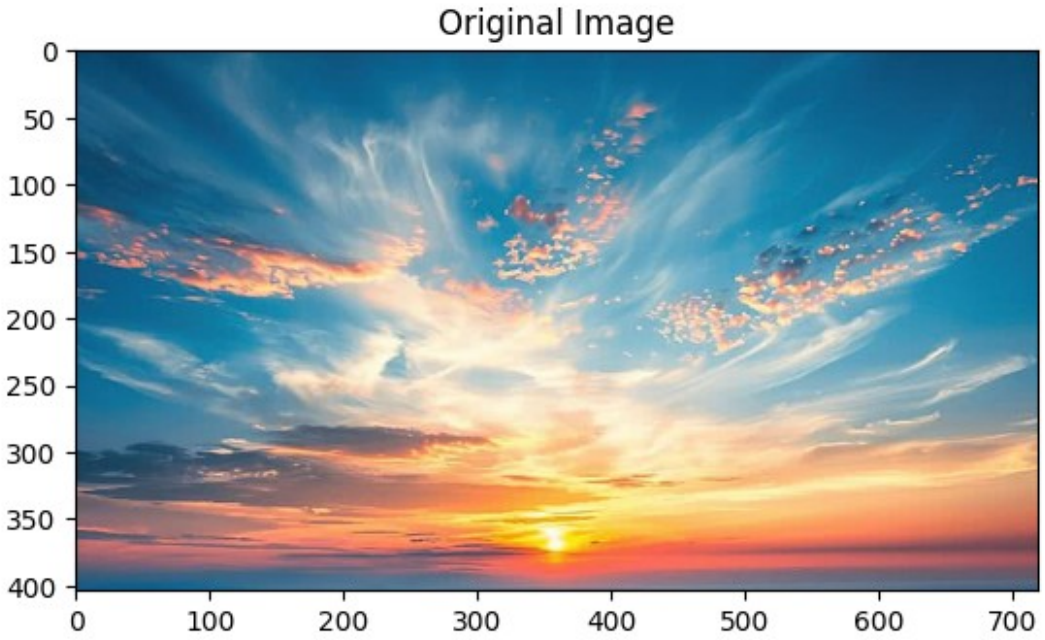
• Load the image using matplotlib.

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

# Step 1: Load an Image
image = plt.imread('img.jpg')
plt.imshow(image)
plt.title('Original Image')
plt.show()
```



• Print the shape of the RGB image

```
print("Shape of RGB Image:", image.shape)
```

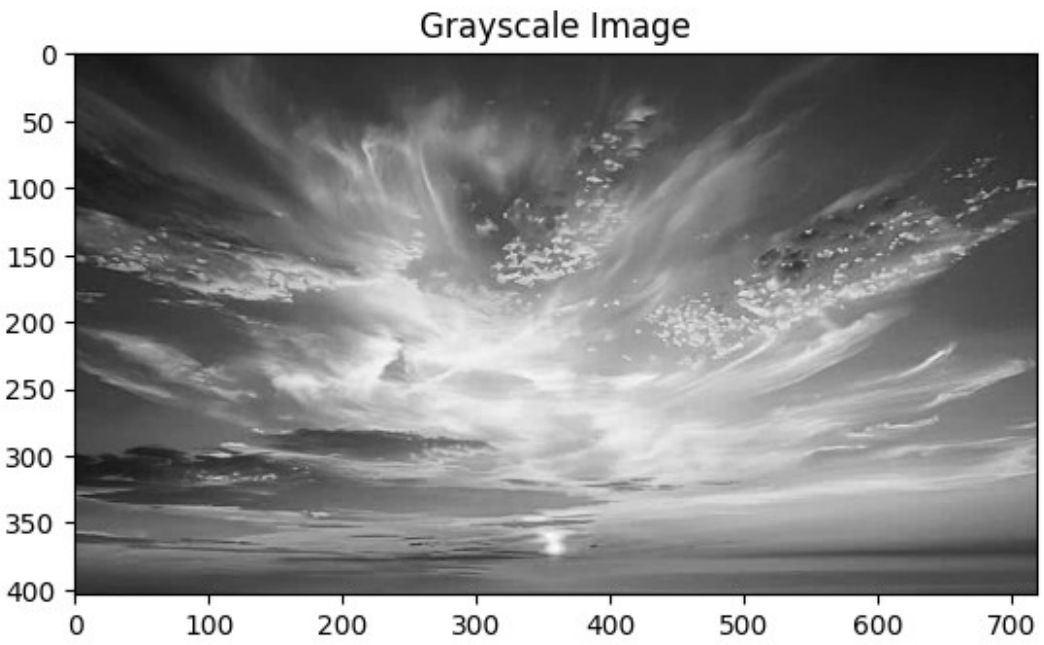
```
Shape of RGB Image: (404, 720, 3)
```

• Print the shape of the grayscale image

```
gray_image = np.mean(image, axis=2).astype(np.uint8)
print("Shape of Grayscale Image:", gray_image.shape)
```

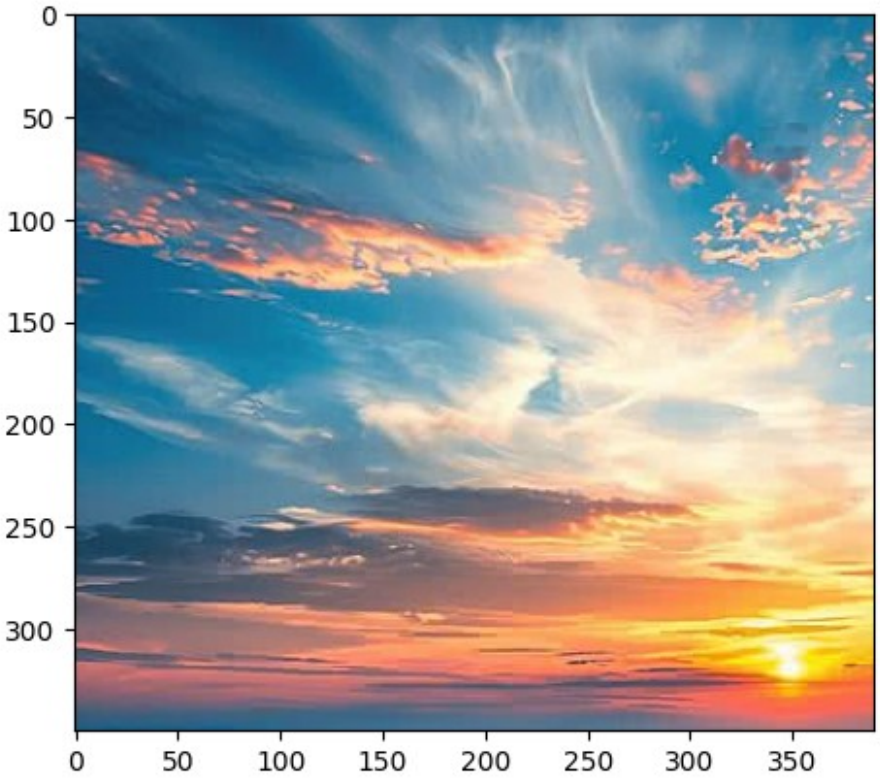
```
Shape of Grayscale Image: (404, 720)
```

```
plt.imshow(gray_image, cmap='gray')
plt.title('Grayscale Image')
plt.show()
```



- Slicing

```
plt.imshow(image[50:400:,10:400,:])
plt.show()
```



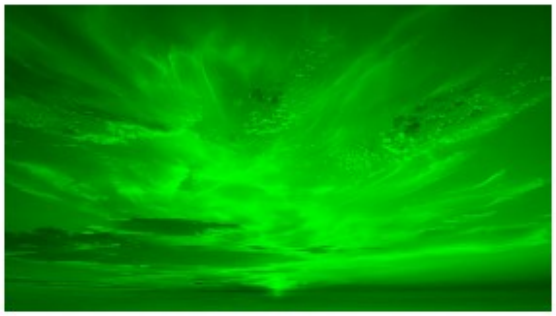
- RGB Channels

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

# Assume 'image' is already loaded as an RGB image
red_channel = image[:,:,0]
green_channel = image[:,:,1]
blue_channel = image[:,:,2]

# Create color versions for visualization
red_image = np.zeros_like(image)
red_image[:,:,0] = red_channel

green_image = np.zeros_like(image)
green_image[:,:,1] = green_channel

blue_image = np.zeros_like(image)
blue_image[:,:,2] = blue_channel

output = [image, red_image, green_image, blue_image]
titles = ['Original Image', 'Red Channel', 'Green Channel', 'Blue
Channel']

plt.figure(figsize=(8,8))
for i in range(4):
    plt.subplot(2,2,i+1)
    plt.axis('off')
    plt.title(titles[i])
    plt.imshow(output[i])

plt.show()
```

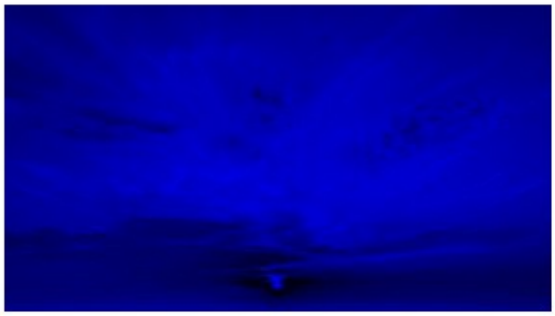
Original Image

Red Channel

Green Channel



Blue Channel



– Stacking them together

```
output = np.dstack((red_channel, green_channel, blue_channel))
plt.imshow(output)
plt.show()
```

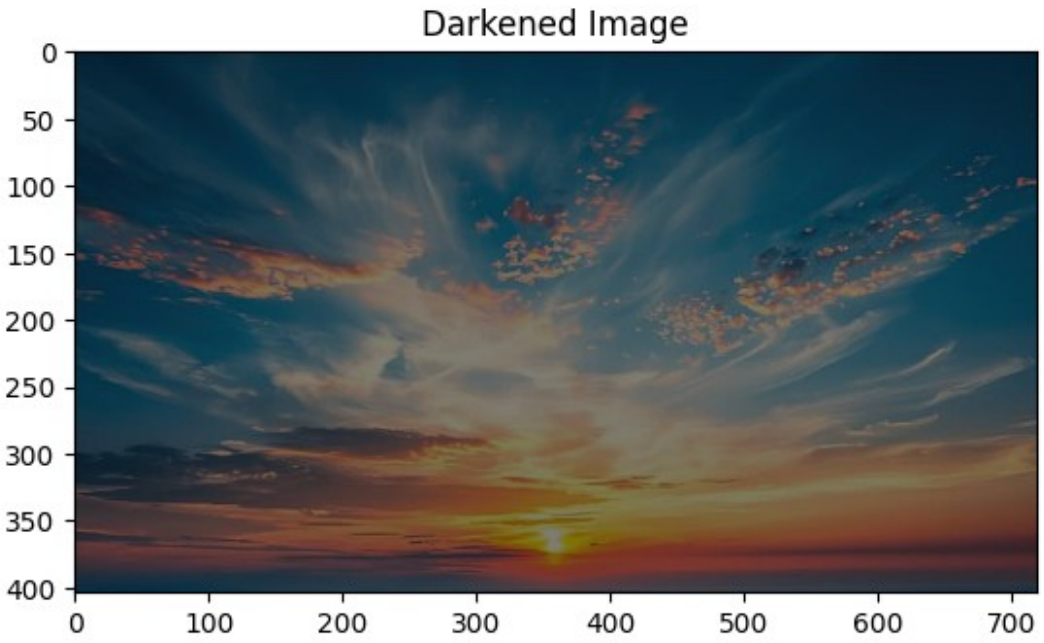
- Motion Difference

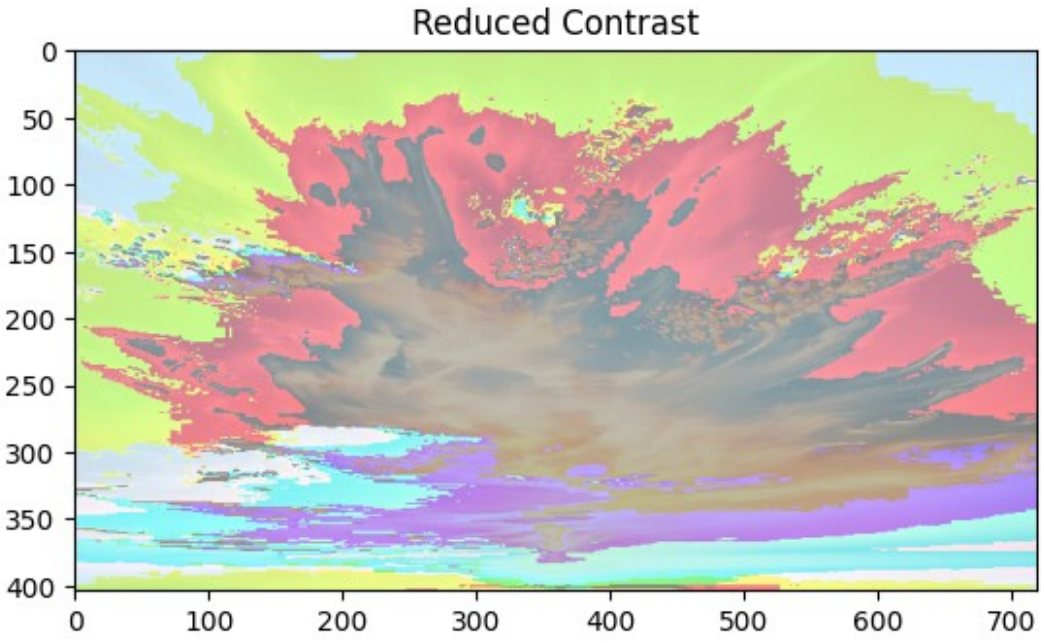
```
# Load both images in color
t4 = np.array(Image.open("tp4.jpg").convert('RGB'))
t5 = np.array(Image.open("tp5.jpg").convert('RGB'))

# Resize to same size
t5_resized = np.array(Image.fromarray(t5).resize((t4.shape[1],
t4.shape[0])))

# Compute motion difference in color
motion_diff = np.abs(t5_resized.astype(int) -
t4.astype(int)).astype(np.uint8)

# Display results
plt.figure(figsize=(12,4))
plt.subplot(1,3,1)
plt.imshow(t4)
plt.title("t4")
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(t5_resized)
plt.title("t5 (resized)")
plt.axis('off')

plt.subplot(1,3,3)
plt.imshow(motion_diff)
plt.title("Motion Difference (Color)")
plt.axis('off')
```

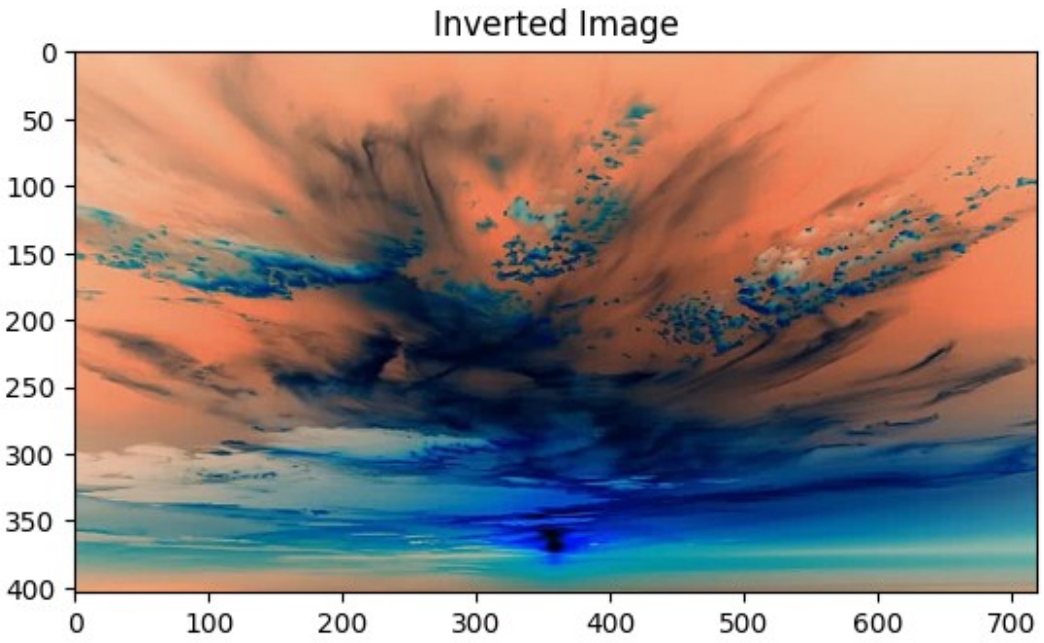
```
plt.show()
```

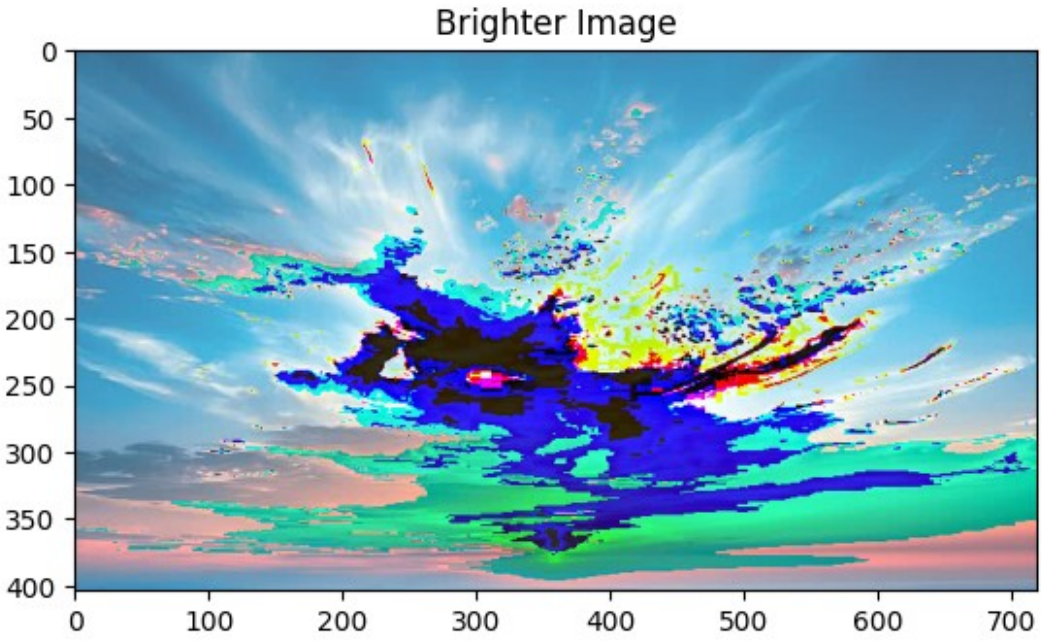
• Apply various point processing techniques to the loaded image.

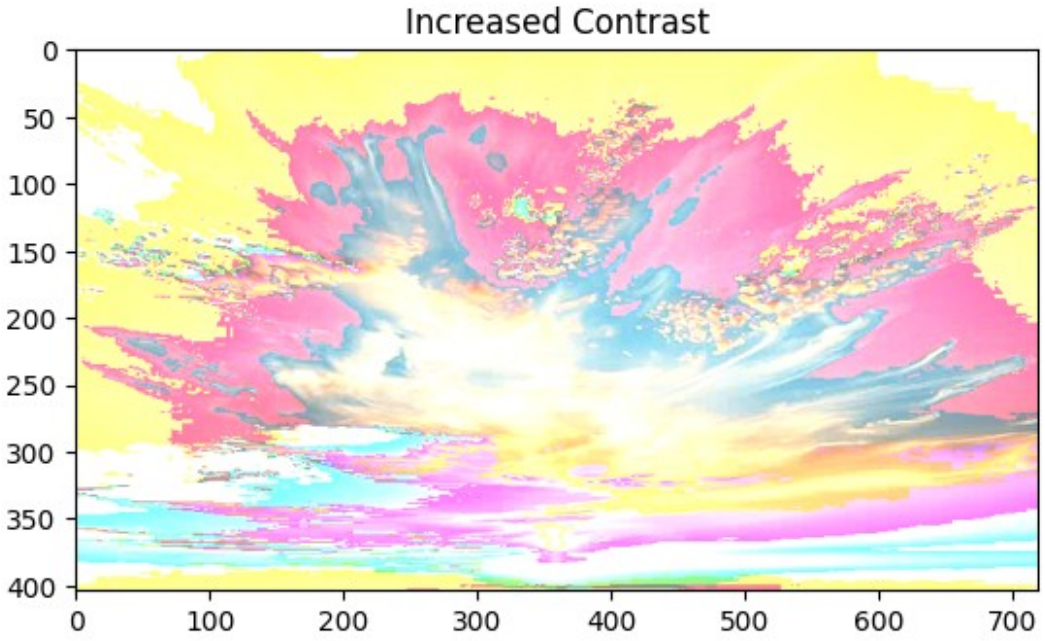
```
# a) Darken Image
dark_image = (image * 0.5).astype(np.uint8)
plt.imshow(dark_image)
plt.title('Darkened Image')
plt.show()

# b) Reduce Contrast
reduced_contrast = ((image - 128) * 0.5 + 128).astype(np.uint8)
plt.imshow(reduced_contrast)
plt.title('Reduced Contrast')
plt.show()

# c) Invert Image
inverted_image = 255 - image
plt.imshow(inverted_image)
plt.title('Inverted Image')
plt.show()

# d) Increase Brightness
bright_image = np.clip(image + 50, 0, 255).astype(np.uint8)
plt.imshow(bright_image)
plt.title('Brighter Image')
plt.show()

# e) Increase Contrast
increased_contrast = np.clip((image - 128) * 1.5 + 128, 0,
255).astype(np.uint8)
plt.imshow(increased_contrast)
plt.title('Increased Contrast')
plt.show()
```


Darkened Image
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700


Reduced Contrast
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700


Inverted Image
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700


Brighter Image
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700


Increased Contrast
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700