Ahmad Siddiq Priaji (22/496854/PA/21370)

Tugas Metode Numerik Penerapan Finite Difference untuk Edge Detection

Import Library yang Diperlukan

Menggunakan library sebagi berikut.

- numpy
- cv2
- pyplot
- math

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
import math
```

Membaca File Image dan Membuat Fungsi untuk Menampilkan Image

```
image = cv.imread('supra.jpg') # menggunakan cv2 untuk membaca file image
image = image[:,:,::-1] # Agar warna image tidak terbaik

# Menampilkan image
def display(img):
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    plt.show()
    return

display(image)
```

Mengubah Image dari RGB ke Grayscale

Mengubah image dari RGB ke Grayscale menggunakan metode weighted average.

```
# fungsi mengubah image dari RGB ke Grayscale
def rgb_to_gray(img):
    gray_image = np.zeros(img.shape[:2])  # Array untuk gambar
grayscale
    # Mengambil warna RGB
    R = np.array(img[:, :, 0])
    G = np.array(img[:, :, 1])
    B = np.array(img[:, :, 2])
    # Minghitung rata-rata weighted RGB
    avg = ((R * 0.299) + (G * 0.587) + (B * 0.114))
    # Mengisi array gambar grayscale
    gray_image[:, :] = np.round(avg)
    return gray_image

gray_image = rgb_to_gray(image)

display(gray_image)
print(gray_image)
```

```
[[23. 23. 22. ... 22. 22. 21.]
 [23. 23. 22. ... 23. 23. 23.]
 [23. 23. 22. ... 23. 23. 23.]
 ...
 [23. 23. 22. ... 23. 22. 23.]
 [25. 25. 25. ... 23. 23. 23.]
 [22. 22. 21. ... 24. 23. 23.]]
```

Membuat Fungsi untuk Menghitung Difference Sumbu-x dan Sumbu-y

Perhitungan nilai gradient dengan menggunakan Central Difference, Forward Difference, dan Backward Difference untuk sumbu-x dan sumbu-y.

Central Difference: $f'(x) = \frac{f(x+h) - f(x-h)}{2h}$

Forward Difference: $f'(x) = \frac{f(x+h) - f(x)}{h}$

Backward Difference: $f'(x) = \frac{f(x) - f(x-1)}{h}$

```
def central_diff_x(gray_img, x, y):
    gradient_x = int(int(gray_img[x+1,y]) - int(gray_img[x-1,y]))/2
    return gradient_x

def central_diff_y(gray_img, x, y):
    gradient_y = int(int(gray_img[x,y+1]) - int(gray_img[x,y-1]))/2
    return gradient_y
```

```
def forward_diff_x(gray_img, x, y):
    gradient_x = int(gray_img[x+1,y]) - int(gray_img[x,y])
    return gradient_x

def forward_diff_y(gray_img, x, y):
    gradient_y = int(gray_img[x,y+1]) - int(gray_img[x,y])
    return gradient_y

def backward_diff_x(gray_img, x, y):
    gradient_x = int(gray_img[x,y]) - int(gray_img[x-1,y])
    return gradient_x

def backward_diff_y(gray_img, x, y):
    gradient_y = int(gray_img[x,y]) - int(gray_img[x,y-1])
    return gradient_y
```

Membuat Fungsi untuk Menghitung Gradient Vector

Perhitungan nilai gradient kebanyakan menggunakan Central Difference. Namun, pada pixel Image paling pinggir tidak dapat dilakukan. Karena itu, diperlukan perhitungan menggunakan Forward Difference atau Backward Difference.

Forward Difference digunakan ketika pixel sumbu-x = 0 dan sumbu-y = 0.

Backward Difference digunakan ketika pixel sumbu-x = (max_x - 1) dan sumbu-y = (max_y - 1).

Setelah didapatkan gradient sumbu-x dan sumbu-y dilakukan perhitungan gradient vector dengan rumus berikut.

$$|)\nabla f|)=\sqrt{f_x^2+f_y^2}$$

```
def gradient_vector(gray_img, x, y, max_x, max_y):
    if x==0: # Untuk pixel paling kiri
        gradient_x = forward_diff_x(gray_img, x, y)
    elif x==max_x-1: # Untuk pixel paling kanan
        gradient_x = backward_diff_x(gray_img, x, y)
    else:
        gradient_x = central_diff_x(gray_img, x, y)
    if y==0: # Untuk pixel paling atas
        gradient_y = forward_diff_y(gray_img, x, y)
    elif y==max_y-1: # Untuk pixel paling atas
        gradient_y = backward_diff_y(gray_img, x, y)
    else:
        gradient_y = central_diff_y(gray_img, x, y)
    gradient_vector = int(math.sqrt(gradient_x**2 + gradient_y**2))
    return gradient_vector
```

Membuat Fungsi Edge Detection

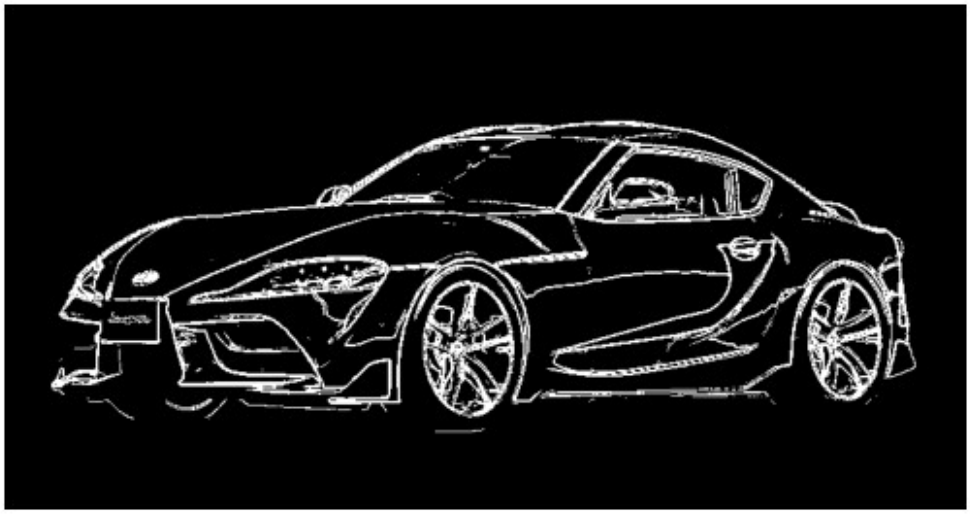
Membuat image yang menampilkan edge dengan melakukan perhitungan dan pengecekan di setiap pixel. Apabila nilai gradient vector lebih besar daripada threshold, maka pixel tersebut merupakan edge. Pixel edge diberi nilai 255 sedangkan pixel bukan edge diberi nilai 0.

```
def edge_detection(gray_img, threshold):
    edge_img = np.zeros(gray_img.shape)

    for x in range(gray_img.shape[0]):
        for y in range(gray_img.shape[1]):
            gradient_vector_img = gradient_vector(gray_img, x, y,
gray_img.shape[0], gray_img.shape[1])
            if gradient_vector_img > threshold:
                edge_img[x,y] = 255 # menghasilkan warna putih
            else:
                edge_img[x,y] = 0 # menghasilkan warna hitam
    return edge_img
```

Melakukan Edge Detection dan Menampilkan Edge dalam Hitam Putih

```
edge_image = edge_detection(gray_image, 15)
display(edge_image)
```



Membuat Fungsi untuk Menampilkan Edge pada Image Awal

Untuk menampilkan edge pada image awal, dapat dilakukan dengan mengganti nilai G (green) pada pixel edge dengan nilai 255.

```
def show_edge(img, edge_img):
    for x in range(img.shape[0]):
        for y in range(img.shape[1]):
            if edge_img[x,y] == 255:
                img[x,y] = [0, 255, 0]
    return img
```

Menampilkan Edge pada Image Awal

```
new_image = show_edge(image, edge_image)
display(new_image)
```