Image Processing Assignment 9

Color Image Reading, Writing, and Manipulations

Aditya Chavan
Roll No: 231080019
T.Y B.Tech IT

Date: October 25, 2025

1 Aim

To write a Python program for reading, writing, and performing basic manipulations on color images using OpenCV.

2 Theory

Color image manipulation refers to operations that modify or enhance the pixel values of images to achieve a desired effect. Digital color images are typically represented in the RGB model, where each pixel contains red, green, and blue intensity values. These can be converted into different color spaces like HSV or Grayscale for specific operations such as segmentation, filtering, or enhancement.

2.1 Key Concepts

1. **Image Reading and Writing:** OpenCV uses `cv2.imread()` and `cv2.imwrite()` for image I/O.
2. **Color Channels:** Images are stored as 3D arrays — (Height $\times$ Width $\times$ 3). Each channel corresponds to Blue, Green, and Red.
3. **Color Conversion:** Converting between color spaces using `cv2.cvtColor()`.
4. **Manipulations:** Includes operations like brightness/contrast adjustment, resizing, cropping, and channel merging/splitting.

2.2 Applications

- Preprocessing for computer vision.
- Enhancing visual quality of images.
- Preparing datasets for ML/AI applications.

3 Algorithm

1. Read a color image from disk using `cv2.imread()`.

2. Display the image using Matplotlib.
3. Split the image into R, G, and B channels using `cv2.split()`.
4. Merge the channels using `cv2.merge()`.
5. Convert the image to grayscale and HSV formats.
6. Adjust brightness and contrast using `cv2.convertScaleAbs()`.
7. Resize and crop the image.

4 Implementation

The implementation was carried out in Google Colab using Python and OpenCV.

4.1 Required Packages

```
!pip install opencv-python matplotlib numpy
```

4.2 Code Implementation

Google Colab Notebook: Click here to view the Colab Notebook

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

```python
# 1. Read the image
img = cv2.imread('771476.jpg')  # Upload your image
print("Image shape:", img.shape)
```

```python
# 2. Display original image
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis('off')
plt.show()
```

```python
# 3. Split RGB channels
B, G, R = cv2.split(img)
plt.figure(figsize=(10,3))
plt.subplot(1,3,1); plt.imshow(B, cmap='gray'); plt.title("Blue
    Channel")
plt.subplot(1,3,2); plt.imshow(G, cmap='gray'); plt.title("Green
    Channel")
plt.subplot(1,3,3); plt.imshow(R, cmap='gray'); plt.title("Red
    Channel")
plt.show()
```

```python
# 4. Merge channels
merged = cv2.merge((B, G, R))
```

```python
# 5. Convert to Grayscale and HSV
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

plt.figure(figsize=(10,3))
plt.subplot(1,2,1); plt.imshow(gray, cmap='gray'); plt.title("
    Grayscale")
plt.subplot(1,2,2); plt.imshow(cv2.cvtColor(hsv, cv2.
    COLOR_HSV2RGB)); plt.title("HSV Image")
plt.show()
```

```python
# 6. Brightness and Contrast Adjustment
bright = cv2.convertScaleAbs(img, alpha=1.2, beta=40)
plt.imshow(cv2.cvtColor(bright, cv2.COLOR_BGR2RGB))
plt.title("Brightness & Contrast Adjusted")
plt.axis('off')
plt.show()
```

```python
# 7. Resize and Crop
resized = cv2.resize(img, (300, 200))
cropped = img[50:200, 100:300]
plt.figure(figsize=(10,3))
plt.subplot(1,2,1); plt.imshow(cv2.cvtColor(resized, cv2.
    COLOR_BGR2RGB)); plt.title("Resized")
plt.subplot(1,2,2); plt.imshow(cv2.cvtColor(cropped, cv2.
    COLOR_BGR2RGB)); plt.title("Cropped")
plt.show()
```

5 Results and Analysis

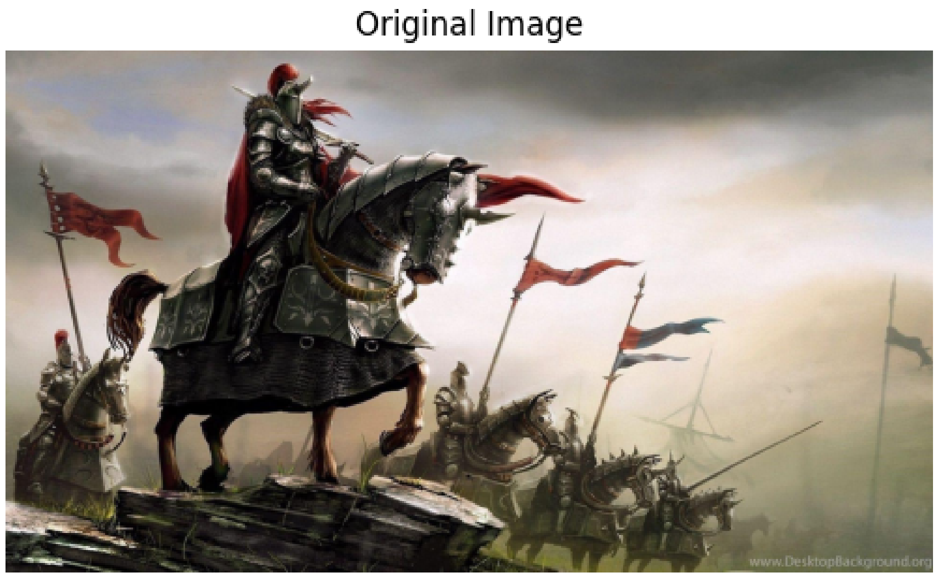

Figure 1: Original Color Image

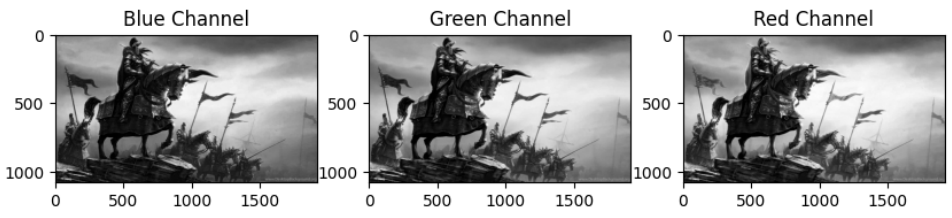

Figure 2: Separated R, G, and B Channels

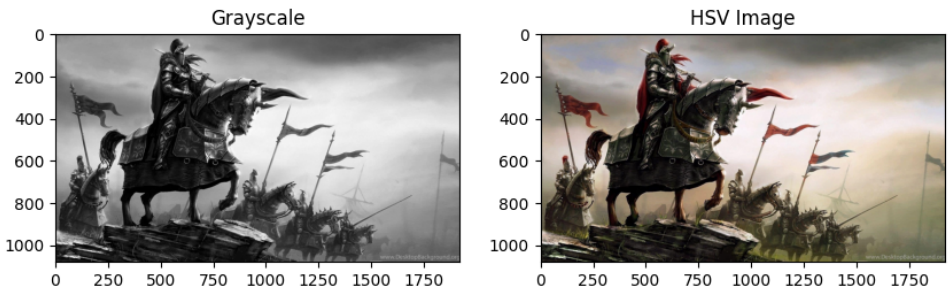

Figure 3: Converted Grayscale and HSV Images

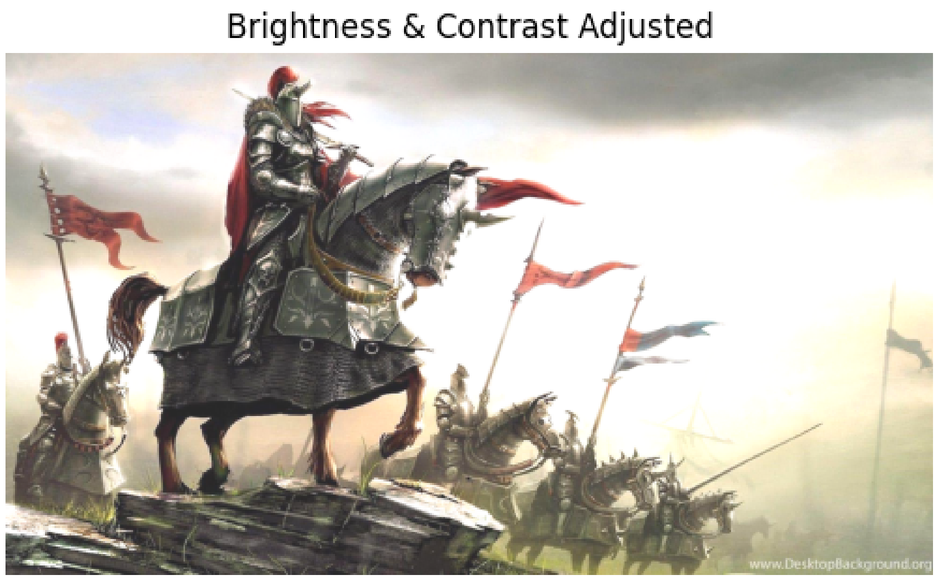

Figure 4: Brightness Adjusted Image

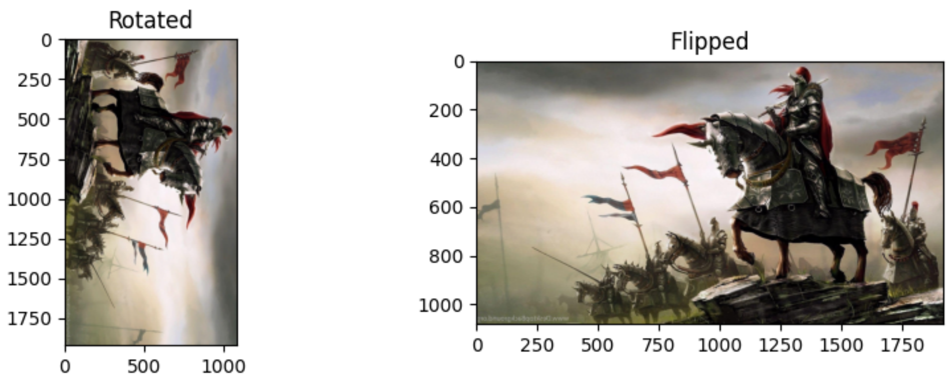

Figure 5: Rotated Image

6 Conclusion

This experiment demonstrated how to read, display, and manipulate color images using OpenCV. Various operations like channel splitting, color space conversion, and brightness/contrast adjustment were successfully implemented. These operations form the foundation for more advanced image processing techniques.