レポート提出票

科目名: 情報工学実験３

実験課題名: 課題４　画像変換

実施日: 2024年 7月 4日

学籍番号: 4622045

氏名: 小澤　翼

共同実験者:

1　要旨

Jupyter Notebook や Python のライブラリの使い方を習得し、濃淡変換の実装を通じて画像処理の基本を学んだ。

2　目的

濃淡変換の実装と実験を通じて、Python 言語とそのライブラリの使用方法、および画像処理の基本を理解する。

3　課題 1

3.1　実験方法

以下の手順でプログラムを作成しなさい。ただし、for 文，numpy，matplotlib は使ってもよい。skimage ライブラリ関数は画像読み込み以外では使わないこととする。

1. 画像ファイル“data/cat.jpg”を読み込み画像 im1 とする
2. 画像 im1 の指定の領域内の画素について、R 値 (赤) が 80 以上のとき、指定の変換を行い画像 im2 を作成する。
   - 指定の領域：2 点 (185, 200), (225, 230) を対角とする矩形
   - 指定の変換：R 値 (赤) を 0 に置換．G 値 (緑) を 30 増やす，B 値 (青) を 90 減らす
3. 画像 im2 の猫を鏡写しになるように 2 匹表示させた画像 im3（高さ 512 ×幅 512）を作成する
4. 画像 im1、im2、im3 にタイトルをつけて、横に並べて表示する

3.2　実験結果

実行結果は以下の図 1 のようになった。

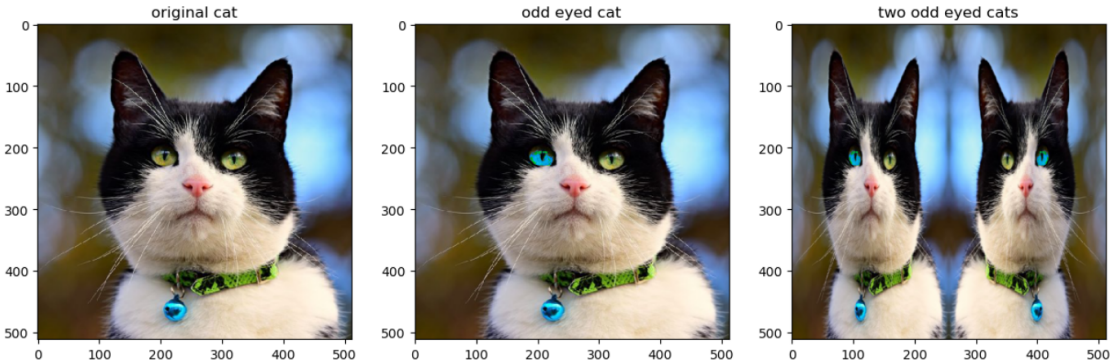

図 1: 課題 1 の結果

3.3 検討・考察

im3 を作成する際に、以下のコードを使用することで画質を保つようにした。

```
1  im2_resized = resize(im2, (512, 256), anti_aliasing=True, preserve_range=
       True).astype(np.uint8)
```

これにより、リサイズ後の画像が滑らかになり、品質が向上している。

4 課題 2

4.1 実験方法

2 枚の画像 $f, \mathbf{g}$ 間の平均二乗誤差 (Mean Squared Error; MSE) は次のように定義される：
(a) グレースケール画像の場合は,

$$\mathrm{MSE}(f,g) = \frac{1}{WH}\sum_{j=0}^{H-1}\sum_{i=0}^{W-1}\left(f(i,j)-g(i,j)\right)^2,$$

(b) カラー画像の場合は,

$$\mathrm{MSE}(f,g) = \frac{1}{|C|WH}\sum_{c\in C}\sum_{j=0}^{H-1}\sum_{i=0}^{W-1}\left(f(i,j,c)-g(i,j,c)\right)^2.$$

ここで， $C=\{R,G,B\}$ はチャンネルを表す．つまり，画像 f と g の差分を，画素ごとに求めて，その二乗平均をとったものが MSE の値となる．したがって，画像 f, g が完全に一致すれば $\mathrm{MSE}=0$ となり，差分が大きければ MSE が大きな値をとる．

1. 平均二乗誤差を求める関数 mse() を作成しなさい．配列要素の平均を求める Numpy の関数 np.mean() を使用するとよい．以下のコードを実行して，作成した関数が正しく動作していることを確認しなさい．

```
1 f = np.ones((10, 10), dtype=np.float)
2 g = np.ones((10, 10), dtype=np.float)
3 print(mse(f, g)) # --> 0.0
4 print(mse(f, g+2)) # --> 4.0
5 g[0,0]=0
6 print(mse(f, g)) # --> 0.01 = 1/(10*10)
```

2. 以下のコードを実行すると，意図しない結果が得られた．その理由を説明しなさい．

```
1 f = imread('data/apple.jpg')
2 print(mse(f, f)) # --> 0 OK
3 print(mse(f, f+10)) # --> 100 OK
4 print(mse(f, f+16)) # --> 0 ???
```

4.2　実験結果

1,2 の結果は以下のようになった。

```
1 mse(f, g)= 0.0
2 mse(f, g+2)= 4.0
3 mse(f, g)= 0.01
4 mse(f, f)= 0.0
5 mse(f, f+10)= 100.0
6 mse(f, f+16)= 0.0
```

4.3　検討・考察

意図しない結果が得られた箇所は mse(f, f+16) である。普通ならば $16^2 = 256$ と得られるはずだが、0.0 として結果が得られた。この理由として考えられるのは、画像データの画素値は８ビットまでで表されており整数値だと 0 ～ 255 までででしか表されていない。なので、256 は二進数で 100000000 の 9 ビットで表されてしまい、この 8 ビットまでの 00000000 が反映されて 0.0 という結果が得られてしまったのではないかと考える。

5　課題３

5.1　実験方法

解読対象の透かし埋め込み画像を下図（左）に示す。明るさの異なる 4 つの領域から構成されており、埋め込まれたメッセージを目視で読み取ることはできない。しかし、このグレースケール画像を水平な直線 (y=240) で切って 画素値の変化を観察してみると, 文字領域に対応する画素値（4 本の赤い矢印が指している箇所）がわずかに高く（明るく）なっていることが分かる。このわずかな輝度差を強調すれば、右図に重畳したような文字列 (TUS) を浮かび上がらせることができる。

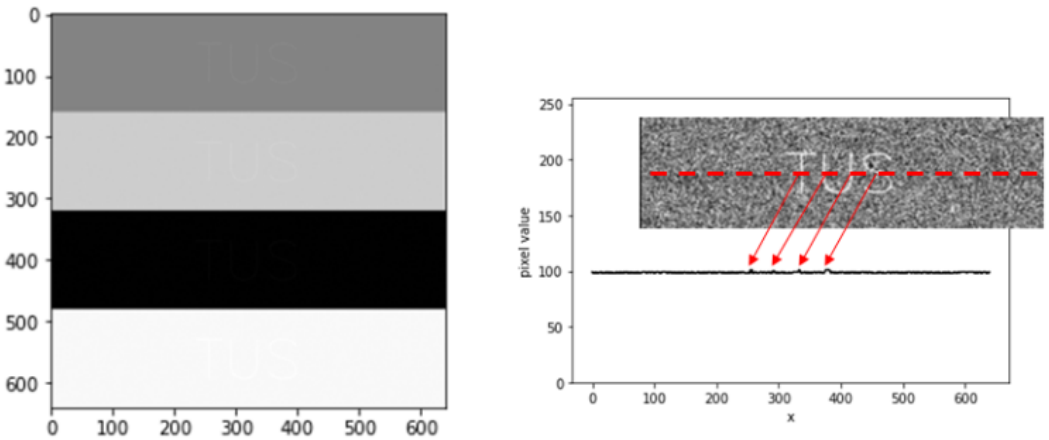

図 2: 透かし埋め込み画像

以上の性質を利用して、画像に埋め込まれたメッセージを解読する Python コードを作成せよ。scikit-image, numpy のライブラリを使用してもよい。

5.2 実験結果

実行結果は以下の図 3 のようになった。

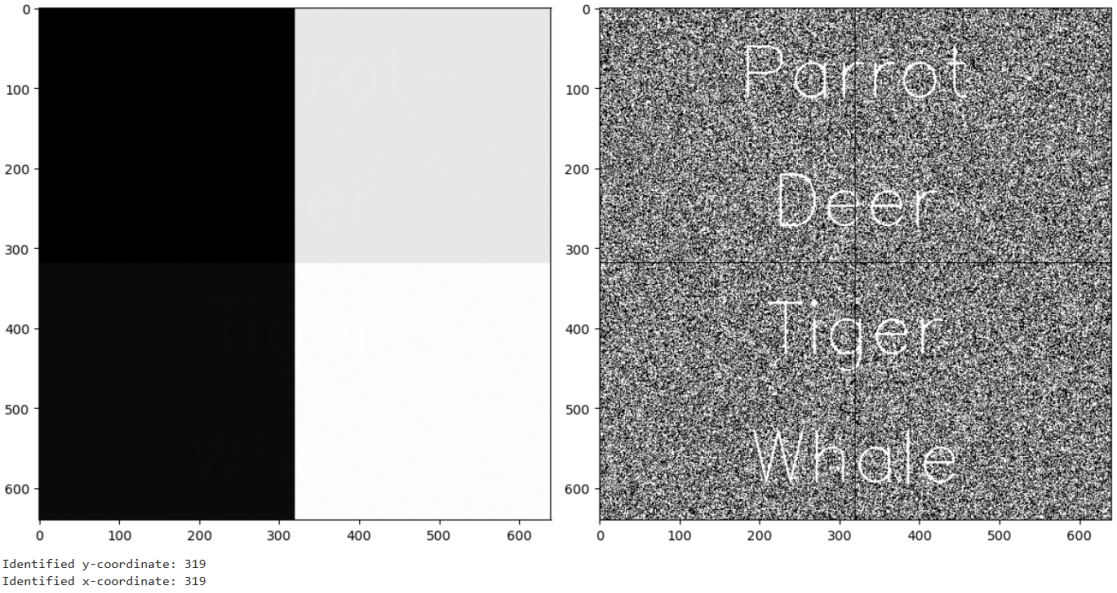

図 3: 透かし埋め込み画像

5.3 検討・考察

- **ソースコードについて：**画像を目視で見ると 3(4) 分割しているのが分かるので、その分割されている x,y 座標を求めるために課題 2 で行った画素値の平均二乗誤差を利用して分割されている x,y 座標を探した。これで得られた x,y 座標を基に二値化し、埋め込まれたメッセージを解読した。

- **パラメータを変化：**図 3 で二値化したパラメータを右上は変化させず、右下は+1、左上は+2、左下は+3 変化させると以下の図 4 となった。

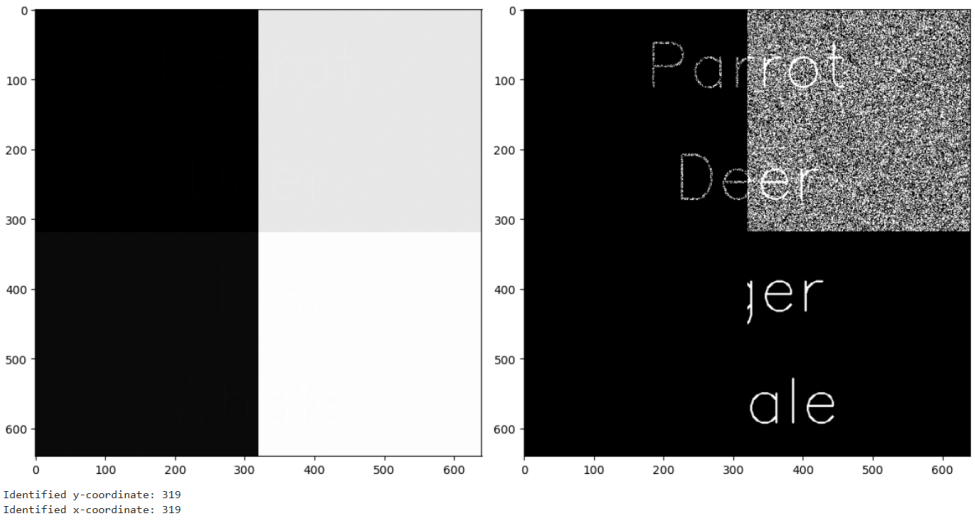

図 4: 図 3 からパラメータを変化

このことから浮かび上がった文字が一番見やすい箇所は右下であるので、図 3 を二値化したパラメータを+1 させたパラメータが一番適した数値と分かる。

6　まとめ

目的である濃淡変換の実装と実験を通じて、Python 言語とそのライブラリの使用方法、および画像処理の基本を理解することが出来たと感じた。

A　ソースコード

課題 1：画像処理プログラミング

```
from skimage.io import imread, imsave
import numpy as np
import matplotlib.pyplot as plt

# 1. 画像ファイルを読み込み
im1 = imread('data/cat.jpg')

#im1 をコピー
im2 = im1.copy()

#指定領域内で画素値変更
for i in range(200, 230):
    for j in range(185, 225):
        if im2[i, j, 0] >= 80: # R 値が 80 以上の場合
            im2[i, j, 0] = 0 # R 値を 0 に置換
            im2[i, j, 1] = im2[i, j, 1] + 30 # G 値を 30 増やす
            im2[i, j, 2] = im2[i, j, 2] - 90 # B 値を 90 減らす

# 3. 画像im2 を鏡写しにして 2 匹表示させた画像 im3 を作成
im3 = np.zeros((512, 512, 3), dtype=np.uint8)
im2_resized = resize(im2, (512, 256), anti_aliasing=True, preserve_range=
    True).astype(np.uint8)
im3[:512, :256] = im2_resized
im3[:512, 256:] = im2_resized[:, ::-1]

# 4. 画像im1, im2, im3 にタイトルをつけて表示
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.imshow(im1)
plt.title('original cat')

plt.subplot(1, 3, 2)
plt.imshow(im2)
plt.title('odd eyed cat')

plt.subplot(1, 3, 3)
plt.imshow(im3)
```

```
plt.title('two odd eyed cats')

plt.show()
```

課題 2：Mean Squared Error (MSE)

```
import numpy as np
from skimage.io import imread, imsave
import matplotlib.pyplot as plt
%matplotlib inline

def mse(f, g):
    # ↓ここにコードを追加 (1行)
    return np.mean((f -g) ** 2)

f = np.ones((10, 10), dtype=float)
g = np.ones((10, 10), dtype=float)
print("mse(f, g)=", mse(f, g))
print("mse(f, g+2)=", mse(f, g+2))
g[0,0]=0
print("mse(f, g)=", mse(f, g)) # --> 0.01

f = imread('data/apple.jpg')
print("mse(f, f)=", mse(f, f)) # --> 0 OK
print("mse(f, f+10)=", mse(f, f+10)) # --> 100 OK
print("mse(f, f+16)=", mse(f, f+16)) # --> 0 ???
```

課題 3：画像透かし

```
import numpy as np
from skimage.io import imread
import matplotlib.pyplot as plt
%matplotlib inline

# 学籍番号の下 1桁が<n>のとき、'watermark<n>.png'の画像を解読しなさい。例えば、"4622999"であれば"watermark9.png"を使用.
im = imread('exercise/watermark5.png', as_gray=True)

def mse(f, g):
    return np.mean((f - g) ** 2)

# 各水平線のMSE 値を格納するリストを初期化
mse_values_y = []
mse_values_x = []

# 各水平線のMSE を次のラインと比較して計算
for y in range(640 - 1):
    mse_value_y = mse(im[y, :], im[y + 1, :])
    mse_values_y.append(mse_value_y)
for x in range(640 - 1):
    mse_value_x = mse(im[x, :], im[x + 1, :])
    mse_values_x.append(mse_value_x)

# 最大のMSE を持つ x,y 座標を特定
```

```
y_line = np.argmax(mse_values_y)
x_line = np.argmax(mse_values_x)

# 特定された水平線に沿った画素値を抽出
horizontal_line_y = im[y_line, :]
horizontal_line_x = im[x_line, :]

#x,y 座標を基に二値化
def binarize(I):
    I[0:y_line, x_line:640] = np.where(I[0:y_line, x_line:640] >= 231+0,
       255, 0) # 231を基準に二値化する
    I[y_line:640, x_line:640] = np.where(I[y_line:640, x_line:640] >=
       252+1, 255, 0)
    I[0:y_line, 0:x_line] = np.where(I[0:y_line, 0:x_line] >= 1+3, 255, 0)
    I[y_line:640, 0:x_line] = np.where(I[y_line:640, 0:x_line] >= 11+2,
       255, 0)
    return I

im2 = binarize(im.copy())

# 特定された水平線を含む画像をプロット
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(im, cmap='gray')


plt.subplot(1, 2, 2)
plt.imshow(im2, cmap="gray")


plt.tight_layout()
plt.show()

print(f'Identified y-coordinate: {y_line}')
print(f'Identified x-coordinate: {x_line}')
```