ROBOT PERCEPTION 3
TASK 2 : Sobel Filter

Name: Aniruddhan
Reg: 21BRS1682

Aim: To apply the Sobel Filter to get the edge detection, in order to know image boundaries.

Algorithm:

Step 1: Import Required Libraries

1. Import necessary libraries:
    - numpy: For handling arrays and performing numerical operations.
    - scipy.ndimage: For applying the Sobel filter, which is used to detect edges in images.
    - skimage.io: For reading and saving images.
    - skimage.color: For converting color images to grayscale.
    - skimage.util: For converting image formats.
    - matplotlib.pyplot: For displaying images.

Step 2: Load and Convert Image to Grayscale

1. Define the path to the image file.
2. Load the image using io.imread(image_path).
3. Convert the loaded image to grayscale using color.rgb2gray(image). Grayscale conversion simplifies the image to a single channel, which is ideal for edge detection.

Step 3: Define the Sobel Filter Function

1. **Sobel Filter Function:**

- Define a function sobel_filter(image) that takes a grayscale image as input.
- Apply the Sobel filter along the x-axis using ndi.sobel(image, axis=0), which detects vertical edges.
- Apply the Sobel filter along the y-axis using ndi.sobel(image, axis=1), which detects horizontal edges.
- Compute the gradient magnitude using np.hypot(sobel_x, sobel_y), which combines the results from both axes to detect overall edge strength.
- Normalize the gradient magnitude to a range of [0, 1] by subtracting the minimum value and dividing by the range (difference between the maximum and minimum values).

2. **Return the normalized magnitude** as the result of the Sobel filter function.

Step 4: Apply the Sobel Filter to the Grayscale Image

1. Call the sobel_filter(gray_image) function with the grayscale image as input.
2. Store the result in sobel_result, which contains the edge-detected image.

Step 5: Convert the Filtered Image to 8-bit Format

1. Define a function convert_to_ubyte(image) that converts the filtered image to 8-bit format using util.img_as_ubyte(image). This step ensures compatibility with standard image formats and prepares the image for saving or further processing.
2. Convert the sobel_result to 8-bit format and store the result in sobel_result_ubyte.

Step 6: Save the Sobel Filtered Image

1. Save the 8-bit Sobel-filtered image to disk using io.imsave('sobel_filtered_image.png', sobel_result_ubyte).

Step 7: Display the Sobel Filtered Image

1. Define a function display_image(image, title) that takes an image and a title as input.

- Use plt.figure(figsize=(8, 8)) to create a new figure for the image.
- Display the image using plt.imshow(image, cmap='gray') to show it in grayscale.
- Set the title using plt.title(title) and hide the axis using plt.axis('off').
- Use plt.show() to display the image.

2. Call the display_image(sobel_result, 'Sobel Filter') function to visualize the Sobel-filtered image.

Code:

```
import numpy as np
import scipy.ndimage as ndi
from skimage import io, color, util
import matplotlib.pyplot as plt

# Load and convert image to grayscale
image_path = "/content/img1.jpg"
image = io.imread(image_path)
gray_image = color.rgb2gray(image)  # Convert to grayscale

# Sobel Filter
def sobel_filter(image):
    sobel_x = ndi.sobel(image, axis=0)
    sobel_y = ndi.sobel(image, axis=1)
    magnitude = np.hypot(sobel_x, sobel_y)
    # Normalize magnitude to [0, 1]
    magnitude = (magnitude - np.min(magnitude)) / (np.max(magnitude) - np.min(magnitude))
    return magnitude

# Apply filters
sobel_result = sobel_filter(gray_image)

# Convert results to 8-bit format
def convert_to_ubyte(image):
    return util.img_as_ubyte(image)

sobel_result_ubyte = convert_to_ubyte(sobel_result)

io.imsave('sobel_filtered_image.png', sobel_result_ubyte)
```

```
# Display images using matplotlib
def display_image(image, title):
    plt.figure(figsize=(8, 8))
    plt.imshow(image, cmap='gray')
    plt.title(title)
    plt.axis('off')  # Hide axis
    plt.show()

display_image(sobel_result, 'Sobel Filter')
```

O/P: