Arithmetic Operations on Images

Introduction

Arithmetic operations on images involve pixel-wise addition, subtraction, multiplication, and division using image matrices. These operations are important for image blending, brightness adjustment, change detection, and combining images in various proportions. In this lab, we apply arithmetic operations using OpenCV's cv2.addWeighted() function and NumPy-based manual operations for combining two images.

Operations and Significance:

1.Addition: Combines pixel intensities (e.g., I_3(r,c) = I_1(r,c) + I_2(r,c)). Used for image blending, noise averaging, or overlaying images.

2.Subtraction: Computes the difference between pixel intensities (e.g., I_3(r,c) = |I_1(r,c) - I_2(r,c)|). Useful for change detection, motion detection, or background subtraction.

3.Multiplication: Scales pixel intensities (e.g., I_3(r,c) = k * I_1(r,c)). Applied in image enhancement or masking.

4.Division: Normalizes or compares pixel intensities (e.g., I_3(r,c) = I_1(r,c) / I_2(r,c)). Used in image normalization or ratio-based analysis.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Load and Resize Images

```
# Read images in grayscale mode
img1= cv2.imread('/content/drive/MyDrive/Colab Notebooks/imageprocessingimages/everest.jpg', 0)
img2= cv2.imread('/content/drive/MyDrive/Colab Notebooks/imageprocessingimages/flag.jpg', 0)

# Resize both images to the same size for arithmetic operations
img1 = cv2.resize(img1, (250, 250))
img2 = cv2.resize(img2, (250, 250))
```

Image Addition (Weighted)

```
# Adds img1 and img2 with weights 0.7 and 0.3 respectively
# This blends the two images together — brighter areas where both images overlap
img_add = cv2.addWeighted(img1, 0.7, img2, 0.3, 0)

# Display the result
plt.imshow(img_add, cmap='gray')
plt.title('Weighted Addition')
plt.show()
```

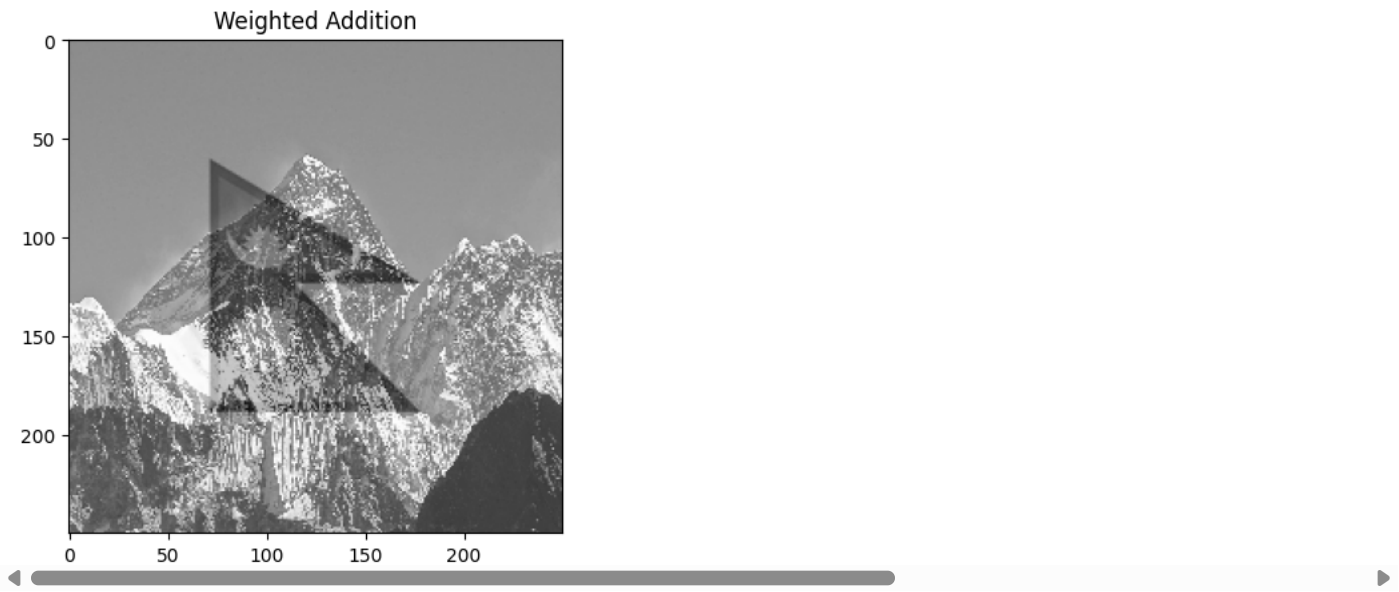

Image Subtraction

```
# Subtracts img2 from img1 pixel-wise
# Highlights areas where img1 is brighter than img2 and suppresses darker areas
img_subtract = cv2.subtract(img1, img2)

# Display the result
plt.imshow(img_subtract, cmap='gray')
plt.title('Subtraction')
plt.show()
```

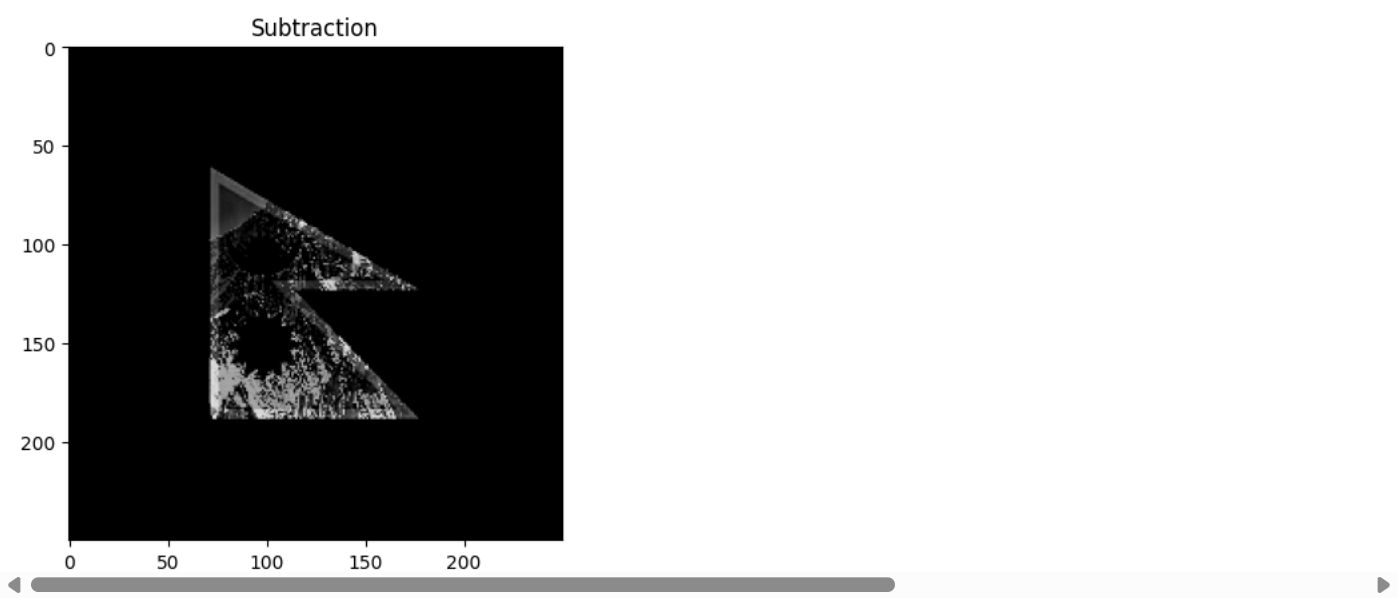

Image Multiplication

```
# Convert images to float32 and scale to [0,1]
img1_float = img1 / 255.0
img2_float = img2 / 255.0

# Perform pixel-wise multiplication in float
img_multiply = img1_float * img2_float

# Rescale back to [0,255]
img_multiply_scaled = np.uint8(img_multiply * 255
```

```
# Display the result
plt.imshow(img_multiply_scaled, cmap='gray')
plt.title('Proper Multiplication')
plt.show()
```

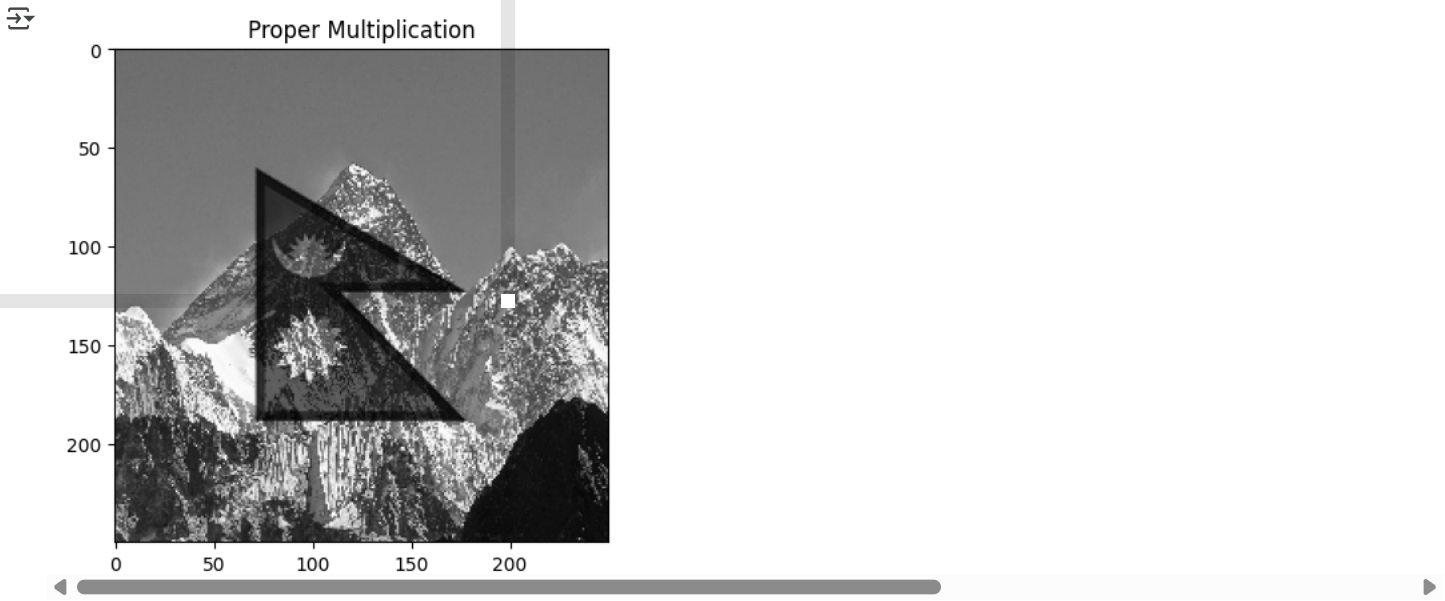

Image Division

```
# Divides pixels of img1 by pixels of img2 (adding 1 to avoid division by zero)
# Highlights areas where img1 is relatively brighter compared to img2
img_divide = cv2.divide(img1, img2 + 1)

# Display the result
plt.imshow(img_divide, cmap='gray')
plt.title('Division')
plt.show()
```