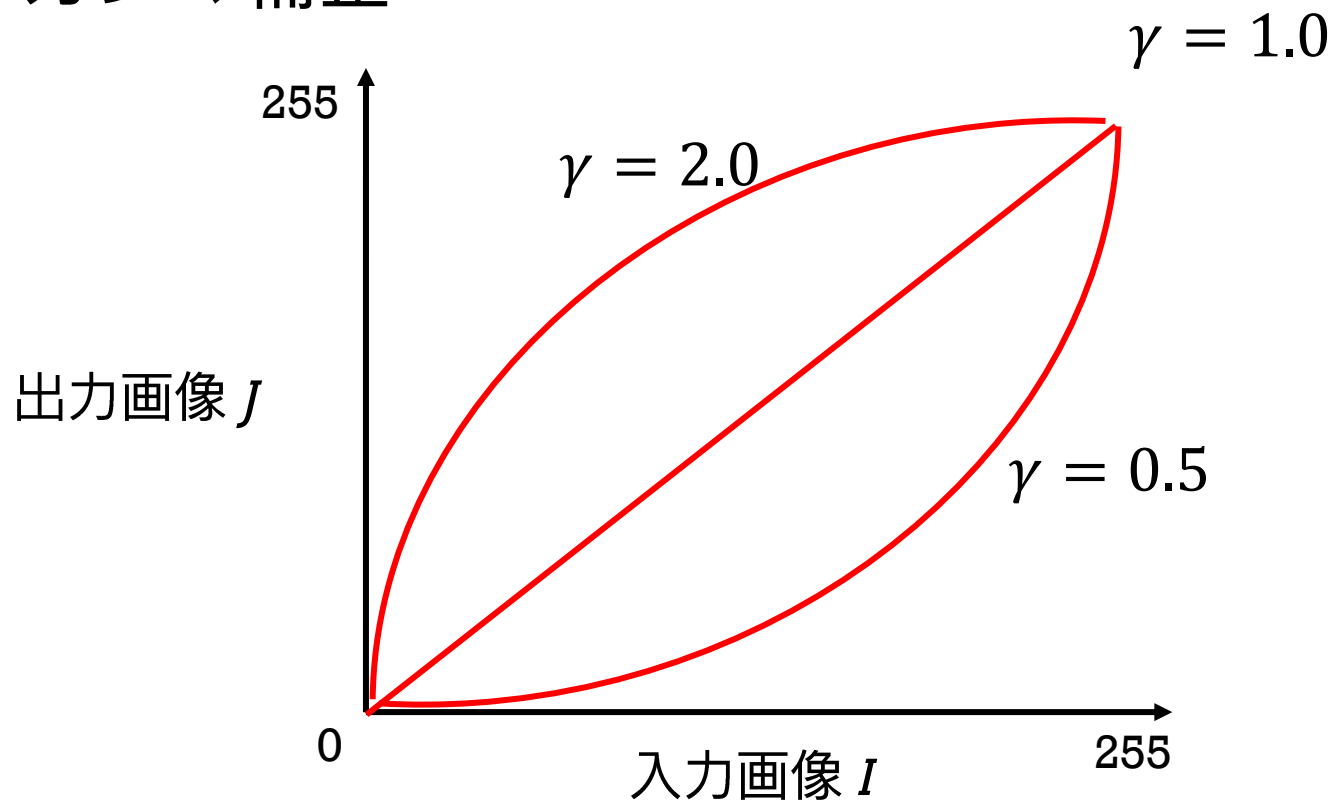
- 入力画像 I の画素値と出力画像 J の画素値を次の式によって算出する手法.

$$J[y, x] = 255 \times \left(\frac{I[y, x]}{255} \right)^{\frac{1}{\gamma}}$$

- $\gamma = 1.0$ のとき変化なし. $\gamma > 1.0$ のとき出力画像は明るくなり, $\gamma < 1.0$ のとき出力画像は暗くなる.
- 多くの場合, 入力画像のすべての画素値 (すなわち 0~255) に対応する出力画像の画素値をあらかじめ算出して表(ルックアップテーブル)にまとめておき, 演算時にはこの表を参照して演算後の画素値を得ている.

ガンマ補正とは (2/2)

■ ガンマ補正



ルックアップテーブル

I	0	1	2	3	...	255
J	0	16	23	28	...	255

ガンマ補正の実現

■ ルックアップテーブルの生成

```
gamma = 2.0  
lookup_table = np.zeros(256, np.float32)  
  
for i in range(0, 256):  
    lookup_table[i] = 255.0*pow(float(i)/255.0, 1.0/gamma)
```

lookup_table : ルックアップテーブル (1次元配列, 値は実数)

■ ガンマ補正画像の出力

■ 1チャンネル画像(グレイ画像)の場合

```
output_gray = cv2.LUT(input_gray, lookup_table)
```

■ 3チャンネル画像(BGR画像)の場合 (Rチャンネルをガンマ補正)

```
output_color[:, :, 2] = cv2.LUT(input_color[:, :, 2], lookup_table)
```

*output_colorは入力画像input_colorと同一内容のものを定義しておく必要がある。

本日の内容

- 第6回：画像のコントラスト改善
 - 6.1 前回の復習と補足
 - 6.2 代表的なコントラスト改善手法
 - 6.2.1 ヒストグラム均一化
 - 6.2.2 ガンマ補正
 - 6.2.3 その他の手法
 - 6.3 本日の課題

その他の手法 (1/3)

- 先週扱ったDCPを用いたコントラスト改善手法が提案されている[2]. その概要は下記の通り.
 - 処理①: 暗い写真 I をネガポジ反転させる(画像中の全画素 p_{ij} を $255 - p_{ij}$ で置き換える).すると, ネガポジ反転画像 J は「霧が濃い画像」とみなすことができるようになる.
 - 処理②: 画像 J にDCPをかけて霧を取り除く(出力画像を K とする).
 - 処理③: 画像 K を再度ネガポジ反転すると, その出力は画像 I をコントラスト改善した写真となっている.

その他の手法 (2/3)

処理②

画像 J

画像 K

$$= t + (1 - t)$$

霧 A

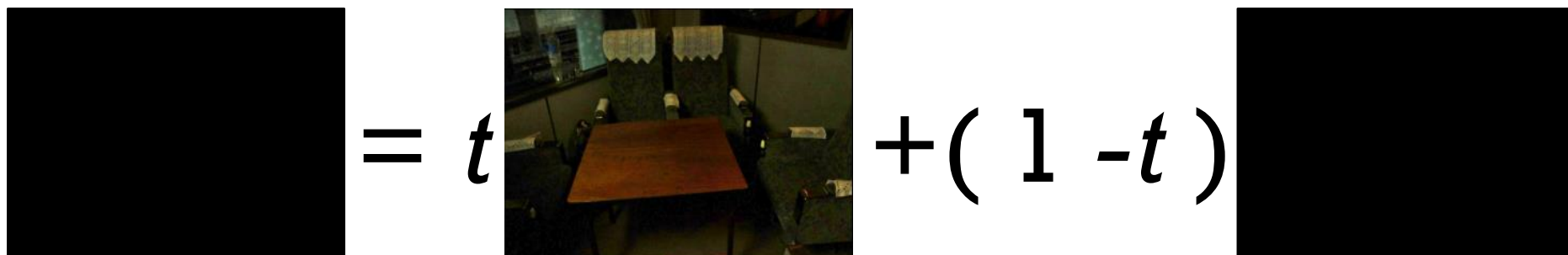
処理①

処理③

入力画像 I

出力画像

その他の手法 (3/3)


$$\text{入力画像 } I = t \cdot \text{出力画像} + (1 - t) \cdot \text{黒い霧 } A'$$

入力画像 I 出力画像 黒い霧 A'

- 別の考え方

“入力画像 I は 高コントラスト画像(出力画像)と黒い霧 A' を $t : 1 - t$ で混ぜあわせた画像”

本日の内容

- 第6回：画像のコントラスト改善
 - 6.1 前回の復習と補足
 - 6.2 代表的なコントラスト改善手法
 - 6.2.1 ヒストグラム均一化
 - 6.2.2 ガンマ補正
 - 6.2.3 その他の手法
 - 6.3 本日の課題

課題0601

- 実験画像をグレースケール画像に変換した後, ヒストグラム均一化処理を行うプログラムを作成せよ.
- ただし, プログラム内で入力画像・出力画像(いずれもグレースケール画像)のヒストグラムを作成し表示すること.

課題0602

- ガンマ補正を用いてコントラスト改善を行うプログラムを作成し、実験画像から出力画像(高コントラスト画像)を生成するプログラムを作成せよ.
- ただし、ガンマ補正を行うにあたっては、まず入力画像(BGR画像)をHSV画像*に変換し、HSV画像のVチャンネルのみをガンマ補正すること.

* H: Hue (色相), S: Saturation (彩度), V: Value(明度)

[注意]

- 出力画像の出来栄も評価の対象とする.つまり、出力画像毎に最適となるようにパラメータ(γ)を調整すること.
- 可能であれば、プログラム内で入力画像・出力画像それぞれのVチャンネルのヒストグラムを表示することが望ましい.

課題0603

- DCP法を用いてコントラスト改善を行うプログラムを作成し、実験画像から出力画像(高コントラスト画像)を生成するプログラムを作成せよ。

[注意]

- DCPについては課題0501で作成したプログラムを流用してもよい。
- 出力画像の出来栄も評価の対象とする。つまり、出力画像毎に最適となるように各種パラメータを調整すること。
- 可能であれば、プログラム内で入力画像と出力画像をグレースケールに変換した後、それらのヒストグラムを生成し表示することが望ましい。

本日のレポート課題

- 課題0601, 課題0602, 課題0603について下記の通り提出すること.
- 提出物・提出先・提出期限
 - 提出物：
 - すべての課題に対し, **jupyter notebook ファイル (.ipynb)**と(5枚の実験画像に対する)**出力画像5枚**.
 - 提出期限：
 - 6月1日(木) 21:00
 - 提出先：
 - moodleの本講義のページ