Project Documentation

Overview

Project Name	1. Image Resizing 2. Image Edge Detection
Background	This project focuses on fundamental image processing techniques using OpenCV. It involves two main tasks: image resizing and edge detection. The resizing part demonstrates how to scale images up or down using different interpolation methods, which is essential for optimizing image dimensions in various applications. The edge detection part applies the Canny algorithm to identify sharp intensity changes, helping to highlight object boundaries and important features within the image—crucial for tasks like object recognition and segmentation.
Objectives	• To resize images using upscaling and downscaling techniques with OpenCV. • To detect edges in images using the Canny edge detection algorithm.
Target Audience	This project is targeted at beginners and students interested in learning fundamental image processing techniques using Python and OpenCV.

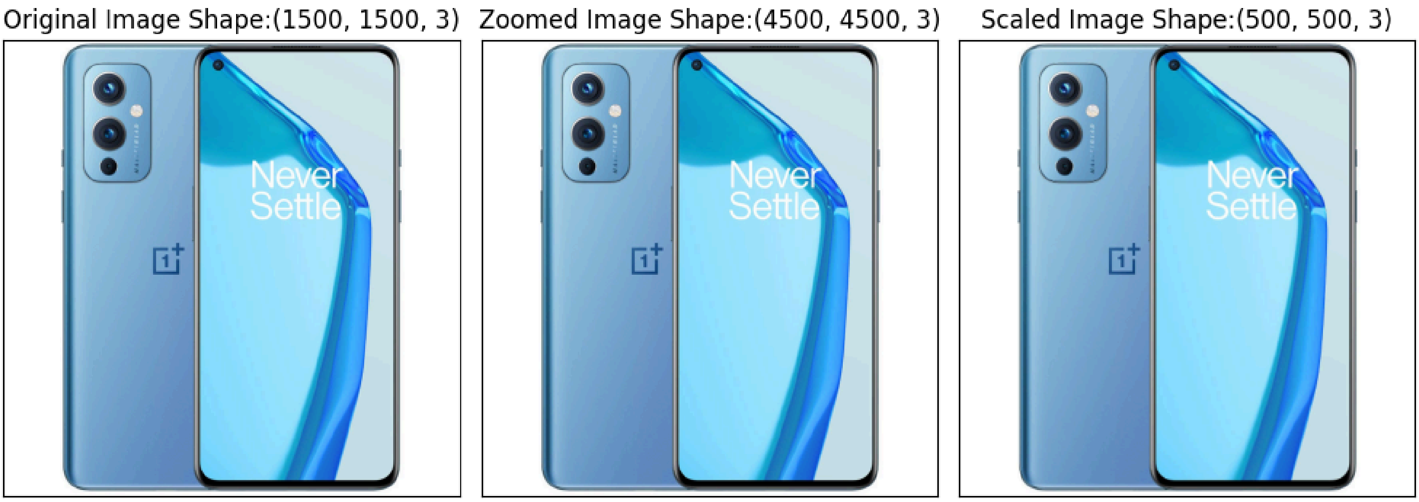
Project 1 : Image Resizing

Python Source Code:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

import os



# Verify the file path

print(os.listdir('/content/'))  # List files in the current directory to ensure the image is there



# Load the image with the correct path

image_path = '/content/one_plus.jpg'  # Updated file path

image = cv2.imread(image_path)



# Check if the image is loaded

if image is None:

    print(f"Failed to load the image from path: {image_path}")

else:

    print(f"Image loaded successfully from: {image_path}")



    # Convert image to RGB (OpenCV loads images in BGR format)

    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)



    # Define scale factors for resizing

    scale_factor_1 = 3.0

    scale_factor_2 = 1/3.0



    # Get the height and width of the original image
    height, width = image_rgb.shape[:2]


    # Resize the image (Zoomed in version)

    new_height = int(height * scale_factor_1)

    new_width = int(width * scale_factor_1)

    zoomed_image = cv2.resize(src=image_rgb,

                              dsize=(new_width, new_height),

                              interpolation=cv2.INTER_CUBIC)


    # Resize the image (Scaled down version)

    new_height1 = int(height * scale_factor_2)

    new_width1 = int(width * scale_factor_2)

    scaled_image = cv2.resize(src=image_rgb,

                              dsize=(new_width1, new_height1),

                              interpolation=cv2.INTER_AREA)


    # Plotting the original, zoomed, and scaled images

    fig, axs = plt.subplots(1, 3, figsize=(10, 4))

    axs[0].imshow(image_rgb)

    axs[0].set_title('Original Image Shape:'+str(image_rgb.shape))

    axs[1].imshow(zoomed_image)

    axs[1].set_title('Zoomed Image Shape:'+str(zoomed_image.shape))

    axs[2].imshow(scaled_image)

    axs[2].set_title('Scaled Image Shape:'+str(scaled_image.shape))


    # Remove x and y ticks for cleaner visualization

    for ax in axs:
```

```
    ax.set_xticks([])

    ax.set_yticks([])


# Adjust layout and display the plot

plt.tight_layout()

plt.show()
```

Output :


Original Image Shape:(1500, 1500, 3)
Zoomed Image Shape:(4500, 4500, 3)
Scaled Image Shape:(500, 500, 3)
Never Settle
Never Settle
Never Settle

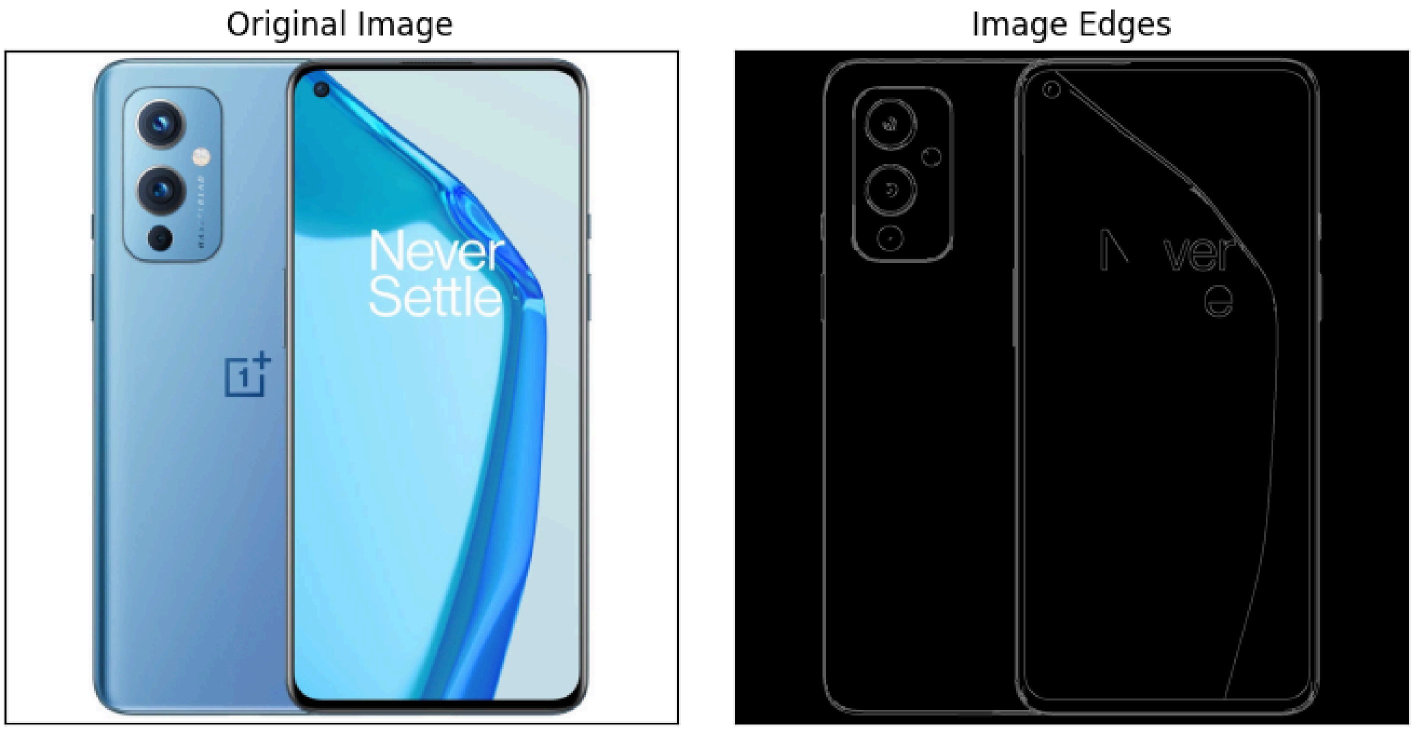
Project 2 : Image Edge Detection

Python Source Code :

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

import os


# Verify if the file exists

print("Files in /content/:", os.listdir('/content/'))


# Load the uploaded image with correct name
```

```python
img = cv2.imread('/content/one_plus.jpg')  # <- Updated path

# Check if image is loaded
if img is None:
    print("Error: Could not load image.")
else:
    # Convert BGR to RGB
    image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Apply Canny edge detection
    edges = cv2.Canny(image_rgb, 100, 700)

    # Plot the original and edge-detected images
    fig, axs = plt.subplots(1, 2, figsize=(8, 4))
    axs[0].imshow(image_rgb)
    axs[0].set_title('Original Image')

    axs[1].imshow(edges, cmap='gray')  # Set colormap to gray for edge image
    axs[1].set_title('Image Edges')

    # Remove axis ticks
    for ax in axs:
        ax.set_xticks([])
        ax.set_yticks([])

    plt.tight_layout()
    plt.show()
```

Output :



Conclusion :

These projects demonstrate essential image processing techniques—resizing and edge detection—using OpenCV. Resizing helps in adjusting image dimensions for various applications, while edge detection highlights key features within images, aiding in visual analysis and computer vision tasks. Through hands-on implementation and visualization, this work reinforces the foundational concepts of image manipulation and feature extraction, providing a practical understanding useful for further studies in digital image processing.