**LAPORAN PRAKTIKUM**

**WEEK 5**

**16TIN5033 PENGOLAHAN CITRA DIGITAL**



Disusun Oleh:

Muhammad Aziz Taufiqurrahman 201524014

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2022**

**KODE PROGRAM**

Berikut merupakan penjelasan dari kode program yang dilakukan pada google collab :

|  |
| --- |
| url='https://upload.wikimedia.org/wikipedia/en/7/7d/Lenna\_%28test\_image%29.png'  image\_0 = io.imread(url)  image\_2 = cv.cvtColor(image\_0, cv.COLOR\_BGR2RGB)  final\_frame = cv.hconcat((image\_0, image\_2))  cv2\_imshow(image\_2) |
| Code tersebut digunakan untuk menampilkan citra dari sumber tertentu sesuai dengan tujuannya. Adapun code cv.COLOR\_BGR2RGB digunakan untuk menukarkan kanal rgb pada citra sehingga hasil warna pada citra menjadi berubah/berbeda dengan yang aslinya. Serta fitur concat digunakan untuk menggabungkan satu file dengan file lainnya. |
| def brightness\_add(image):      image = np.asarray(image).astype('uint16')      image = image+100      image = np.clip(image, 0, 255)      new\_image = image.astype('uint8')      new\_image = Image.fromarray(new\_image)      return new\_image |
| Pada fungsi tersebut citra yang masuk akan diubah menjadi sebuah array kemudian tiap elemen array tersebut ditambahkan dengan 100 sehingga menjadikan hasil akhir dari citra tersebut menjadi lebih terang dibandingkan dengan citra originalnya. |
| def brightness\_addcv(image):      new\_image = cv.add(image, 100)      return new\_image |
| Fungsi tersebut digunakan untuk menambahkan brightness antar elemen dengan nilai 100, akan tetapi pada fungsi ini tidak dilakukan secara manual, melainkan menggunakan bantuan dari library OpenCV. Hasil yang diperoleh lebih terang daripada citra sebelumnya (original) |
| def brightness\_subtraction(image):      image = image.astype('uint16')      image = image-100      image = np.clip(image, 0, 255)      image = image.astype('uint8')      return image |
| Fungsi ini digunakan untuk menurunkan kecerahan dari sebuah citra. Hal yang dilakukan adalah menguraikan citra menjadi sebuah array kemudian mengurangi masing-masing elemen array dengan 100, sehingga menjadikan citra tersebut tidak secerah citra original. Karena jika elemen array dikurangkan dengan 100 hasilnya negative, yang dimasukkan ke dalam elemen array adalah 0 (gelap). |
| def brightness\_subtractioncv(image):      new\_image = cv.subtract(image, 100)      new\_image = np.clip(new\_image, 0, 255)      return new\_image |
| Fungsi ini sama seperti fungsi sebelumnya yakni mengurangi/menurunkan kecerahan dari suatu citra akan tetapi dengan bantuan library dari opencv. |
| def brightness\_multiplication(image):      image = np.asarray(image).astype('uint16')      image = image\*1.25      image = np.clip(image, 0, 255)      new\_image = image.astype('uint8')      new\_image = Image.fromarray(new\_image)      return new\_image |
| Fungsi ini digunakan untuk mengalikan setiap elemen array citra dengan bilangan tertentu, sehingga hasil akhir dari citra bisa menjadi lebih terang dari sebelumnya. Hal tersebut dilakukan untuk alternatif selain dengan menambahkan dengan bilangan tertentu. |
| def brightness\_multiplicationcv(image):      new\_image = cv.multiply(image, 1.25)      new\_image= np.clip(new\_image, 0, 255)      return new\_image |
| Fungsi ini juga digunakan untuk mengalikan setiap elemen array citra dengan bilangan tertentu akan tetapi dibantu dengan library opencv. |
| def brightness\_dividecv(image):      new\_image = cv.divide(image, 2)      new\_image= np.clip(new\_image, 0, 255)      return new\_image |
| Fungsi ini digunakan untuk membagi setiap elemen array pada citra dengan bilangan tertentu dan dibanti dengan library opencv. |
| def brightness\_divide(image):      image = np.asarray(image).astype('uint16')      image = image/2      image = np.clip(image, 0, 255)      new\_image = image.astype('uint8')      #new\_image = Image.fromarray(new\_image)      return new\_image |
| Fungsi tersebut digunakan untuk membagi setiap elemen array pada citra dengan bilangan tertentu secara manual. |
| def bitwise\_and(image):      bit\_and = cv.bitwise\_and(image, image)      return bit\_and |
| Fungsi ini digunakan untuk mendapatkan nilai dari sebuah elemen antar citra dengan menggunakan operasi and. |
| def bitwise\_or(image):      bit\_or = cv.bitwise\_or(image, image)      return bit\_or |
| Fungsi ini digunakan untuk mendapatkan nilai dari sebuah elemen antar citra dengan menggunakan operasi or. |
| def bitwise\_not(image):      bit\_not = cv.bitwise\_not(image)      return bit\_not |
| Fungsi ini digunakan untuk mendapatkan nilai dari sebuah elemen antar citra dengan menggunakan operasi not. |
| def bitwise\_xor(image):      bit\_xor = cv.bitwise\_xor(image, image)      return bit\_xor |
| Fungsi ini digunakan untuk mendapatkan nilai dari sebuah elemen antar citra dengan menggunakan operasi xor. |

Adapun code yang digunakan Praktikum Week 5 untuk menampilkan GUI menggunakan salah satu library dari SimplePy, serta code yang digunakan untuk menampilkan beragam pilihan seperti brightness, dll adalah sebagai berikut :

import matplotlib

matplotlib.use('TkAgg')

import matplotlib.pyplot as plt

import numpy as np

import cv2 as cv

import PySimpleGUI as sg

from reportlab.graphics import renderPM

from svglib.svglib import svg2rlg

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

IMAGE\_SIZE = (500, 500)

UI\_STRING\_FORMAT = "Image Preview are 500x500, image are {0}x{1}"

menu\_def = [

    ['&File', ['&Open', '---', 'Exit']],

    ['&Edit', ['Brightness',

               'Inverse (Negative)', 'Restore', 'Histogram Equalization', 'Downsample by 0.5', 'Upsample by 1.5', 'Quantize', 'Low Pass Filter (Average)', 'High Pass Filter (Edge Detection)', 'Band Pass Filter (Sharpening)']],

    ['&View', ['Original Image', 'Histogram']]

]

layout = [

    [sg.Menu(menu\_def, tearoff=False)],

    [sg.Column([[sg.Text("Image Preview")], [sg.Image(size=IMAGE\_SIZE, filename="", key="image")]]), sg.Column([[sg.Text("Original Image")], [

        sg.Image(size=IMAGE\_SIZE, filename="", key="original-image")]], visible=False, key="original-image-container")],

    [sg.Column([[sg.Text("Image Preview Histogram")], [sg.Canvas(

        key="image-histogram", size=IMAGE\_SIZE)]], visible=False, key="image-histogram-wrapper")],

    [sg.Text("Image preview are 500x500", key="ui\_text")],

]

window = sg.Window("PCD", layout)

image\_element = window["image"]

original\_image\_element = window["original-image"]

ui\_text = window["ui\_text"]

image\_histogram = window["image-histogram"]

original\_image = None

shown\_image = None

is\_show\_original\_image = False

is\_show\_histogram = False

fig\_agg = None

def encode\_img(img):

    resized\_img = cv.resize(img, IMAGE\_SIZE)

    return cv.imencode(".png", resized\_img)[1].tobytes()

def draw\_figure(canvas, figure):

    figure\_canvas\_agg = FigureCanvasTkAgg(figure, canvas)

    figure\_canvas\_agg.draw()

    figure\_canvas\_agg.get\_tk\_widget().pack(side='top', fill='both', expand=1)

    return figure\_canvas\_agg

def delete\_fig\_agg(fig\_agg):

    fig\_agg.get\_tk\_widget().forget()

    plt.close('all')

def draw\_histogram(img, canvas):

    for i, col in enumerate(['b', 'g', 'r']):

        hist = cv.calcHist([img], [i], None, [256], [0, 256])

        plt.plot(hist, color=col)

        plt.xlim([0, 256])

    fig = plt.gcf()

    fig.set\_dpi(50)

    return draw\_figure(canvas, fig)

def quantimage(image, k):

    i = np.float32(image).reshape(-1, 3)

    condition = (cv.TERM\_CRITERIA\_EPS + cv.TERM\_CRITERIA\_MAX\_ITER, 20, 1.0)

    ret, label, center = cv.kmeans(

        i, k, None, condition, 10, cv.KMEANS\_RANDOM\_CENTERS)

    center = np.uint8(center)

    final\_img = center[label.flatten()]

    final\_img = final\_img.reshape(image.shape)

    return final\_img

def brightness\_window():

    global shown\_image, image\_element, fig\_agg, image\_histogram

    temp\_shown\_image = shown\_image

    layout = [

        [sg.Text("Brightness value")],

        [sg.Slider(range=(-255, 255), default\_value=0,

                   key="brightness", orientation="horizontal", enable\_events=True, change\_submits=True)],

        [sg.Button("Done")]

    ]

    window = sg.Window("Brightness Adjust", layout)

    slider\_value = 0

    while True:

        event, values = window.read(timeout=100)

        if event == sg.WIN\_CLOSED:

            image\_element.update(data=encode\_img(shown\_image))

            if fig\_agg is not None:

                delete\_fig\_agg(fig\_agg)

            fig\_agg = draw\_histogram(shown\_image,

                                     image\_histogram.TKCanvas)

            break

        elif event == 'Done':

            shown\_image = temp\_shown\_image

            break

        elif slider\_value != int(values["brightness"]):

            value = int(values["brightness"])

            hsv\_image = cv.cvtColor(original\_image, cv.COLOR\_BGR2HSV)

            hsv\_image[:, :, 2] = cv.add(hsv\_image[:, :, 2], value)

            temp\_shown\_image = cv.cvtColor(hsv\_image, cv.COLOR\_HSV2BGR)

            image\_element.update(data=encode\_img(temp\_shown\_image))

            if not fig\_agg is None:

                delete\_fig\_agg(fig\_agg)

            fig\_agg = draw\_histogram(temp\_shown\_image,

                                     image\_histogram.TKCanvas)

    window.close()

while True:

    event, values = window.read()

    print(event)

    if event in (sg.WIN\_CLOSED, 'Exit'):

        break

    if event == 'Open':

        filename = sg.popup\_get\_file("Image File", file\_types=(

            ("Windows Bitmaps", "\*.BMP;\*.DIP"), ("JPEG", "\*.JPEG;\*.JPG;\*.JPE"), ("Portable Network Graphics", "\*.PNG"), ("TIFF files", "\*.TIFF;\*.TIF"), ("GIF Files", "\*.GIF"), ("Vector Graphics", "\*.SVG"), ))

        if filename != None and filename != '':

            file\_format = filename.split(".")[-1]

            if file\_format == "gif":

                cap = cv.VideoCapture(filename)

                \_, image = cap.read()

                cap.release()

                shown\_image = image

            elif file\_format == "svg":

                image = svg2rlg(filename)

                image = np.array(renderPM.drawToPIL(image))

                shown\_image = cv.cvtColor(image, cv.COLOR\_RGB2BGR)

            else:

                shown\_image = cv.imread(filename, cv.IMREAD\_COLOR)

            original\_image = shown\_image

            ui\_text.update(UI\_STRING\_FORMAT.format(

                shown\_image.shape[0], shown\_image.shape[1]))

            original\_image\_element.update(data=encode\_img(original\_image))

            image\_element.update(data=encode\_img(shown\_image))

            if not fig\_agg is None:

                delete\_fig\_agg(fig\_agg)

            fig\_agg = draw\_histogram(

                shown\_image, image\_histogram.TKCanvas)

    elif not shown\_image is None:

        if event == 'Inverse (Negative)':

            shown\_image = 255 - shown\_image

            image\_element.update(data=encode\_img(shown\_image))

        elif event == 'Restore':

            shown\_image = original\_image

            image\_element.update(data=encode\_img(shown\_image))

            ui\_text.update(UI\_STRING\_FORMAT.format(

                shown\_image.shape[0], shown\_image.shape[1]))

        elif event == 'Brightness':

            brightness\_window()

        elif event == 'Original Image':

            is\_show\_original\_image = not is\_show\_original\_image

            window["original-image-container"].update(

                visible=is\_show\_original\_image)

        elif event == "Histogram":

            is\_show\_histogram = not is\_show\_histogram

            window["image-histogram-wrapper"].update(

                visible=is\_show\_histogram

            )

        elif event == 'Histogram Equalization':

            ycbcr\_shown\_image = cv.cvtColor(shown\_image, cv.COLOR\_BGR2YCR\_CB)

            ycbcr\_shown\_image[:, :, 0] = cv.equalizeHist(

                ycbcr\_shown\_image[:, :, 0])

            shown\_image = cv.cvtColor(ycbcr\_shown\_image, cv.COLOR\_YCR\_CB2BGR)

            image\_element.update(data=encode\_img(shown\_image))

        elif event == 'Upsample by 1.5':

            shown\_image = cv.resize(

                shown\_image, (int(shown\_image.shape[0] \* 1.5), int(shown\_image.shape[1] \* 1.5)), interpolation=cv.INTER\_AREA)

            image\_element.update(data=encode\_img(shown\_image))

            ui\_text.update(UI\_STRING\_FORMAT.format(

                shown\_image.shape[0], shown\_image.shape[1]))

        elif event == 'Downsample by 0.5':

            shown\_image = cv.resize(

                shown\_image, (int(shown\_image.shape[0] / 2), int(shown\_image.shape[1] / 2)), interpolation=cv.INTER\_AREA)

            image\_element.update(data=encode\_img(shown\_image))

            ui\_text.update(UI\_STRING\_FORMAT.format(

                shown\_image.shape[0], shown\_image.shape[1]))

        elif event == 'Quantize':

            k\_value = sg.popup\_get\_text("Number of colour:")

            try:

                if k\_value != None:

                    shown\_image = quantimage(shown\_image, int(k\_value))

                    image\_element.update(data=encode\_img(shown\_image))

            except Exception:

                sg.popup("Error", "The value you input are not valid")

        elif event == 'Low Pass Filter (Average)':

            kernel = np.ones((3, 3), np.float32) / 9

            shown\_image = cv.filter2D(shown\_image, -1, kernel)

            image\_element.update(data=encode\_img(shown\_image))

        elif event == 'High Pass Filter (Edge Detection)':

            kernel = np.array(

                [

                    [1, 1, 1],

                    [0, 0, 0],

                    [-1, -1, -1]

                ]

            )

            shown\_image = cv.filter2D(shown\_image, -1, kernel)

            image\_element.update(data=encode\_img(shown\_image))

        elif event == 'Band Pass Filter (Sharpening)':

            kernel = np.array(

                [

                    [0, -1, 0],

                    [-1, 5, -1],

                    [0, -1, 0]

                ]

            )

            shown\_image = cv.filter2D(shown\_image, -1, kernel)

            image\_element.update(data=encode\_img(shown\_image))

        if not fig\_agg is None:

            delete\_fig\_agg(fig\_agg)

        fig\_agg = draw\_histogram(

            shown\_image, image\_histogram.TKCanvas)

window.close()

**TAMPILAN ANTARMUKA PROGRAM**

1. Tampilan awal

Ukuran foto yang akan dimasukan harus sudah disesuaikan dengan permintaan yakni berukuran 500 x 500. Jika ukuran citra lebih atau kurang dari yang diminta, maka sistem akan secara otomatis mengubah ukuran nya menjadi 500 x 500.

**Graphical user interface

Description automatically generated with medium confidence**

1. Proses pengambilan folder citra

**Graphical user interface, application

Description automatically generated**

1. File setelah diupload ke dalam sistem

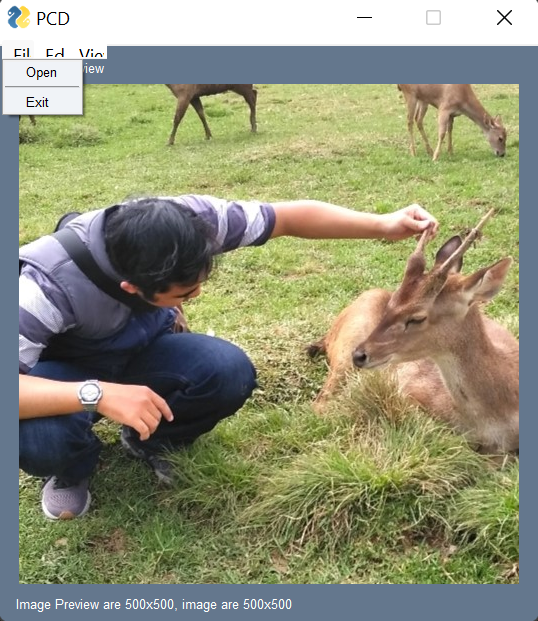
Setelah file foto dipload ke dalam sistem, maka hasilnya adalah sebagai berikut

**A person petting a deer

Description automatically generated with medium confidence**

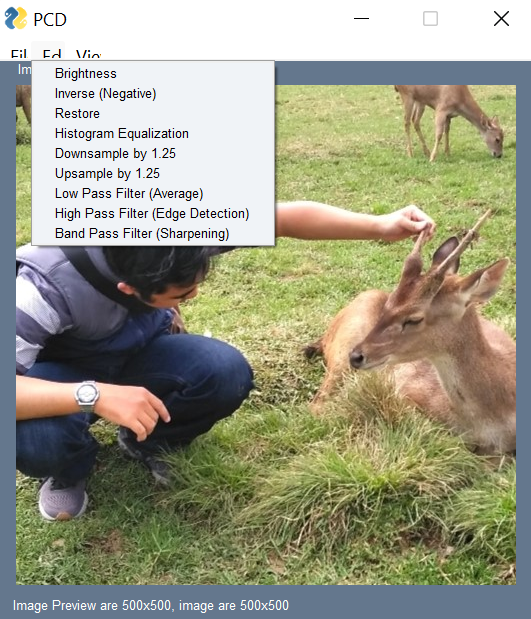
1. Pilihan yang bisa user gunakan untuk perubahan pada citra tersebut

* Saat memilih file

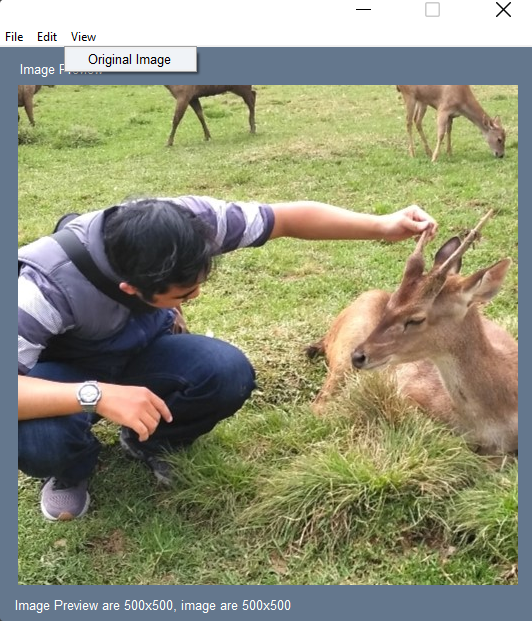
****

* Saat memilih Edit

1. Brightness digunakan untuk meningkatkan kecerahan dari suatu citra
2. Inverse negative adalah menampilkan citra negative dari file yang dipilih, cara yang dilakukan adalah 255 – kanal citra
3. Restore adalah untuk mereset citra yang sudah dilakukan pengeditan
4. Histogram equalization
5. Downsample by 1.25 adalah penurunan kualitas sebuah citra sebesar 1.25
6. Upsample by 1.25 adalah penaikan kualitas sebuah citra sebesar 1.25
7. Low pass filter adalah mencari nilai rata-rata dari sebuah array citra image
8. High pass filter adalah digunakan untuk menemukan tepi-tepi pada sebuah citra
9. Band pass filter digunakan untuk menajamkan hasil dari sebuah citra

****

* Saat memilih view

****

1. Original image digunakan untuk melihat perbandingan antara image yg original dengan image yang sudah dilakukan pengeditan

Untuk code keseluruhan bisa diakses pada github : <https://github.com/aziztaufiqurrahman/PCD/tree/main/Pertemuan%205>

Untuk hasil dari program bisa dilihat pada link berikut :

**KENDALA**

Pada pengerjaan praktikum ini kendala yang saya peroleh adalah pada saat menentukan logika dan syntax dari teori yang sudah disampaikan. Kesulitan yang ditemukan seperti tertukarnya antara kanal rgb sehingga tidak sesuai dengan yang diharapkan, proses downsampling karena harus menurunkan elemen citranya, dan upsampling.

**SOLUSI**

Solusi yang dilakukan adalah dengan mempelajari kembali teori tentang pengolahan citra digital ini dibarengi dengan melihat tutorial yang sudah ada serta melihat proses penyusunan code untuk kasus-kasus tertentu

**LESSON LEARN**

Bisa mengolah sebuah citra dengan berbagai macam pilihan