```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from PIL import Image
4 from skimage import exposure
```

```
1 !wget -O 'input.jpg' 'https://upload.wikimedia.org/wikipedia/commons/9/95/SunLou2.jpg'
```

```
--2023-08-13 18:21:56--  https://upload.wikimedia.org/wikipedia/commons/9/95/SunLou2.jpg
Resolving upload.wikimedia.org (upload.wikimedia.org)... 208.80.154.240, 2620:0:861:ed1a::2:b
Connecting to upload.wikimedia.org (upload.wikimedia.org)|208.80.154.240|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 88095 (86K) [image/jpeg]
Saving to: ‘input.jpg’

input.jpg           100%[===================>]  86.03K  --.-KB/s    in 0.02s

2023-08-13 18:21:56 (3.58 MB/s) - ‘input.jpg’ saved [88095/88095]
```

```
1 image_path = 'input.jpg'
2 input_image = np.array(Image.open(image_path))
3 # input_image.shape
```

```
1 def save_img(np_img, name):
2   pillow_img = Image.fromarray(np.uint8(np_img)).convert('RGB')
3   pillow_img.save('/content/'+name+'.png')
4   display(pillow_img)
```

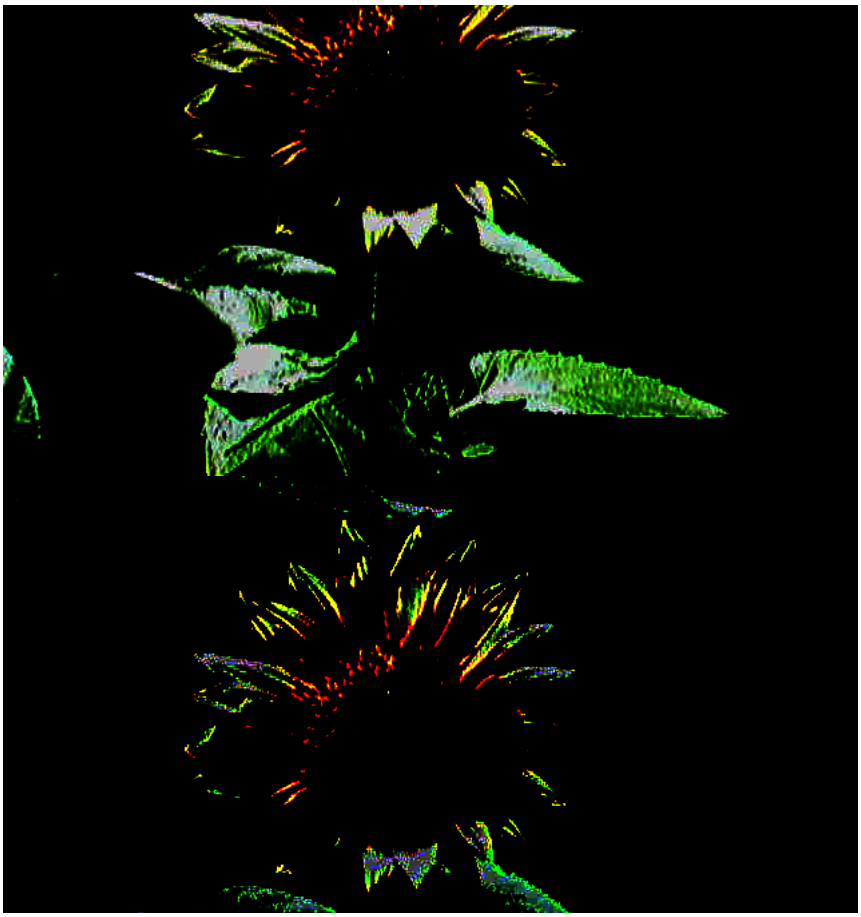
```
1 save_img(input_image, 'original')
```

```
1 negative_image = 255 - input_image
2 save_img(negative_image, 'negative')
```

```
c = 255/(np.log(1+255))
log_transformed_image = c*np.log1p(input_image)
save_img(log_transformed_image, 'log_transformed_image')
antilog_transformed_image = np.exp(input_image/c)
save_img(antilog_transformed_image, 'antilog_transformed_image')
```

```
gamma_values = [0.4, 2.5, 10, 25, 100]
gamma_corrected_images = []
```

```
for gamma in gamma_values:
    img = np.power(input_image / 255.0, gamma) * 255.0
    gamma_corrected_images.append(img)
    save_img(img, 'gamma_'+str(gamma))
```

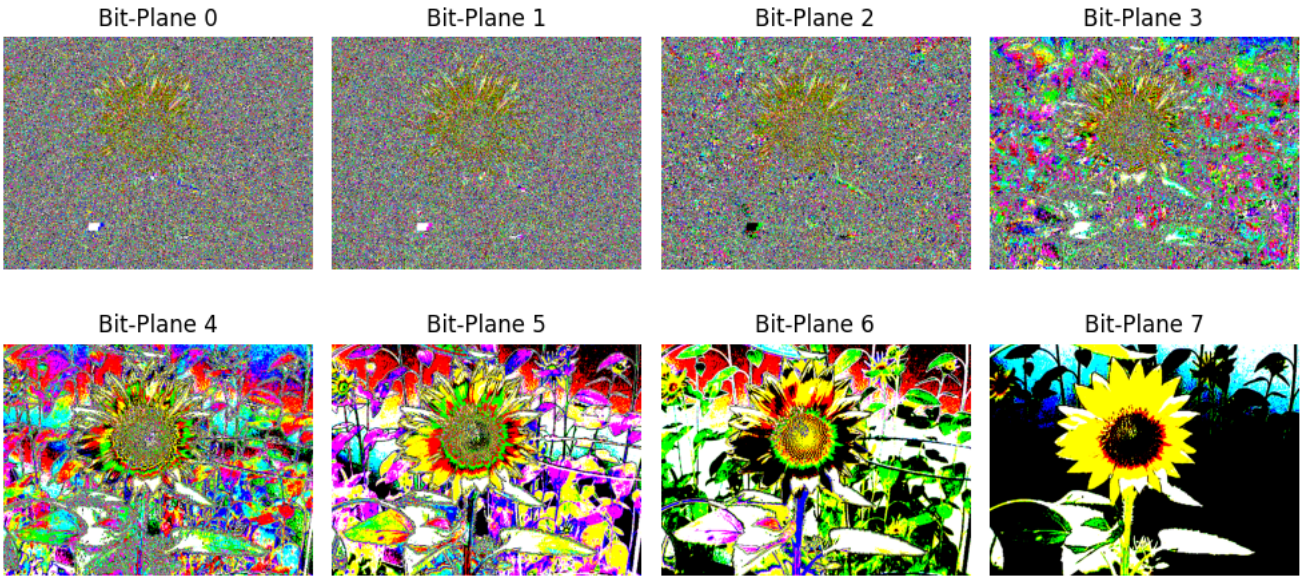


```
power_2_image = np.power(input_image, 2)
power_3_image = np.power(input_image, 3)
power_4_image = np.power(input_image, 4)
```

```
import matplotlib
matplotlib.scale.get_scale_names()
```

```
['asinh', 'function', 'functionlog', 'linear', 'log', 'logit', 'symlog']
```

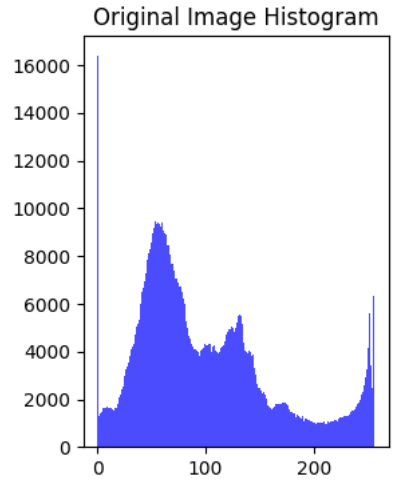
```
plt.figure(figsize=(10, 5))
for i in range(8):
    bit_plane = ((input_image >> i) & 1).astype(float)
    plt.subplot(2, 4, i + 1)
    plt.imshow(bit_plane)
    plt.title(f'Bit-Plane {i}')
    plt.axis('off')
plt.tight_layout()
```




```
plt.figure(figsize=(10, 4))
plt.subplot(1, 3, 1)
```

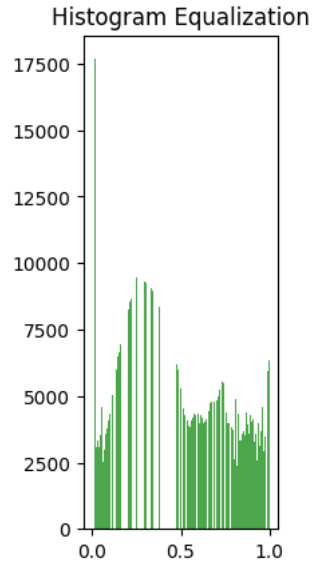
```
plt.hist(input_image.ravel(), bins=256, range=(0, 256), color='blue', alpha=0.7)
plt.title('Original Image Histogram')
```

```
Text(0.5, 1.0, 'Original Image Histogram')
```




```
histeq_image = exposure.equalize_hist(input_image)
plt.subplot(1, 3, 2)
plt.hist(histeq_image.ravel(), bins=256, range=(0, 1), color='green', alpha=0.7)
plt.title('Histogram Equalization')
```

```
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:394: UserWarning: This might be a color image. The histo
  return func(*args, **kwargs)
Text(0.5, 1.0, 'Histogram Equalization')
```




```
highlighted_range_image = input_image.copy()
# highlighted_range_image[highlighted_range_image < 0] = 0
# highlighted_range_image[highlighted_range_image > 255] = 255
highlighted_range_image[(highlighted_range_image >= 120) & (highlighted_range_image <= 200)] = 255
plt.subplot(1, 3, 3)
plt.imshow(highlighted_range_image , cmap='gray')
plt.title('Highlighted Range [120, 200]')
plt.axis('off')
plt.tight_layout()

plt.show()
```


```
# plt.hist(input_image.ravel(), bins=256, range=(-1, 256), color='blue', alpha=0.7)
```