|  |  |  |
| --- | --- | --- |
|  | **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-9 antara lain:   1. 1 button dengan nama “**Erosi**” yang akan melakukan erosi pada citra hasil deteksi tepi Canny yang sebelumnya sudah Anda buat 2. 1 text box dengan label “**St. El. Size**” yang akan menerima input berupa bilangan bulat dan nantinya akan digunakan untuk menentukan ukuran structuring element yang digunakan pada operasi morfologi 3. 1 button dengan nama “**Closing**” yang akan melakukan operasi closing pada citra hasil deteksi tepi Canny yang sebelumnya sudah Anda buat   Pastikan pada tugas kali ini Anda menggunakan program GUI yang sudah Anda buat untuk pertemuan ke-9. Pastikan juga aplikasi mampu menampilkan citra asli dan citra hasil deteksi tepi dan citra hasil morfologi **dalam satu jendela** **secara berdampingan**. 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 setResultErode(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblResultErode.configure(*image*=imgTk)      lblResultErode.image = imgTk      lblResultErode.pack()  def setResultClosing(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblResultClosing.configure(*image*=imgTk)      lblResultClosing.image = imgTk      lblResultClosing.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 canny(*img*):      imgCanny = cv2.Canny(*img*, 100, 200)      return imgCanny  def erosi(*img*, *kernel*):      imgErode = cv2.erode(*img*, *kernel*, *iterations*= 1)      return imgErode  def dilasi(*img*, *kernel*):      imgDilate = cv2.dilate(*img*, *kernel*, *iterations*= 1)      return imgDilate  def closing(*img*):      se = cv2.getStructuringElement(cv2.MORPH\_RECT, (3,3))      imgDilate = dilasi(*img*, se)      imgErode= erosi(imgDilate, se)      return imgErode  def erode(*img*):  *img* = canny(*img*)      m, n = *img*.shape        # k = 5 # 3, 5, 7, 9      k = int(txtStElSize.get())      kernel = np.ones((k,k), *dtype*=np.uint8)      constant = (k-1) // 2      imgErode = np.zeros((m,n), *dtype*=np.uint8)      for i in range(constant, m-constant): # (2, m-2)          for j in range(constant, n-constant): #(2, n-2)              temp = *img*[i-constant:i+constant+1, j-constant:j+constant+1]              product = temp \* kernel              imgErode[i,j] = np.min(product)        txtStElSize.delete(0, END)      return imgErode  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), 128, 128))      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, 128, 128)))    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, 128, 128)))  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, 128, 128)))  def btnCannyClicked():      global fln      img = cv2.imread(fln)      setResultCanny(opencv2Pill(resizeImg(canny(img), 128, 128)))  def btnErodeClicked():      global fln      img = canny(cv2.imread(fln, 0))      setResultErode(opencv2Pill(resizeImg(erode(img), 128, 128)))  def btnClosingClicked():      global fln      img = canny(cv2.imread(fln, 0))      setResultClosing(opencv2Pill(resizeImg(closing(img), 128, 128)))    if \_\_name\_\_ == '\_\_main\_\_':      style = Style()      window = style.master      # Frame      frm = ttk.Frame(window, *style*='primary.TFrame')      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*=128, *height*=128)      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*=128, *height*=128)      frmImgFilter.pack\_propagate(0)      frmImgFilter.pack(*side*="left", *padx*=20, *pady*=20)      frmMid = ttk.Frame(frm, *style*='secondary.TFrame', *width*=500, *height*=550)      frmMid.grid(*row*=1, *column*=0, *padx*=10, *pady*=(10,20))      frmImgCanny = ttk.Frame(frmMid, *style*='info.TFrame', *width*=128, *height*=128)      frmImgCanny.grid(*row*=0, *column*=0, *padx*=10, *pady*=(20,2))      frmImgCanny.grid\_propagate(0)      frmImgSobel = ttk.Frame(frmMid, *style*='info.TFrame', *width*=128, *height*=128)      frmImgSobel.grid(*row*=0, *column*=1, *padx*=10, *pady*=(20,2))      frmImgSobel.grid\_propagate(0)      frmImgPrewitt = ttk.Frame(frmMid, *style*='info.TFrame', *width*=128, *height*=128)      frmImgPrewitt.grid(*row*=0, *column*=2, *padx*=10, *pady*=(20,2))      frmImgPrewitt.grid\_propagate(0)      frmBtnMid = ttk.Frame(frmMid, *style*='secondary.TFrame', *width*=848, *height*=43)      frmBtnMid.grid(*row*=1, *column*=0, *columnspan*=3, *padx*=10, *pady*=(3,20))      frmBtnMid.grid\_propagate(0)      frmBottom = ttk.Frame(frm, *style*='secondary.TFrame', *width*=800, *height*=550)      frmBottom.grid(*row*=2, *column*=0, *padx*=10, *pady*=(10,20))      frmImgErode = ttk.Frame(frmBottom, *style*='info.TFrame', *width*=128, *height*=128)      frmImgErode.grid(*row*=0, *column*=0, *padx*=(175,0), *pady*=(20,2))      frmImgErode.grid\_propagate(0)      frmImgClosing = ttk.Frame(frmBottom, *style*='info.TFrame', *width*=128, *height*=128)      frmImgClosing.grid(*row*=0, *column*=1, *padx*=50, *pady*=(20,2))      frmImgClosing.grid\_propagate(0)      frmBtnBottom = ttk.Frame(frmBottom, *style*='secondary.TFrame', *width*=848, *height*=43)      frmBtnBottom.grid(*row*=1, *column*=0, *columnspan*=3, *padx*=10, *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(frmBtnMid, *text*='Canny', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=btnCannyClicked)      btnCanny.grid(*row*=0, *column*=0, *padx*=80, *pady*=(10,0))      btnSobel = ttk.Button(frmBtnMid, *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(frmBtnMid, *text*='Prewitt', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=prewitt)      btnPrewitt.grid(*row*=0, *column*=2, *padx*=96, *pady*=(10,0))      lblStElSize = ttk.Label(frmBtnBottom, *text*=f'St. El. Size : ', *style*='secondary.Inverse.TLabel')      lblStElSize.grid(*row*=0, *column*=0, *padx*=(30,0), *pady*=(10,0))      txtStElSize = ttk.Entry(frmBtnBottom, *font*="Normal 10",*style*='info.TEntry', *width*=7)      txtStElSize.grid(*row*=0, *column*=1, *padx*=(0,4), *pady*=(10,0))      btnErode = ttk.Button(frmBtnBottom, *text*='Erode', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=btnErodeClicked)      btnErode.grid(*row*=0, *column*=2, *padx*=(50,0), *pady*=(10,0))      btnClosing = ttk.Button(frmBtnBottom, *text*='Closing', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=btnClosingClicked)      btnClosing.grid(*row*=0, *column*=3, *padx*=(184), *pady*=(10,0))      # Label      lblImgOriginal = ttk.Label(frmImgOriginal)      # lblImgOriginal.grid(row=0, column=0)      lblResultFilter = ttk.Label(frmImgFilter)      # lblResultFilter.grid(row=0, column=0)      lblResultCanny = ttk.Label(frmImgCanny)      # lblResultCanny.grid(row=0, column=0)      lblResultSobel = ttk.Label(frmImgSobel)      # lblResultSobel.grid(row=0, column=0)      lblResultPrewitt = ttk.Label(frmImgPrewitt)      # lblResultPrewitt.grid(row=0, column=0)      lblResultErode = ttk.Label(frmImgErode)      # lblResultErode.grid(row=0, column=0)      lblResultClosing = ttk.Label(frmImgClosing)      # lblResultClosing.grid(row=0, column=0)      window.title("Erode & Closing - 5200411488")      # window.geometry("1280x720")      window.resizable(0, 0)      window.mainloop() | | |
| **Hasil Running Aplikasi** | | |
| **//paste-kan tampilan aplikasi Anda di sini** | | |