**Laporan**

**Tugas Besar - Pengolahan Citra Digital**

|  |  |  |
| --- | --- | --- |
| **Nama** | **:** | Arieska Restu Harpian Dwika |
| **NPM** | **:** | 5200411488 |

**Penjelasan Algoritma Ekstraksi Area Region of Interest (ROI)**

|  |
| --- |
| Sebelum mengambil area ROI, citra dibaca terlebih dahulu dengan menggunakan fungsi **cv2.imread()** dan memasukkan lokasi file citra pada parameter fungsi tersebut. Kemudian mencari titik tengah dari citra yang telah dibaca dengan menggunakan rumus sebagai berikut.  Setelah mendapatkan titik tengah, program akan mencari titik tengah dari area ROI dengan rumus sebagai berikut.  Kemudian dari titik tengah ROI tersebut, akan diambil area sebesar 50 piksel ke kanan, kiri, atas, dan bawah seperti ilustrasi berikut.      Setelah itu, program akan menghilangkan warna selain warna mobil. Deteksi warna dilakukan dengan cara mengubah warna gambar yang sebelumnya BGR menjadi HSV. Kemudian user akan memasukkan input berupa nilai dari hue min, sat min, value min, hue max, sat max, dan value max. Lalu gambar yang telah diubah menjadi HSV difilter warnanya sesuai dengan nilai yang diinputkan oleh user dengan menggunakan fungsi **cv2.inRange()**.  Selanjutnya gambar hasil deteksi warna ditampilkan dengan fungsi **cv2.bitwise\_and()**. Fungsi ini berfungsi untuk menampilkan area citra asli dengan area citra mask yang beririsan. Setelah mendapatkan area ROI yang hanya berisi warna mobil, area ROI akan ditulis dan disimpan sebagai citra yang baru. |

**Kode Program**

|  |  |
| --- | --- |
| Kode program untuk mendapatkan ROI adalah sebagai berikut:   |  | | --- | | from typing import IO  import cv2 as cv  import numpy as np  import os  from tkinter import \*  from tkinter import messagebox  from tkinter import filedialog  from tkinter import ttk  from ttkbootstrap import Style          # ttkbootstrap version 0.5.1  import tkinter as tk  from PIL import Image, ImageTk    def setOriginal(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblImgOri.configure(*image*=imgTk)      lblImgOri.image = imgTk      lblImgOri.pack()  def setRoi(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblImgRoi.configure(*image*=imgTk)      lblImgRoi.image = imgTk      lblImgRoi.pack()  def setResult(*img*):      imgTk = ImageTk.PhotoImage(*img*)      lblImgRes.configure(*image*=imgTk)      lblImgRes.image = imgTk      lblImgRes.pack()    def opencv2Pill(*img*):  *img* = cv.cvtColor(*img*, cv.COLOR\_BGR2RGB)      imgPill = Image.fromarray(*img*)      return imgPill    def resizeImg(*img*, *width*, *height*):  *img* = cv.resize(*img*, (*width*, *height*), *interpolation*=cv.INTER\_CUBIC)      return *img*  def getXMid(*img*):      return int(*img*.shape[1] / 2)  def getYMid(*img*):      return int(*img*.shape[0] / 2)  def drawCircleMid(*img*):      xMid = getXMid(*img*)      yMid = getYMid(*img*)      cv.circle(*img*, (xMid,yMid), 5, (0,0,255), cv.FILLED)    def getRectanglePos(*img*, *xMid*, *yMid*):      xRecMid = int((34/100) \* *img*.shape[1])      # yRecMid = int((55/100) \* img.shape[0])      yRecMid = *yMid*      xRec =  xRecMid - 25      yRec = yRecMid - 25      wRec, hRec = 50, 50      return xRec, yRec, wRec, hRec  def drawRectangle(*img*, *xMid*, *yMid*):      xRec, yRec, wRec, hRec = getRectanglePos(*img*, *xMid*, *yMid*)      cv.circle(*img*, (*xMid*,yRec), 5, (0,255,255), cv.FILLED)      cv.rectangle(*img*, (xRec,yRec), (xRec+wRec, yRec+hRec), (255,0,0), 2)  def getColorObject():      hueMin = int(sldHueMin.get())      satMin = int(sldSatMin.get())      valueMin = int(sldValueMin.get())      hueMax = int(sldHueMax.get())      satMax = int(sldSatMax.get())      valueMax = int(sldValueMax.get())      return hueMin, satMin, valueMin, hueMax, satMax, valueMax  def getFileName(*path*):      return *path*.split('/')[-1]  def writeData(*fileName*, *data*):      try:          fileFull = open("Code/Data/Name & Value.txt",'a')          fileValue = open("Code/Data/Value.txt",'a')          fileFull.write(f'{*fileName*}:{*data*}\n')          fileValue.write(f'{*data*}\n')          print(f'{*fileName*} \t: {*data*}')          print(f'{*data*}\n')          messagebox.showinfo('Notification', 'Write data successfully!')      except *IOError*:          messagebox.showerror('Error', 'Write data failed!')  def getResult(*img*):      hueMin, satMin, valueMin, hueMax, satMax, valueMax = getColorObject()      imgHSV = cv.cvtColor(*img*, cv.COLOR\_BGR2HSV)      lower = np.array([hueMin, satMin, valueMin])      upper = np.array([hueMax, satMax, valueMax])      mask = cv.inRange(imgHSV, lower, upper)        imgResult = cv.bitwise\_and(*img*, *img*, *mask* = mask)      return imgResult  def getRoi(*img*):      xMid = getXMid(*img*)      yMid = getYMid(*img*)      xRec, yRec, wRec, hRec = getRectanglePos(*img*, xMid, yMid)      imgRoi = *img*[yRec:yRec+hRec, xRec:xRec+wRec]      return imgRoi    def btnBrowseClicked():      global fln      fln = filedialog.askopenfilename(*initialdir*=os.getcwd(), *title*="Select Image File",  *filetypes*=(                                          ("All Files", "\*.\*",),                                          ("PNG File", "\*.png"),                                          ("JPG File", "\*.jpg"),                                          ("JFIF File", "\*.jfif"))                                      )        print(f'fln : {fln}\n')        img = cv.imread(fln)        xMid = getXMid(img)      yMid = getYMid(img)      # img = drawRectangle(img, xMid, yMid)      imgShow = opencv2Pill(resizeImg(img, 354, 472))        setOriginal(imgShow)  def sldMove(*e*):      global fln        hueMin, satMin, valueMin, hueMax, satMax, valueMax = getColorObject()        lblHueMin.configure(*text*=f'HUE Min : {hueMin}')      lblSatMin.configure(*text*=f'SAT Min : {satMin}')      lblValueMin.configure(*text*=f'VALUE Min : {valueMin}')      lblHueMax.configure(*text*=f'HUE Max : {hueMax}')      lblSatMax.configure(*text*=f'SAT Max : {satMax}')      lblValueMax.configure(*text*=f'VALUE Max : {valueMax}')        img = cv.imread(fln)      imgResult = getResult(img)      imgShow = opencv2Pill(resizeImg(imgResult, 354, 472))        setResult(imgShow)    def btnRoiClicked():      global fln        imgResult = getResult(cv.imread(fln))      imgRoi = getRoi(imgResult)      imgShow = opencv2Pill(resizeImg(imgRoi, 100, 100))        setRoi(imgShow)  def btnSaveClicked():      global fln        imgResult = getResult(cv.imread(fln))      imgRoi = getRoi(imgResult)      imgSave = opencv2Pill(resizeImg(imgRoi, 100, 100))      hueMin, satMin, valueMin, hueMax, satMax, valueMax = getColorObject()      data = [hueMin, satMin, valueMin, hueMax, satMax, valueMax]      fileName = getFileName(fln)        extension = [("JPG File", "\*.jpg")]      file = filedialog.asksaveasfile(*filetypes* = extension, *defaultextension* = extension)      if file:          imgSave.save(file)          messagebox.showinfo('Notification', 'Image save successfully!')          writeData(fileName, data)      else:          messagebox.showerror('Error', 'Image save failed!')        if \_\_name\_\_ == '\_\_main\_\_':        style = Style()      window = style.master      fln = None      frm = ttk.Frame(window, *style*='primary.TFrame')      # frm.pack(side='top')      frm.pack\_propagate(0)      frm.pack(*fill*=tk.BOTH, *expand*=1)      # Size window : 852 x 480      # Frame      frmImg = ttk.Frame(frm, *style*='secondary.TFrame', *width*=1000, *height*=472)      frmImg.grid(*row*=0, *column*=0, *columnspan*=3, *padx*=50, *pady*=20)      frmImgOri = ttk.Frame(frmImg, *style*='info.TFrame', *width*=354, *height*=472)      frmImgOri.grid(*row*=1, *column*=0, *padx*=30, *pady*=(10,20))      frmImgRoi = ttk.Frame(frmImg, *style*='info.TFrame', *width*=100, *height*=100)      frmImgRoi.grid(*row*=1, *column*=1, *padx*=30, *pady*=(10,20))      frmImgRes = ttk.Frame(frmImg, *style*='info.TFrame', *width*=354, *height*=472)      frmImgRes.grid(*row*=1, *column*=2, *padx*=30, *pady*=(10,20))      frmSlider = ttk.Frame(frm, *style*='secondary.TFrame', *width*=1000, *height*=150)      frmSlider.grid(*row*=1, *column*=0, *columnspan*=2, *padx*=50, *pady*=20)      frmSliderMin = ttk.Frame(frmSlider, *style*='info.TFrame', *width*=500, *height*=150)      frmSliderMin.grid(*row*=0, *column*=0, *padx*=20, *pady*=20)      frmBtn = ttk.Frame(frmSlider, *style*='secondary.TFrame', *width*=40, *height*=150)      frmBtn.grid(*row*=0, *column*=1, *padx*=10, *pady*=20)      frmSliderMax = ttk.Frame(frmSlider, *style*='info.TFrame', *width*=500, *height*=150)      frmSliderMax.grid(*row*=0, *column*=2, *padx*=20, *pady*=20)      # Label Description      lblDescOri = ttk.Label(frmImg, *text*='Original', *font*='20', *style*='secondary.Inverse.TLabel')      lblDescOri.grid(*row*=0, *column*=0, *padx*=20, *pady*=(10,0))      lblDescRoi = ttk.Label(frmImg, *text*='ROI', *font*='20', *style*='secondary.Inverse.TLabel')      lblDescRoi.grid(*row*=0, *column*=1, *padx*=20, *pady*=(10,0))      lblDescRes = ttk.Label(frmImg, *text*='Color Detection', *font*='20', *style*='secondary.Inverse.TLabel')      lblDescRes.grid(*row*=0, *column*=2, *padx*=20, *pady*=(10,0))      # Label Image      # Size Image : 354 x 472      lblImgOri = ttk.Label(frmImgOri)      # lblImgRes.pack()      lblImgRoi = ttk.Label(frmImgRoi)      # lblImgRes.pack()      lblImgRes = ttk.Label(frmImgRes)      # lblImgRes.pack()      # Button      btnBrowse = ttk.Button(frmBtn, *text*='Browse Image', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=btnBrowseClicked)      btnBrowse.grid(*row*=0, *column*=0, *padx*=10, *pady*=10)      btnRoi = ttk.Button(frmBtn, *text*='ROI', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=btnRoiClicked)      btnRoi.grid(*row*=1, *column*=0, *padx*=10, *pady*=10)      btnSave = ttk.Button(frmBtn, *text*='Save ROI', *style*='success.TButton', *cursor*="hand2", *width*=12, *command*=btnSaveClicked)      btnSave.grid(*row*=0, *column*=1, *padx*=10, *pady*=10)      btnExit = ttk.Button(frmBtn, *text*='Exit', *style*='danger.TButton', *cursor*="hand2", *width*=12, *command*=lambda: exit())      btnExit.grid(*row*=1, *column*=1, *padx*=10, *pady*=10)      # Slider      sldHueMin = ttk.Scale(frmSliderMin, *from\_*=0, *to*=179, *value*=0, *orient*='horizontal', *style*='info.Horizontal.TScale', *length*=255, *command*=sldMove)      lblHueMin = ttk.Label(frmSliderMin, *text*=f'HUE Min : {sldHueMin.get()}', *style*='info.Inverse.TLabel', *width*=15)      lblHueMin.grid(*row*=0, *column*=0, *padx*=20, *pady*=10)      sldHueMin.grid(*row*=0, *column*=1, *padx*=20, *pady*=10)      sldSatMin = ttk.Scale(frmSliderMin, *from\_*=0, *to*=255, *value*=0, *orient*='horizontal', *style*='info.Horizontal.TScale', *length*=255, *command*=sldMove)      lblSatMin = ttk.Label(frmSliderMin, *text*=f'SAT Min : {sldSatMin.get()}', *style*='info.Inverse.TLabel', *width*=15)      lblSatMin.grid(*row*=1, *column*=0, *padx*=20, *pady*=10)      sldSatMin.grid(*row*=1, *column*=1, *padx*=20, *pady*=10)      sldValueMin = ttk.Scale(frmSliderMin, *from\_*=0, *to*=255, *value*=0, *orient*='horizontal', *style*='info.Horizontal.TScale', *length*=255, *command*=sldMove)      lblValueMin = ttk.Label(frmSliderMin, *text*=f'VALUE Min : {sldValueMin.get()}', *style*='info.Inverse.TLabel', *width*=15)      lblValueMin.grid(*row*=2, *column*=0, *padx*=20, *pady*=10)      sldValueMin.grid(*row*=2, *column*=1, *padx*=20, *pady*=10)      sldHueMax = ttk.Scale(frmSliderMax, *from\_*=0, *to*=179, *value*=179, *orient*='horizontal', *style*='info.Horizontal.TScale', *length*=255, *command*=sldMove)      lblHueMax = ttk.Label(frmSliderMax, *text*=f'HUE Max : {sldHueMax.get()}', *style*='info.Inverse.TLabel', *width*=15)      lblHueMax.grid(*row*=0, *column*=0, *padx*=20, *pady*=10)      sldHueMax.grid(*row*=0, *column*=1, *padx*=20, *pady*=10)      sldSatMax = ttk.Scale(frmSliderMax, *from\_*=0, *to*=255, *value*=255, *orient*='horizontal', *style*='info.Horizontal.TScale', *length*=255, *command*=sldMove)      lblSatMax = ttk.Label(frmSliderMax, *text*=f'SAT Max : {sldSatMax.get()}', *style*='info.Inverse.TLabel', *width*=15)      lblSatMax.grid(*row*=1, *column*=0, *padx*=20, *pady*=10)      sldSatMax.grid(*row*=1, *column*=1, *padx*=20, *pady*=10)      sldValueMax = ttk.Scale(frmSliderMax, *from\_*=0, *to*=255, *value*=255, *orient*='horizontal', *style*='info.Horizontal.TScale', *length*=255, *command*=sldMove)      lblValueMax = ttk.Label(frmSliderMax, *text*=f'VALUE Max : {sldValueMax.get()}', *style*='info.Inverse.TLabel', *width*=15)      lblValueMax.grid(*row*=2, *column*=0, *padx*=20, *pady*=10)      sldValueMax.grid(*row*=2, *column*=1, *padx*=20, *pady*=10)      window.title("Color Detection")      # window.geometry("1280x720")      window.resizable(0, 0)      window.mainloop() | |

**Hasil Running Program untuk Proses Ekstraksi ROI**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Citra** | **ROI** | **Keterangan** |
| **1.** |  |  | **Tidak Baik** |
| **2** |  |  | **Kurang Optimal** |
| **3** |  |  | **Sangat Baik** |
| **4** |  |  | **Tidak Baik** |
| **5** |  |  | **Sangat Baik** |
| **6** |  |  | **Sangat Baik** |
| **7** |  |  | **Sangat Baik** |
| **8** |  |  | **Kurang Optimal** |
| **9** |  |  | **Kurang Optimal** |
| **10** |  |  | **Kurang Optimal** |
| **11** |  |  | **Kurang Optimal** |
| **12** |  |  | **Tidak Baik** |
| **13** |  |  | **Kurang Optimal** |
| **14** |  |  | **Kurang Optimal** |
| **15** |  |  | **Kurang Optimal** |
| **16** |  |  | **Tidak Baik** |
| **17** |  |  | **Kurang Optimal** |
| **18** |  |  | **Tidak Baik** |
| **19** |  |  | **Tidak Baik** |
| **20** |  |  | **Kurang Optimal** |
| **21** |  |  | **Tidak Baik** |
| **22** |  |  | **Tidak Baik** |
| **23** |  |  | **Sangat Baik** |
| **24** |  |  | **Sangat Baik** |
| **25** |  |  | **Sangat Baik** |
| **26** |  |  | **Kurang Optimal** |
| **27** |  |  | **Sangat Baik** |
| **28** |  |  | **Sangat Baik** |
| **29** |  |  | **Sangat Baik** |
| **30** |  |  | **Sangat Baik** |
| **31** |  |  | **Kurang Optimal** |
| **32** |  |  | **Tidak Baik** |
| **33** |  |  | **Sangat Baik** |
| **34** |  |  | **Sangat Baik** |
| **35** |  |  | **Sangat Baik** |
| **36** |  |  | **Sangat Baik** |
| **37** |  |  | **Sangat Baik** |
| **38** |  |  | **Sangat Baik** |
| **39** |  |  | **Sangat Baik** |
| **40** |  |  | **Sangat Baik** |
| **41** |  |  | **Sangat Baik** |
| **42** |  |  | **Sangat Baik** |
| **43** |  |  | **Sangat Baik** |
| **44** |  |  | **Sangat Baik** |
| **45** |  |  | **Sangat Baik** |
| **46** |  |  | **Sangat Baik** |
| **47** |  |  | **Sangat Baik** |
| **48** |  |  | **Sangat Baik** |
| **49** |  |  | **Sangat Baik** |
| **50** |  |  | **Sangat Baik** |
| **51** |  |  | **Sangat Baik** |
| **52** |  |  | **Kurang Optimal** |
| **53** |  |  | **Sangat Baik** |
| **54** |  |  | **Sangat Baik** |
| **55** |  |  | **Sangat Baik** |
| **56** |  |  | **Tidak Baik** |
| **57** |  |  | **Sangat Baik** |
| **58** |  |  | **Sangat Baik** |
| **59** |  |  | **Sangat Baik** |
| **60** |  |  | **Sangat Baik** |
| **61** |  |  | **Sangat Baik** |
| **62** |  |  | **Sangat Baik** |
| **63** |  |  | **Sangat Baik** |
| **64** |  |  | **Kurang Optimal** |
| **65** |  |  | **Sangat Baik** |
| **66** |  |  | **Sangat Baik** |
| **67** |  |  | **Sangat Baik** |
| **68** |  |  | **Sangat Baik** |
| **69** |  |  | **Sangat Baik** |
| **70** |  |  | **Sangat Baik** |
| **71** |  |  | **Sangat Baik** |
| **72** |  |  | **Tidak Baik** |
| **73** |  |  | **Sangat Baik** |
| **74** |  |  | **Sangat Baik** |
| **75** |  |  | **Sangat Baik** |
| **76** |  |  | **Sangat Baik** |
| **77** |  |  | **Sangat Baik** |
| **78** |  |  | **Sangat Baik** |
| **79** |  |  | **Kurang Optimal** |
| **80** |  |  | **Sangat Baik** |

**Rekap Hasil Eksperimen Program ROI**

|  |  |  |
| --- | --- | --- |
| **Kategori** | **Jumlah** | **Persentase** |
| **Sangat Baik** | **54/80** | **67,5%** |
| **Kurang Optimal** | **15/80** | **18,75%** |
| **Tidak Baik** | **11/80** | **13,75%** |