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(
```

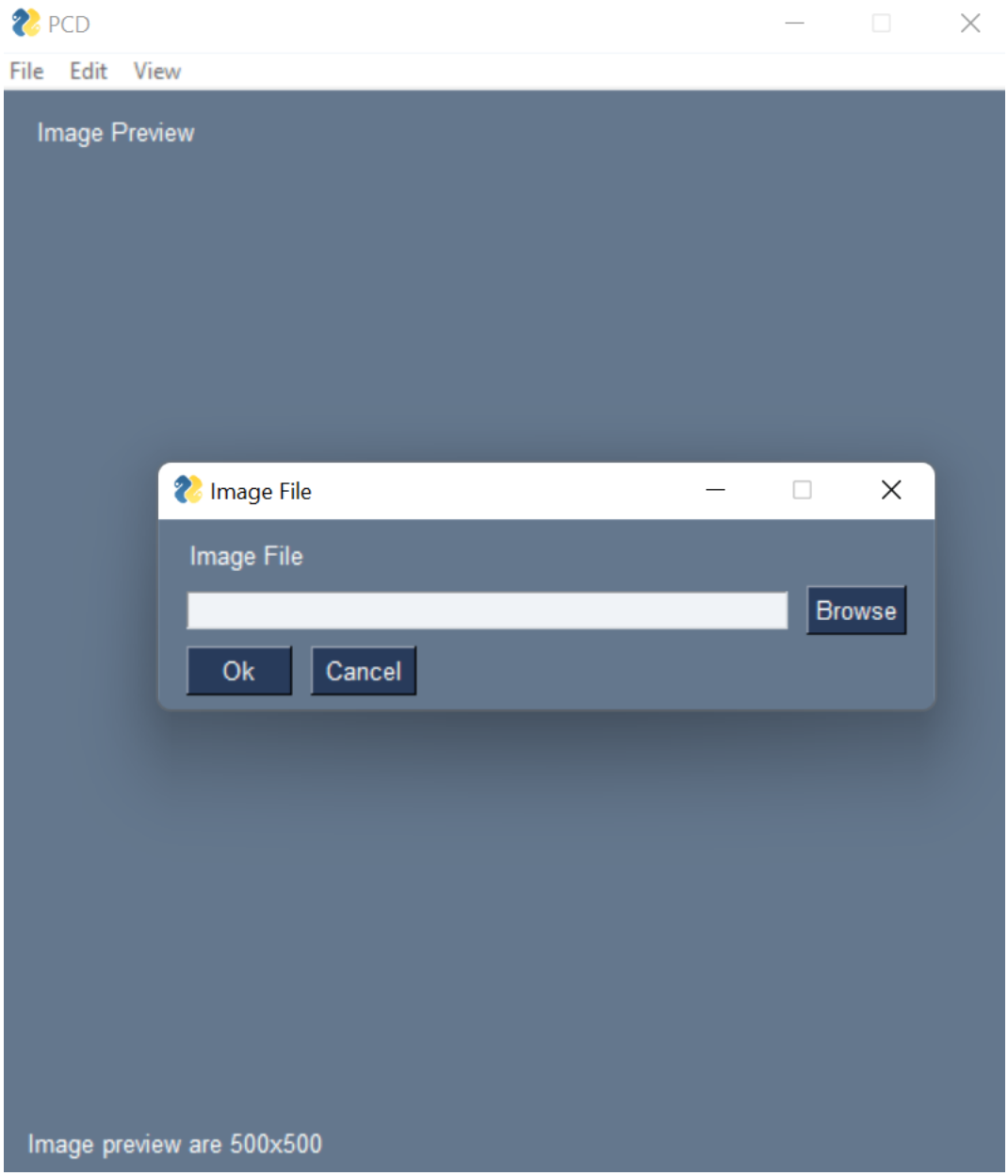
```
                [
                    [0, -1, 0],
                    [-1, 5, -1],
                    [0, -1, 0]
                ]
            )
            shown_image = cv.filter2D(shown_image, -1, kernel)

            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

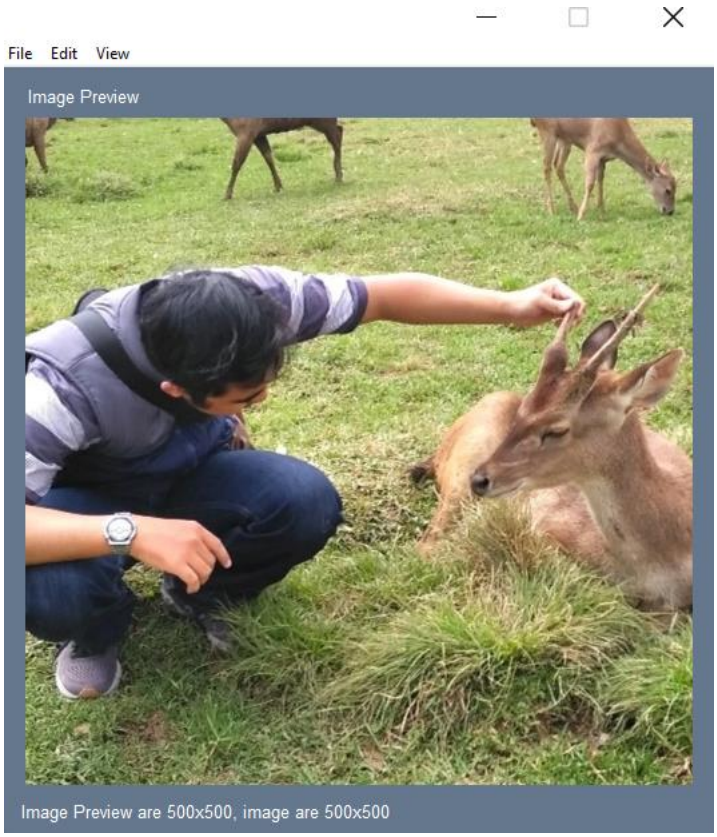
   Ukuran foto yang akan dimasukan harus sudah disesuaikan dengan permintaan yakni berukuran 500 x 500. Jika ukuran citra lebih atau kurang dari yang diminta, maka sistem akan secara otomatis mengubah ukuran nya menjadi 500 x 500.

2. Proses pengambilan folder citra

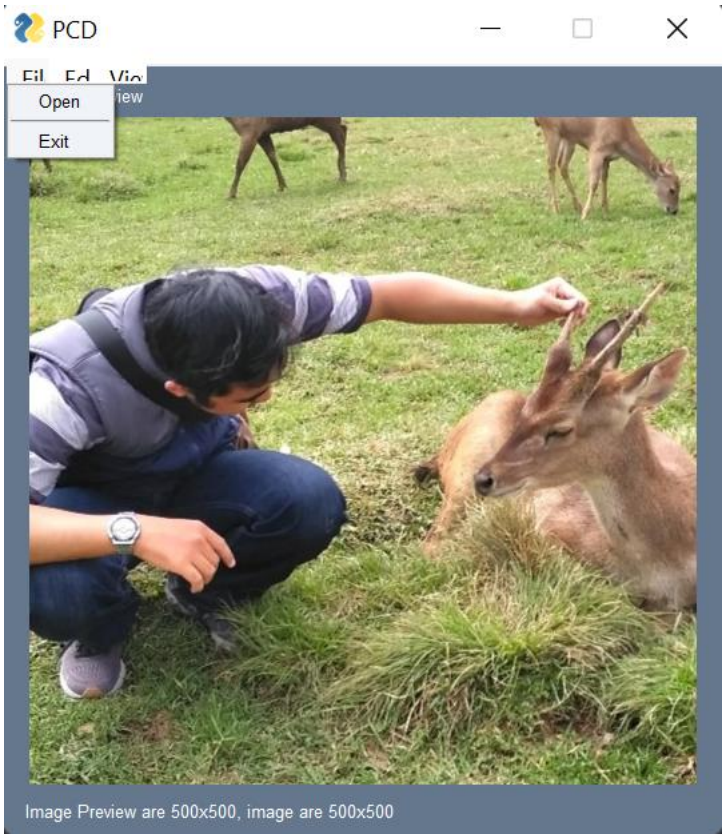

PCD
File
Edit
View
Image Preview
Image File
Image File
Browse
Ok
Cancel
Image preview are 500x500

3. File setelah diupload ke dalam sistem

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

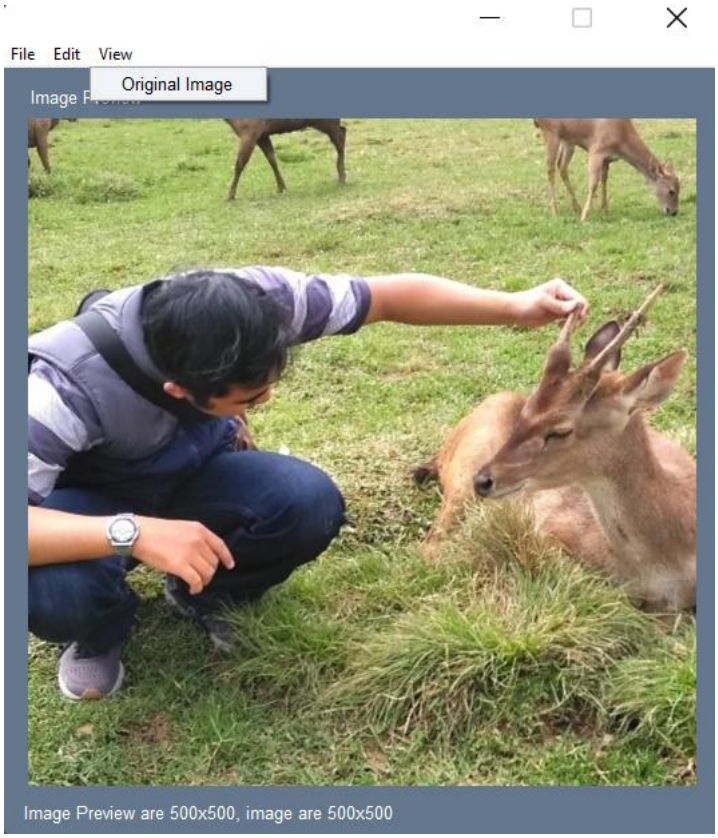


4. Pilihan yang bisa user gunakan untuk perubahan pada citra tersebut
    - Saat memilih file



- Saat memilih Edit
    a. Brightness digunakan untuk meningkatkan kecerahan dari suatu citra
    b. Inverse negative adalah menampilkan citra negative dari file yang dipilih, cara yang dilakukan adalah 255 – kanal citra
    c. Restore adalah untuk mereset citra yang sudah dilakukan pengeditan
    d. Histogram equalization
    e. Downsample by 1.25 adalah penurunan kualitas sebuah citra sebesar 1.25
    f. Upsample by 1.25 adalah penaikan kualitas sebuah citra sebesar 1.25
    g. Low pass filter adalah mencari nilai rata-rata dari sebuah array citra image
    h. High pass filter adalah digunakan untuk menemukan tepi-tepi pada sebuah citra
    i. Band pass filter digunakan untuk menajamkan hasil dari sebuah citra

- Saat memilih view



    a. 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 :

https://youtu.be/yNeHFQPaJ7U

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