**디지털 영상처리**

**미니 포토샵 만들기**

디지털 영상처리

김은이 교수님

20110315 김혜규

**[소스코드] – python**

import cv2

import numpy as np

from math import acos, pi, sqrt

from PIL import Image, ImageTk, ImageFilter, ImageDraw

import tkinter as tk

from tkinter import filedialog as fd

im = None

tk\_img = None

#파일 열기

def open() :

global im, tk\_img

fname = fd.askopenfilename()

im = Image.open(fname)

tk\_img = ImageTk.PhotoImage(im)

canvas.create\_image(250, 250, image=tk\_img)

window.update()

#프로그램 종료

def quit():

window.destroy()

#Power law

def create\_power\_law():

global im, tk\_img

def power\_law():

x1, x2, x3, x4 = int(e1.get()), int(e2.get()), int(e3.get()), int(e4.get())

out = np.array(255\*(im/255)\*\*gamma, dtype = 'uint8')

tk\_img = Image.Tk.PhotoImage(out)

canvas.create\_image(250,250,image=tk\_img)

window.update()

def new\_quit():

newWindow.destroy()

newWindow = tk.Toplevel(window)

l1 = tk.Label(newWindow, text = "gamma 값 : ")

e1 = tk.Entry(newWindow)

b1 = tk.Button(newWindow, text = "확인", command = lambda : [text(), new\_quit()])

l1.grid(row = 0, column = 0, columnspan = 2)

e1.grid(row = 0, column = 1, columnspan = 2)

b1.grid(row = 1, column = 0, columnspan = 2)

#negative

def negative():

global im, tk\_img

out = 1.0 - (im / 255)

tk\_img = Image.Tk.PhotoImage(out)

canvas.create\_image(250,250,image=tk\_img)

window.update()

#히스토그램 이퀄라이징

def equalize\_histogram ():

global im, tk\_img

equ = cv2.equalizeHist(im)

imgHist = cv2.calcHist(images = [im],

channels = [0],

mask = None,

histSize = [256],

ranges = [0,256])

equHist = cv2.calcHist(images = [equ],

channels = [0],

mask = None,

histSize = [256],

ranges = [0,256])

plt.plot(imgHist, color = 'b', label = 'img hist')

plt.plot(equHist, color = 'r', label = 'equ hist')

tk\_img = Image.Tk.PhotoImage(out)

canvas.create\_image(250,250,image=tk\_img)

window.update()

#가우시안 블러

def gaussian\_blur():

global im, tk\_img

kernel1D = cv2.getGaussianKernel(5,3)

krenel2D = np.outer(kernel1D, kernel1D.transpose())

im\_array = np.asarray(im)

blurred\_im\_array = cv2.filter2D(im\_array, -1, kernel2D)

blurred\_im = Image.fromarray(blurred\_im\_array)

tk\_img = Image.Tk.photoImage(blurred\_im)

canvas.create\_image(250,250, image = tk\_img)

window.update()

def average\_blur():

global im, tk\_img

out = cv2.blur(im, (5,5))

tk\_img = Image.Tk.PhotoImage(out)

canvas.create\_image(250,250,image=tk\_img)

window.update()

def median\_blur():

global im, tk\_img

filter\_size = (3,3)

stride = 1

img\_shape = np.shape(im)

after\_shape = tuple(np.int64((np.array(img\_shape)-np.array(filter\_size))/stride+1))

blurred\_im = np.zeros(after\_shape)

for h in range(0, after\_shape[0], stride):

for w in range(0, after\_shape[1], stride):

tmp = im[h:h+filter\_size[0],w:w+filter\_size[1]]

tmp = np.sort(tmp.ravel())

result[h,w] = tmp[int(filter\_size[0]\*filter\_size[1]/2)]

return result

tk\_img = imgae.Tk.photoImage(blurred\_im)

canvas.create\_image(250,250, image = tk\_img)

window.update()

#엣지 검출 필터

def canny\_edge():

global im, tk\_img

median\_intensity = np.median(im)

lower\_threshold = int(max(0, (1.0 - 0.33) \* median\_intensity))

upper\_threshold = int(min(255, (1.0 + 0.33) \* median\_intensity))

out = cv2.Canny(img, lower\_threshold, upper\_threshold)

tk\_img = ImageTk.PhotoImage(out)

canvas.create\_image(250,250,image=tk\_img)

window.update()

def Prewitt():

global im, tk\_img

prewitt\_x = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])

prewitt\_y = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])

prewitt\_x = cv2.convertScaleAbs(cv2.filter2D(im, -1, prewitt\_x))

prewitt\_y = cv2.convertScaleAbs(cv2.filter2D(im, -1, prewitt\_y))

prewitt = cv2.addWeighted(prewitt\_x, 1, prewitt\_y, 1, 0)

tk\_img = ImageTk.PhotoImage(prewitt)

canvas.create\_image(250,250,image=tk\_img)

window.update()

def Sobel() :

global im, tk\_img

sobel\_x = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

sobel\_y = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])

sobel\_x = cv2.convertScaleAbs(cv2.filter2D(im, -1, sobel\_x))

sobel\_y = cv2.convertScaleAbs(cv2.filter2D(im, -1, sobel\_y))

sobel = cv2.addWeighted(sobel\_x, 1, sobel\_y, 1, 0)

tk\_img = ImageTk.PhotoImage(sobel)

canvas.create\_image(250,250,image=tk\_img)

window.update()

def Roberts():

global im, tk\_img

roberts\_x = np.array([[-1, 0, 0], [0, 1, 0], [0, 0, 0]])

roberts\_y = np.array([[0, 0, -1], [0, 1, 0], [0, 0, 0]])

roberts\_x = cv2.convertScaleAbs(cv2.filter2D(img, -1, roberts\_x))

roberts\_y = cv2.convertScaleAbs(cv2.filter2D(img, -1, roberts\_y))

#robertsx = cv2.filter2D(img, cv2.CV\_64F, roberts\_x)

#robertsy = cv2.filter2D(img, cv2.CV\_64F, roberts\_y)

roberts = cv2.addWeighted(roberts\_x, 1, roberts\_y, 1, 0)

tk\_img = ImageTk.PhotoImage(roberts)

canvas.create\_image(250,250,image=tk\_img)

window.update()

#filter2D : mask 적용 함수. 그레이 스케일에 각각의 공식의 가로와 세로 공식에 맞게 적용

#convertScaleAbs : 각각의 값을 절대값화 시키고 정수화. 이 과정을 거쳐야 출력했을 때 정상적인 이미지 나옴

def LoG():

mask1 = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])

gaussian = cv2.GaussianBlur(img, (7,7), 1.2)

LoG = cv2.filter2D(gaussian2, -1, mask1)

tk\_img = ImageTk.PhotoImage(LoG)

canvas.create\_image(250,250,image=tk\_img)

window.update()

#main

window = tk.Tk( )

canvas = tk.Canvas(window, width = 500, height = 500)

canvas.pack()

menubar = tk.Menu(window)

filemenu = tk.Menu(window)

ipmenu = tk.Menu(window)

resmenu = tk.Menu(window)

filemenu.add\_command(label = "이미지 가져오기",command = open)

filemenu.add\_command(label = "종료",command = quit)

ipmenu.add\_command(label = "히스토그램 이퀄라이저", command = equalize\_histogram)

ipmenu.add\_command(label = "흑백전환", command = negative)

ipmenu.add\_command(label = "Power Law", command = create\_power\_law)

ipmenu.add\_command(label = "Gaussian Blur", command = gaussian\_blur)

ipmenu.add\_command(label = "Median Blur", command = median\_blur)

ipmenu.add\_command(label = "Gaussian Blur", command = gaussian\_blur)

ipmenu.add\_command(label = "엣지검출 - Canny", command = canny\_edge)

ipmenu.add\_command(label = "엣지검출 - Prewitt", command = Prewitt)

ipmenu.add\_command(label = "엣지검출 - Sobel", command = Sobel)

ipmenu.add\_command(label = "엣지검출 - Roberts", command = Roberts)

ipmenu.add\_command(label = "2차 미분 - LoG 필터", command = LoG)

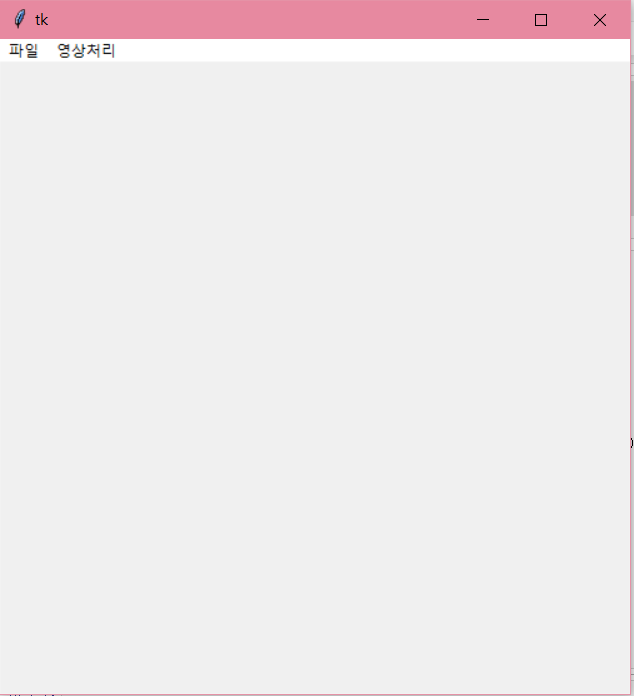
menubar.add\_cascade(label = "파일", menu = filemenu)

menubar.add\_cascade(label = "영상처리", menu = ipmenu)

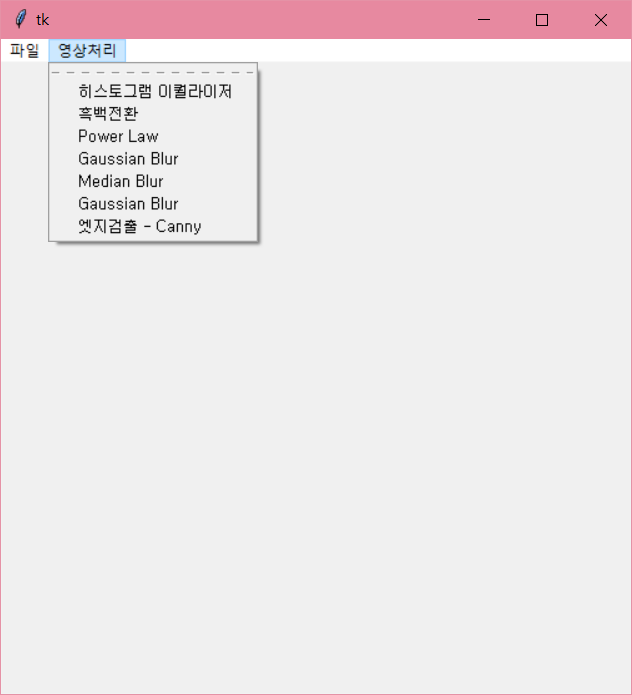
window.config(menu = menubar)

window.mainloop()

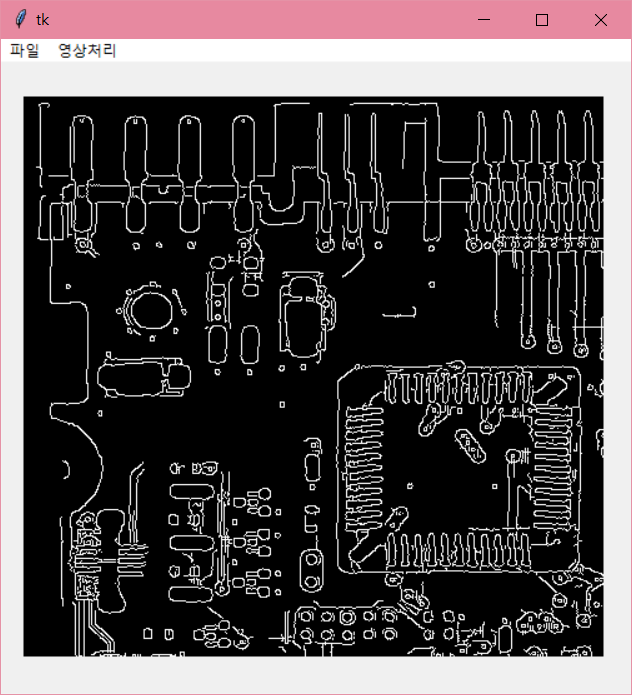
[ 실행 ]



[ 실행 직후 모습]

[메뉴 탭 작동]

[ 영상 가져오기 ]

[기능설명]

Tkinter 라이브러리를 통해 Tcl/Tk를 파이썬에 이용할 수 있도록 한 Lightweight GUI 모듈이 있다는 것을 알게되어 이를 이용하여 포토샵 프로그램 같은 메뉴 탭을 만들었습니다.

메뉴 탭에는 이미지를 불러오고 프로그램을 종료하는 ‘파일’ 탭과 영상처리 기능들을 묶어놓은 ‘영상처리’ 탭을 만들었습니다.

‘파일’ 탭에는 ‘파일 불러오기’ 기능으로 기존에 저장돼있는 이미지 파일을 가져올 수 있게끔 만들었고, ‘종료’ 버튼을 통해 프로그램을 종료할 수 있도록 했습니다.

‘영상처리’ 탭에는 여태껏 과제로 나왔던 ‘Histogram Equalizer’, ‘Negative’, ‘Power Law’, ’ Smoothing(Gaussian, Median, Averaging)’, ‘Edge detection(1차미분 – Roberts, Prewitt, Sobel, / 2차 미분 – LoG / Canny)’ 을 구현하였습니다.

다만, LoG필터에서 마스크 사이즈는 7\*7 사이즈로 고정해놨습니다.