|  |  |  |
| --- | --- | --- |
|  | **Lembar Kerja Mahasiswa**  **Mata Kuliah Pengolahan Citra Digital Praktik (203311-20)**  **Program Studi Informatika**  **Fakultas Sains & Teknologi – Universitas Teknologi Yogyakarta** | |
|  | **Identitas Mahasiswa** | |
| **Nama** | **Arieska Restu Harpian Dwika** |
| **NPM** | **5200411488** |
| **Kelompok Prak** | **Kel. I** |
|  | | |
| **Soal 1.** | | |
| Berdasarkan demo di kelas, tambahkanlah fitur pada aplikasi yang telah anda buat pada Lembar kerja minggu ke-8 antara lain:   1. 3 button masing-masing bertuliskan **Canny**, **Sobel**, dan **Prewitt**  berfungsi untuk melakukan deteksi tepi dengan metode sesuai nama button terhadap citra asli. 2. 3 *image container* masing-masing untuk menampung citra hasil deteksi tepi setelah button deteksi tepi di-klik.   Pastikan pada tugas kali ini Anda menggunakan program GUI yang sudah Anda buat untuk pertemuan ke-8. Pastikan juga aplikasi mampu menampilkan citra asli dan citra hasil deteksi tepi dalam satu jendela. Buatlah layout GUI yang menarik dan tetap mudah digunakan. | | |
| **Hasil Script** | | |
| **//tuliskan script python Anda di sini**  # 5200411488 - Arieska Restu Harpian Dwika  import cv2  import numpy as np  import os  from tkinter import \*  from tkinter import font  from tkinter import filedialog  from ttkbootstrap import Style  from tkinter import ttk  import tkinter as tk  from PIL import Image, ImageTk  def setOriginal(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblImgOriginal.configure(*image*=imgTk)      lblImgOriginal.image = imgTk      lblImgOriginal.pack()  def setResultFilter(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblResultFilter.configure(*image*=imgTk)      lblResultFilter.image = imgTk      lblResultFilter.pack()  def setResultCanny(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblResultCanny.configure(*image*=imgTk)      lblResultCanny.image = imgTk      lblResultCanny.pack()  def setResultSobel(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblResultSobel.configure(*image*=imgTk)      lblResultSobel.image = imgTk      lblResultSobel.pack()  def setResultPrewitt(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblResultPrewitt.configure(*image*=imgTk)      lblResultPrewitt.image = imgTk      lblResultPrewitt.pack()  def opencv2Pill(*img*):  *img* = cv2.cvtColor(*img*, cv2.COLOR\_BGR2RGB)      imgPill = Image.fromarray(*img*)      return imgPill    def resizeImg(*img*, *width*, *height*):  *img* = cv2.resize(*img*, (*width*, *height*), *interpolation*=cv2.INTER\_CUBIC)      return *img*  def browseImage():      global fln      fln = filedialog.askopenfilename(*initialdir*=os.getcwd(), *title*="Select Image File",  *filetypes*=(                                          ("All Files", "\*.\*",),                                          ("PNG File", "\*.png"),                                          ("JPG File", "\*.jpg"))                                      )        img = opencv2Pill(resizeImg(cv2.imread(fln), 256, 256))      setOriginal(img)  def filtering():      global fln        img = cv2.imread(fln)      kernel = np.array(              [              [0, -1, 0],              [-1,5, -1],              [0, -1, 0],              ],  *dtype*='float')      imgFilter = cv2.filter2D(img, -1, kernel)        setResultFilter(opencv2Pill(resizeImg(imgFilter, 256, 256)))    def canny():      global fln        img = cv2.Canny(cv2.imread(fln), 100, 200)      setResultCanny(opencv2Pill(resizeImg(img, 256, 256)))    def sobel():      global fln      img = cv2.imread(fln)      gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)        imgGaussian = cv2.GaussianBlur(gray,(3,3),0)      imgSobelx = cv2.Sobel(imgGaussian,cv2.CV\_8U,1,0,*ksize*=5)      imgSobely = cv2.Sobel(imgGaussian,cv2.CV\_8U,0,1,*ksize*=5)      imgSobel = imgSobelx + imgSobely      setResultSobel(opencv2Pill(resizeImg(imgSobel, 256, 256)))  def prewitt():      global fln      img = cv2.imread(fln)      gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)        imgGaussian = cv2.GaussianBlur(gray,(3,3),0)      kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])      kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])      imgPrewittX = cv2.filter2D(imgGaussian, -1, kernelx)      imgPrewittY = cv2.filter2D(imgGaussian, -1, kernely)      imgPrewitt = imgPrewittX + imgPrewittY        setResultPrewitt(opencv2Pill(resizeImg(imgPrewitt, 256, 256)))    if \_\_name\_\_ == '\_\_main\_\_':      style = Style()      window = style.master      # Frame      frm = ttk.Frame(window, *style*='primary.TFrame')      # frm.pack(side='top')      frm.pack\_propagate(0)      frm.pack(*fill*=tk.BOTH, *expand*=1)      frmTop = ttk.Frame(frm, *style*='secondary.TFrame', *width*=900, *height*=550)      frmTop.grid(*row*=0, *column*=0, *padx*=20, *pady*=20)      frmImgOriginal = ttk.Frame(frmTop, *style*='info.TFrame', *width*=256, *height*=256)      frmImgOriginal.pack\_propagate(0)      frmImgOriginal.pack(*side*="left", *padx*=20, *pady*=20)      frmBtnTop = ttk.Frame(frmTop, *style*='secondary.TFrame', *width*=100, *height*=200)      frmBtnTop.pack(*side*="left", *padx*=20, *pady*=20)      frmImgFilter = ttk.Frame(frmTop, *style*='info.TFrame', *width*=256, *height*=256)      frmImgFilter.pack\_propagate(0)      frmImgFilter.pack(*side*="left", *padx*=20, *pady*=20)      frmBottom = ttk.Frame(frm, *style*='secondary.TFrame', *width*=900, *height*=550)      frmBottom.grid(*row*=1, *column*=0, *padx*=40, *pady*=(10,20))      frmImgCanny = ttk.Frame(frmBottom, *style*='info.TFrame', *width*=256, *height*=256)      frmImgCanny.grid(*row*=0, *column*=0, *padx*=20, *pady*=(20,2))      frmImgCanny.grid\_propagate(0)      frmImgSobel = ttk.Frame(frmBottom, *style*='info.TFrame', *width*=256, *height*=256)      frmImgSobel.grid(*row*=0, *column*=1, *padx*=20, *pady*=(20,2))      frmImgSobel.grid\_propagate(0)      frmImgPrewitt = ttk.Frame(frmBottom, *style*='info.TFrame', *width*=256, *height*=256)      frmImgPrewitt.grid(*row*=0, *column*=2, *padx*=20, *pady*=(20,2))      frmImgPrewitt.grid\_propagate(0)      frmBtnBottom = ttk.Frame(frmBottom, *style*='secondary.TFrame', *width*=848, *height*=43)      frmBtnBottom.grid(*row*=1, *column*=0, *columnspan*=3, *padx*=20, *pady*=(3,20))      frmBtnBottom.grid\_propagate(0)      # Button      btnBrowse = ttk.Button(frmBtnTop, *text*='Browse Image', *style*='info.TButton', *cursor*="hand2", *width*=12, *command*=browseImage)      btnBrowse.pack(*side*='top', *pady*=10)      btnFilter = ttk.Button(frmBtnTop, *text*='Filter', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=filtering)      btnFilter.pack(*side*='top', *pady*=10)      btnExit = ttk.Button(frmBtnTop, *text*='Exit', *style*='danger.TButton', *cursor*="hand2", *width*=12, *command*=lambda: exit())      btnExit.pack(*side*='top', *pady*=10)      btnCanny = ttk.Button(frmBtnBottom, *text*='Canny', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=canny)      btnCanny.grid(*row*=0, *column*=0, *padx*=80, *pady*=(10,0))      btnSobel = ttk.Button(frmBtnBottom, *text*='Sobel', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=sobel)      btnSobel.grid(*row*=0, *column*=1, *padx*=96, *pady*=(10,0))      btnPrewitt = ttk.Button(frmBtnBottom, *text*='Prewitt', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=prewitt)      btnPrewitt.grid(*row*=0, *column*=2, *padx*=96, *pady*=(10,0))      # Label      lblImgOriginal = ttk.Label(frmImgOriginal)      lblResultFilter = ttk.Label(frmImgFilter)      lblResultCanny = ttk.Label(frmImgCanny)      lblResultSobel = ttk.Label(frmImgSobel)      lblResultPrewitt = ttk.Label(frmImgPrewitt)          window.title("Edge Detection - 5200411488")      # window.geometry("1280x720")      window.resizable(0, 0)      window.mainloop() | | |
| **Hasil Running Aplikasi** | | |
| **//paste-kan tampilan aplikasi Anda di sini** | | |