**コンテンツ保護特論　量子化法による電子透かし　課題**

**1月10日　43M22403　古谷康平**

1. ソースコード

import matplotlib.pyplot as plt

import cv2

import random as rd

from scipy.fftpack import dct, idct

# 2次元離散コサイン変換 (DCT)

def dct\_2d(img):

return dct(dct(img, axis=0, norm='ortho'), axis=1, norm='ortho')

# 2次元逆離散コサイン変換 (IDCT)

def idct\_2d(img\_dct):

return idct(idct(img\_dct, axis=0 , norm='ortho'), axis=1 , norm='ortho')

#[1,0,1,1,1,0,...]のようになる

def generateW(bit\_len):

Wb = [1]

for \_ in range(bit\_len):

r = rd.random()

if r < 0.5:

Wb.append(0)

elif r >= 0.5:

Wb.append(1)

return Wb

#配列を文字列形式に変換する関数

def gen\_w(Wb, bit\_len):

s = 0

for x in range(bit\_len):

s += Wb[x] \* 2\*\*(bit\_len - x - 1)

return s

# 画像読み込み

img = cv2.imread('lena.png')

# 画像の色空間の変換

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

img\_yuv = cv2.cvtColor(img\_rgb, cv2.COLOR\_RGB2YUV)

# DCT

img\_dct = dct\_2d(img\_yuv[:,:,0])

# 量子化法による電子透かし

len\_bit = 64

w\_array = generateW(len\_bit)

w = gen\_w(w\_array, len\_bit)

#ランダムに生成した0～1の数を二進数に変換

ww = format(w, 'b')

count = 0

step = 100

for i in range(8):

for j in range(8):

im\_step = int(img\_dct[i][j]/step)

if(im\_step % 2 == 0)and(int(ww[count]) % 2 == 0):

img\_dct[i][j] = im\_step \* step

elif(im\_step % 2 == 1)and(int(ww[count]) % 2 == 0):

img\_dct[i][j] = (im\_step + 1) \* step

elif(im\_step % 2 == 0)and(int(ww[count]) % 2 == 1):

img\_dct[i][j] = (im\_step + 1) \* step

elif(im\_step % 2 == 1)and(int(ww[count]) % 2 == 1):

img\_dct[i][j] = im\_step \* step

count += 1

# 逆DCT

img\_inv = idct\_2d(img\_dct)

# 色空間をYUV表示系からRGB表示系に変換

img\_yuv[:,:,0] = img\_inv

img\_rgb\_inv = cv2.cvtColor(img\_yuv, cv2.COLOR\_YUV2RGB)

plt.figure(figsize=(10,10))

# 原画像を表示

plt.subplot(1,2,1)

plt.imshow(img\_rgb)

plt.title("original")

plt.axis("off")

# 復元した画像をグラフとして表示

plt.subplot(1,2,2)

plt.imshow(img\_rgb\_inv)

plt.title("inverse")

plt.axis("off")

# PSNRを計算

PSNR, \_ = cv2.quality.QualityPSNR\_compute(img, img\_yuv)

print(PSNR)

print('PSNR:', (PSNR[0]+PSNR[1]+PSNR[2])/3)

# SSIMを計算

SSIM, \_ = cv2.quality.QualitySSIM\_compute(img, img\_yuv)

print(SSIM)

print('SSIM:', (SSIM[0]+SSIM[1]+SSIM[2])/3)

1. 埋め込み後の画像(inverse)とそのPSNR値とSSIM値



PSNR値: 15.008436366787677

SSIM値: 0.586408078929917

1. 考察

量子化法による電子透かしの手法は、従来の方法よりひずみ値PSNR値とSSIM値が低いので、より元の画像の状態を保ったまま透かし情報を低中周波成分に埋め込むことが出来ていると考えられる。

1. プラットフォーム

Google colaboratory