**멀티미디어통신 실습 – image processing using Python**

201620350 김지영

1. Image threshold, gamma correction, image filtering과 image histogram을 위한 Python program을 실행하여 결과를 구하고 결과 영상의 특성분석 및 결과영상이 생성되는 원리를 분석하시오. (Python code를이용)

(1) Image threshold

Code

import numpy as np

from matplotlib import pyplot as plt

def threshold\_fn(inimg):

h,w,z=img.shape

outimg=np.zeros([h,w])

for i in range(h):

for j in range(w):

if (lumimg[i,j]>=50):

outimg[i,j] = 255

else:

outimg[i,j] = 0

return outimg

img = plt.imread('png\_img1.png')

img = np.uint8(255\*img)

plt.imshow(img)

plt.show()

#Image threshold

h,w,z = img.shape

lumimg = img[:,:,0] \* 0.2126 + img[:,:,1] \* 0.7152 + img[:,:,2] \* 0.0722

lumimg = np.uint8(lumimg)

imgthres = np.zeros([h,w])

for i in range(h):

for j in range(w):

if (lumimg[i,j]>=50):

imgthres[i,j] = 255

else:

imgthres[i,j] = 0

imgthrescolor = np.zeros(img.shape)

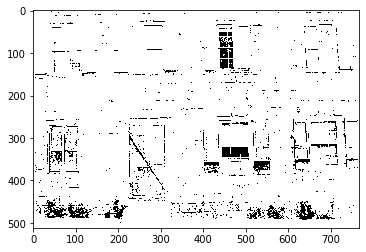
imgthrescolor[:,:,0] = imgthrescolor[:,:,1]= imgthrescolor[:,:,2] = imgthres[:,:]

plt.imshow(imgthrescolor)

plt.show()

결과 영상

(1) Threshold = 50인 경우

(2) Threshold = 128인 경우

해석

임계값이라고 부르는 threshold 함수는 영상을 분할할 수 있는 가장 간단한 방법입니다.

임계값이 50일때가 128일 때 보다 더 흰 부분이 많이 생겨, 원본 이미지에서 완전히 벗어난 모습을 확인할 수 있습니다. 임계값이 50일때는 128보다 0에 훨씬 가깝기 때문에 (임계값이 상대적으로 작기 때문에), 흰색으로 표현되는 픽셀수가 많아져서 이미지가 더 하얗게 보이며 원본이미지에서 확실하게 어두운 부분만 찾아낸 것입니다.

(2) gamma correction

Code

import numpy as np

from matplotlib import pyplot as plt

img = plt.imread('png\_img2.png')

img = np.uint8(255\*img)

plt.imshow(img)

plt.show()

#gamma correction

gcorimg = np.power(img/255.0,1/2.2)

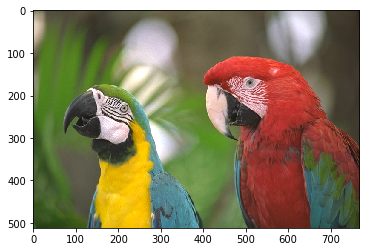
gcorimg = gcorimg\*255.0

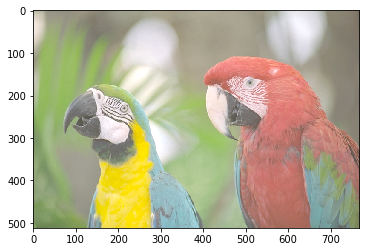
gcorimg = np.uint8(gcorimg)

plt.imshow(gcorimg)

plt.show()

결과 영상





해석

Gamma값을 2.2로 입력한 결과, 1보다 크기 때문에 결과 이미지가 입력 이미지보다 밝아졌습니다. 이러한 감마 보정법을 통해 이미지 밝기의 균형을 유지시킬 수 있는 것입니다.

(3) image filtering

Code

import numpy as np

from scipy import ndimage

from matplotlib import pyplot as plt

# A simple lowpass filter

kernel1 = np.array([[1/9, 1/9, 1/9],

[1/9, 1/9, 1/9],

[1/9, 1/9, 1/9]])

kernel2 = np.array([[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49],

[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49],

[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49],

[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49],

[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49],

[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49],

[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49]])

img = plt.imread('png\_img3.png')

img = np.uint8(255\*img)

plt.imshow(img)

plt.show()

h,w,z = img.shape

lumimg = img[:,:,0] \* 0.2126 + img[:,:,1] \* 0.7152 + img[:,:,2] \* 0.0722

lowpass = ndimage.convolve(lumimg, kernel1)

lowpassc= np.zeros(img.shape)

lowpassc[:,:,0] = lowpassc[:,:,1]= lowpassc[:,:,2] = lowpass[:,:]

lowpassc=np.uint8(lowpassc)

plt.imshow(lowpassc)

plt.show()

lowpass = ndimage.convolve(lumimg, kernel2)

lowpassc= np.zeros(img.shape)

lowpassc[:,:,0] = lowpassc[:,:,1]= lowpassc[:,:,2] = lowpass[:,:]

lowpassc=np.uint8(lowpassc)

plt.imshow(lowpassc)

plt.show()

lowpass1= np.zeros(img.shape)

lowpass1[:,:,0] = ndimage.convolve(img[:,:,0], kernel2)

lowpass1[:,:,1] = ndimage.convolve(img[:,:,1], kernel2)

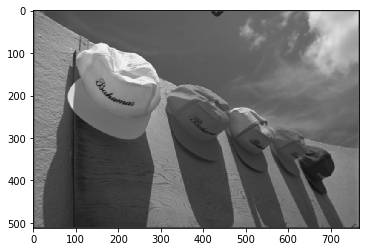
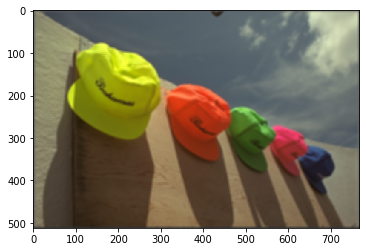
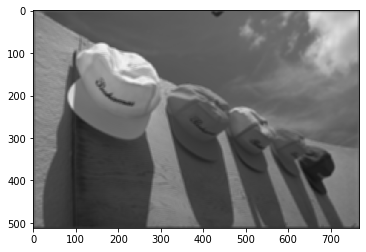
lowpass1[:,:,2] = ndimage.convolve(img[:,:,2], kernel2)

lowpass1=np.uint8(lowpass1)

plt.imshow(lowpass1)

plt.show()

결과 영상

해석

저주파 통과 필터링은 신호 성분 중에서 저주파 성분은 통과시키고 고주파 성분은 차단하는 필터입니다. 주로 잡음을 제거하거나, 흐릿한 영상을 얻을 때 주로 사용됩니다. 위의 이미지는 color 이미지에 3\*3 필터와 7\*7 필터를 통과시킨 모습입니다. 결과값을 비교해보면, 7\*7 크기의 마스크로 필터링한 이미지가 3\*3 일때보다 이미지 blurring이 더 심합니다 (더 흐릿해 보입니다). 이로써, 필터의 마스크 크기가 더 커질수록 영상이 더 흐릿해짐을 알 수 있습니다.

(4) image histogram

Code

import numpy as np

from matplotlib import pyplot as plt

img = plt.imread('png\_img4.png')

img = np.uint8(255\*img)

plt.imshow(img)

plt.show()

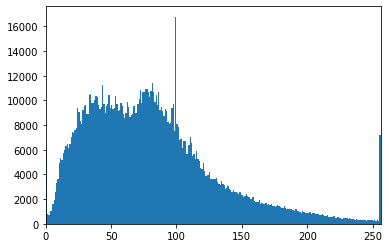
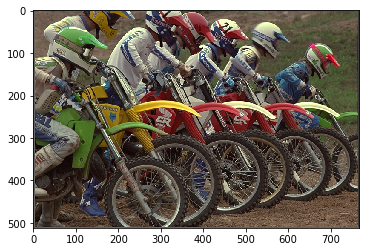
hist, bins = np.histogram(img.flatten(),256,[0,256])

plt.hist(img.flatten(),256,[0,256])

plt.xlim([0,256])

plt.show()

결과 영상



해석

입력이미지의 명암대비가 좋은 영상일수록, 히스토그램 분포가 넓게 퍼진 모습을 나타냅니다. 위의 이미지는 히스토그램 분포가 다소 어두운쪽에 위치해있습니다.

2. Gamma correction을 위한 program을 function을 이용하여 다시 작성하시오.

Code

import numpy as np

from matplotlib import pyplot as plt

def gamma\_fn(img):

gcorimg = np.power(img/255.0,1/2.2)

gcorimg = gcorimg\*255.0

gcorimg = np.uint8(gcorimg)

plt.imshow(gcorimg)

plt.show()

img = plt.imread('png\_img2.png')

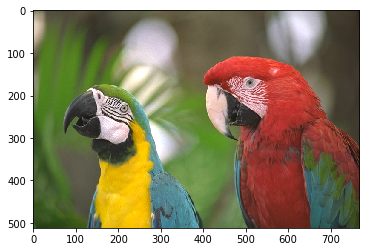
img = np.uint8(255\*img)

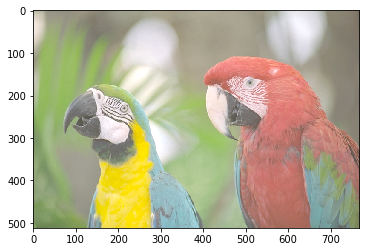
plt.imshow(img)

plt.show()

gamma\_fn(img)

결과 영상





3. Gamma correction과 threshold를 위한 function을 img\_fn.py file에 작성하고, import img\_fn을 이용하여 gamma correction과 threshold를 하는 program을 작성하시오.

Code (img\_fn.py)

import numpy as np

from matplotlib import pyplot as plt

def gamma\_fn(img):

gcorimg = np.power(img/255.0,1/2.2)

gcorimg = gcorimg\*255.0

gcorimg = np.uint8(gcorimg)

plt.imshow(gcorimg)

plt.show()

img = plt.imread('png\_img2.png')

img = np.uint8(255\*img)

plt.imshow(img)

plt.show()

gamma\_fn(img)

def threshold\_fn(inimg):

h,w,z=img.shape

outimg=np.zeros([h,w])

for i in range(h):

for j in range(w):

if (lumimg[i,j]>=50):

outimg[i,j] = 255

else:

outimg[i,j] = 0

return outimg

img = plt.imread('png\_img1.png')

img = np.uint8(255\*img)

plt.imshow(img)

plt.show()

#Image threshold

h,w,z = img.shape

lumimg = img[:,:,0] \* 0.2126 + img[:,:,1] \* 0.7152 + img[:,:,2] \* 0.0722

lumimg = np.uint8(lumimg)

imgthres = np.zeros([h,w])

for i in range(h):

for j in range(w):

if (lumimg[i,j]>=128):

imgthres[i,j] = 255

else:

imgthres[i,j] = 0

imgthrescolor = np.zeros(img.shape)

imgthrescolor[:,:,0] = imgthrescolor[:,:,1]= imgthrescolor[:,:,2] = imgthres[:,:]

plt.imshow(imgthrescolor)

plt.show()

imgthres\_fn = threshold\_fn(img)

Code (test.py)

import img\_fn

import numpy as np

from matplotlib import pyplot as plt

결과 영상

