# ***COMPUTER VISION PRACTICAL DOCUMENTATION***

DHANYA SS

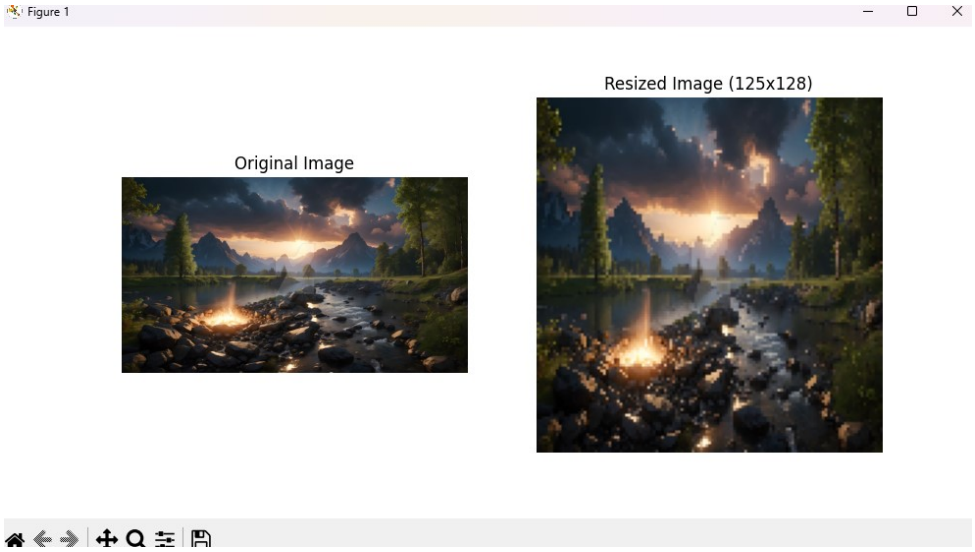
NSTI W TRIVANDRUM

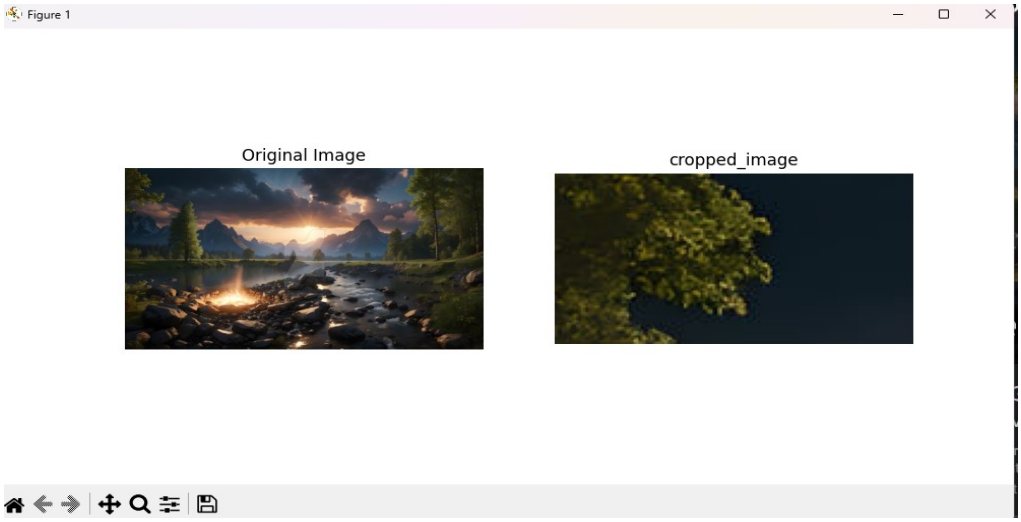
## 1. Image Resizing, Cropping, and Rotation

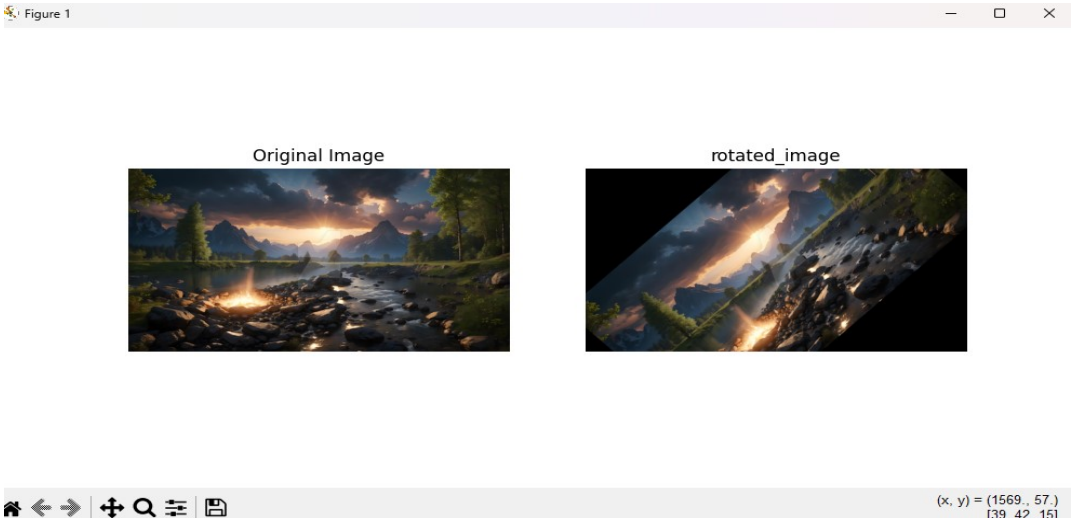
```
# Load the necessary library
import cv2
import matplotlib.pyplot as plt

# Load an image
image = cv2.imread('3.jpg')

# Convert the image from BGR (OpenCV format) to RGB (Matplotlib format)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Resize image to 256x256 pixels
resized_image = cv2.resize(image_rgb, (125, 128))

# Display the original and resized images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title('Resized Image (125x128)')
plt.imshow(resized_image)
plt.axis('off')
plt.show()

# Save or display the resized image
# cv2.imwrite('resized_image.jpg', resized_image)
```

```
# Crop image to a region (x, y, width, height)
cropped_image = image_rgb[50:130, 50:200]

# Display the original and resized images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title('cropped_image')
plt.imshow(cropped_image)
plt.axis('off')
plt.show()

# Rotate image by 45 degrees
(h, w) = image_rgb.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated_image = cv2.warpAffine(image_rgb, M, (w, h))

# Display the original and resized images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title('rotated_image')
plt.imshow(rotated_image)
plt.axis('off')
plt.show()
```

**OUTPUT:-**




**RESULT:- The program is successfully executed.**

# 2. Loading_Image_Formats

```
import cv2
import matplotlib.pyplot as plt

# Load an image using OpenCV
image_path = "3.jpg"
image_cv2 = cv2.imread(image_path)

# Convert the image from BGR to RGB
image_cv2_rgb = cv2.cvtColor(image_cv2, cv2.COLOR_BGR2RGB)

# Display the image
plt.imshow(image_cv2)
plt.title('Image loaded with OpenCV')
plt.show()


from PIL import Image

# Load an image using PIL
image_pil = Image.open(image_path)

# Display the image
plt.imshow(image_pil)
plt.title('Image loaded with PIL')
plt.show()

import imageio

# Load an image using imageio
image_imageio = imageio.imread(image_path)

# Display the image
plt.imshow(image_imageio)
plt.title('Image loaded with imageio')
```

```
plt.show()

# PNG image path
image_path_png = "4.png"
image_path_jpg = "2.jpg"

# OpenCV
image_cv2_png = cv2.imread(image_path_png)
image_cv2_png_rgb = cv2.cvtColor(image_cv2_png, cv2.COLOR_BGR2RGB)
plt.imshow(image_cv2_png_rgb)
plt.title('PNG loaded with OpenCV')
plt.show()

# PIL
image_pil_png = Image.open(image_path_png)
plt.imshow(image_cv2_png_rgb)
plt.title('PNG loaded with OpenCV')
plt.show()

# imageio
image_imageio_png = imageio.imread(image_path_png)
plt.imshow(image_cv2_png_rgb)
plt.title('PNG loaded with OpenCV')
plt.show()
```

**OUTPUT:-**

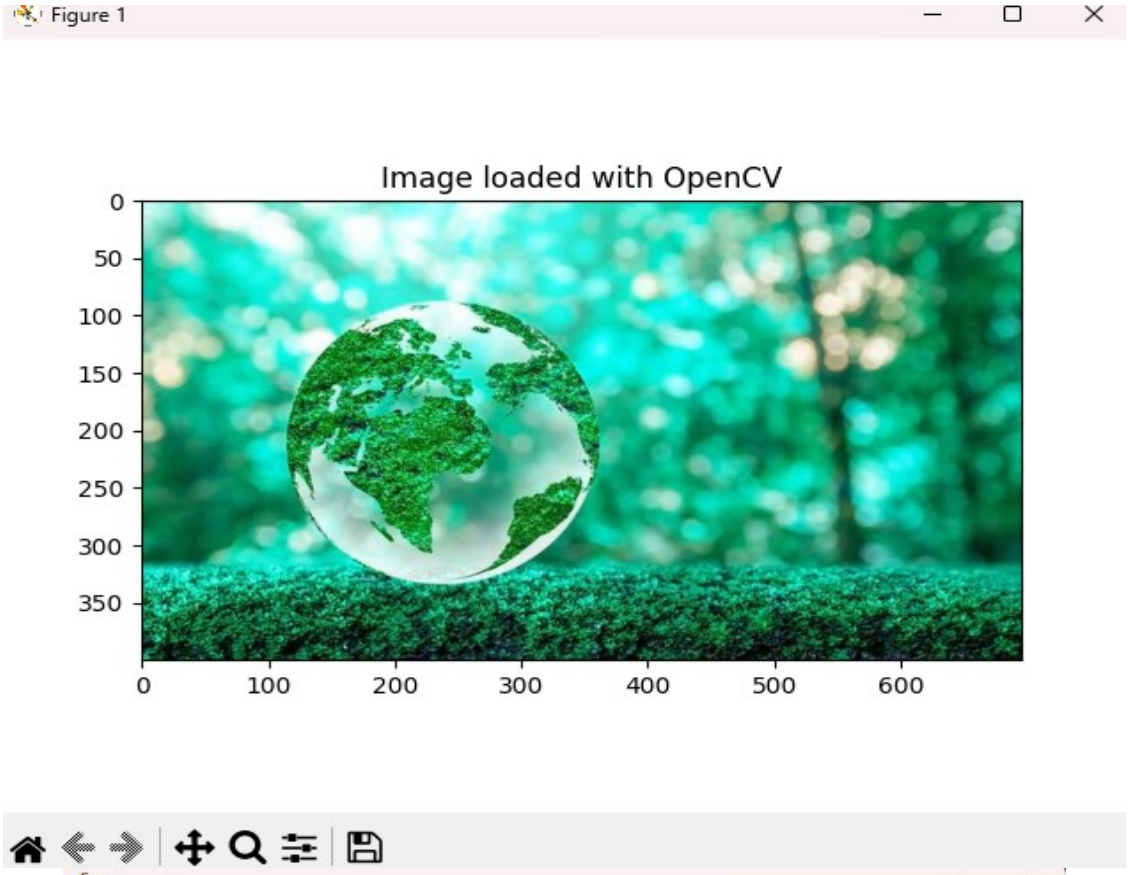


Figure 1
Image loaded with OpenCV
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600

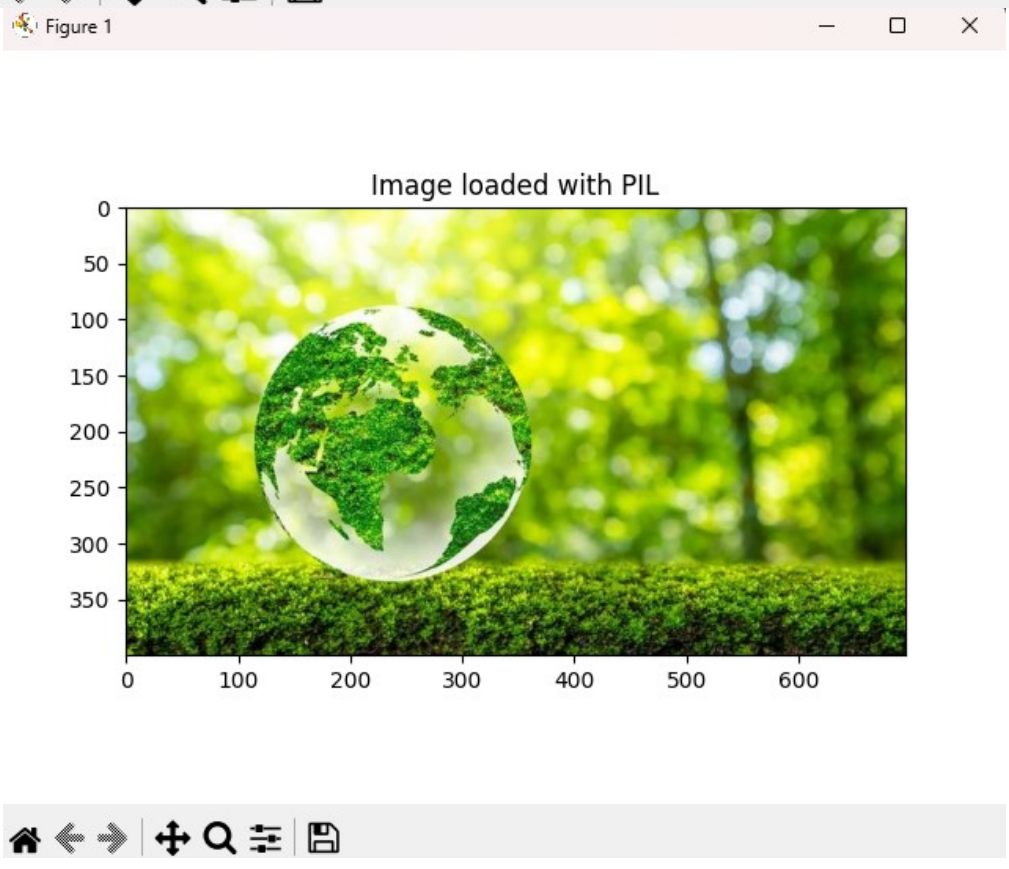


Figure 1
Image loaded with PIL
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600

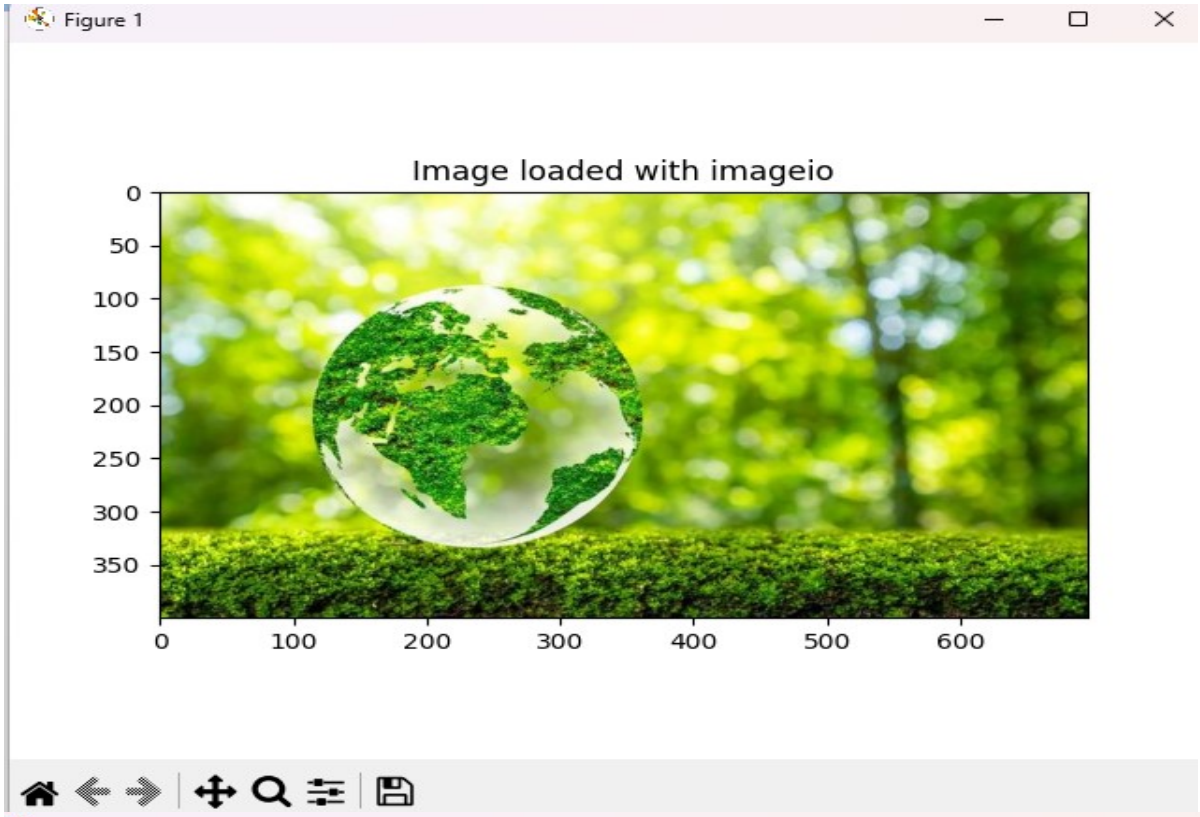


Figure 1
Image loaded with imageio
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600

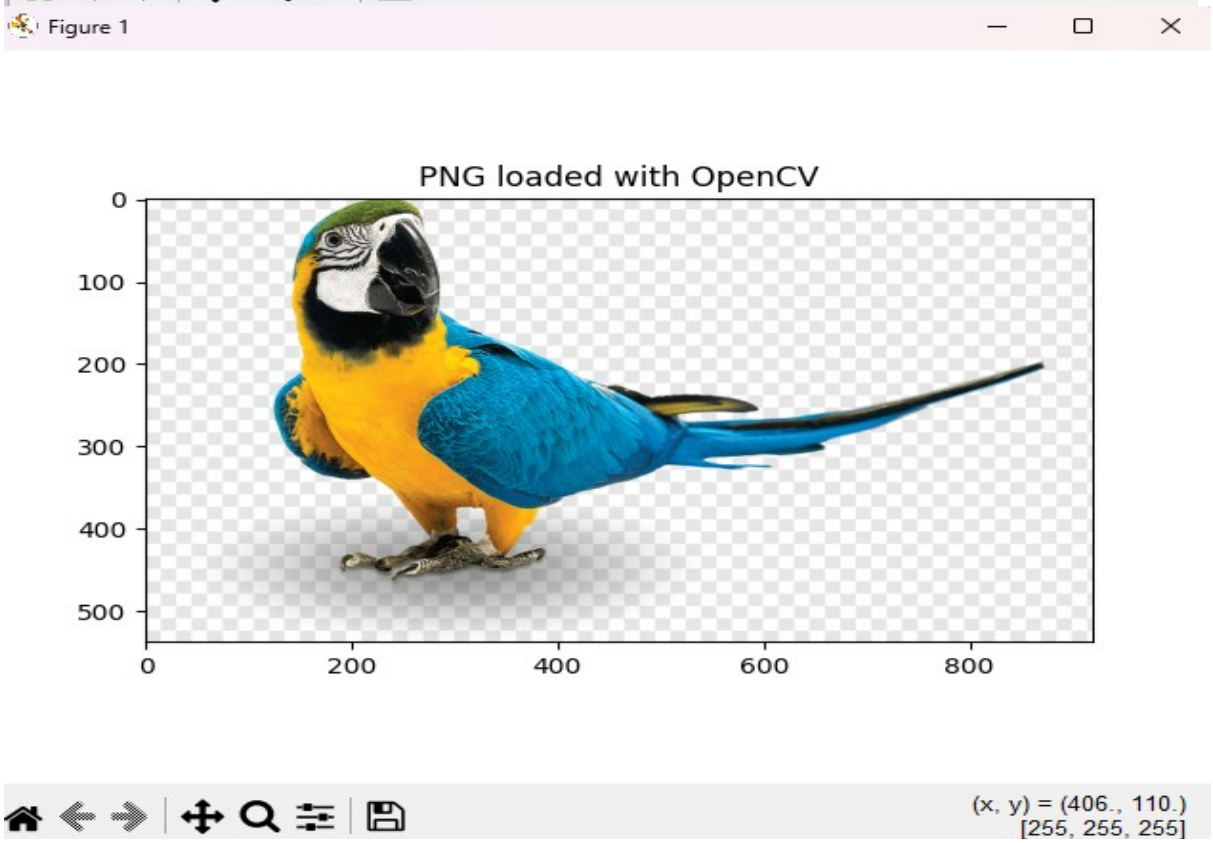
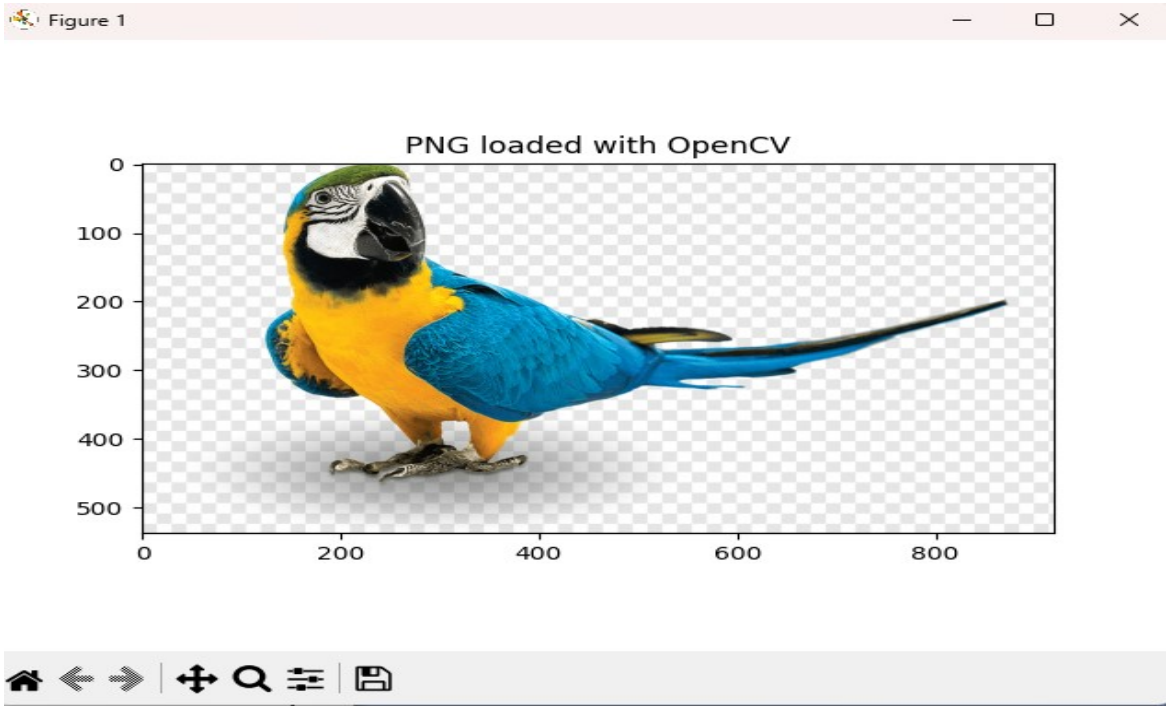
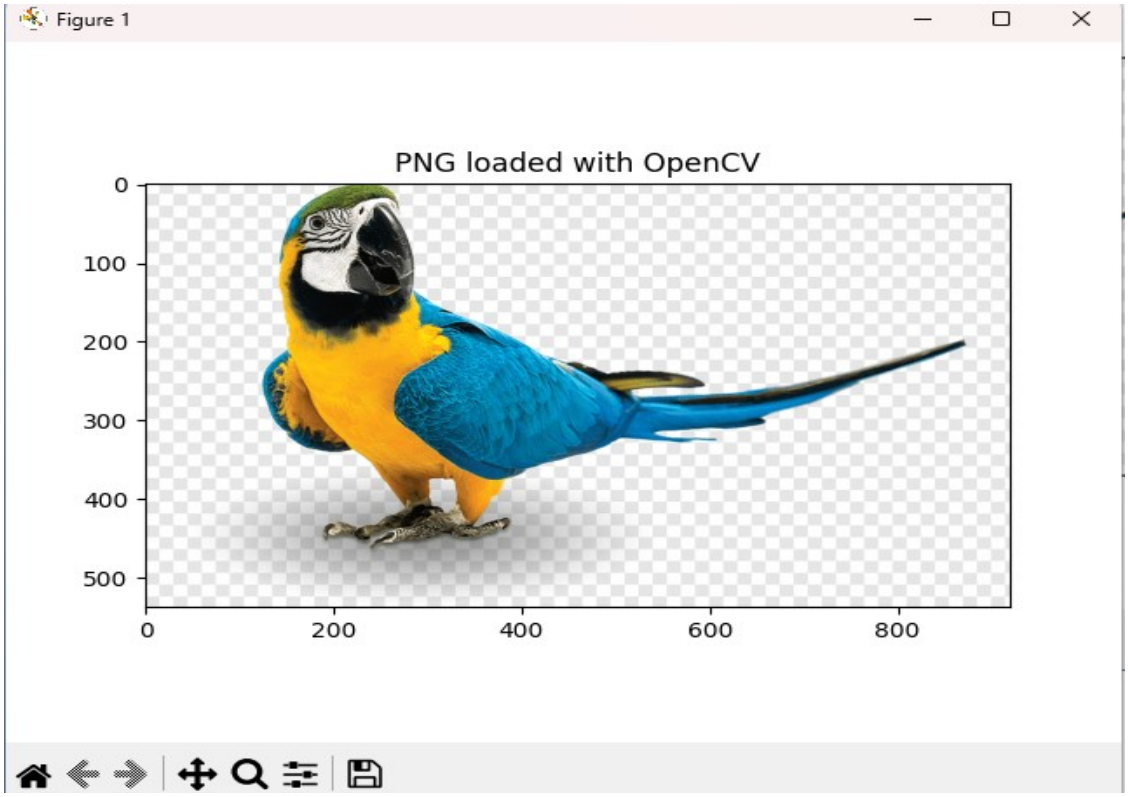


Figure 1
PNG loaded with OpenCV
0
100
200
300
400
500
0
200
400
600
800
(x, y) = (406., 110.)
[255, 255, 255]



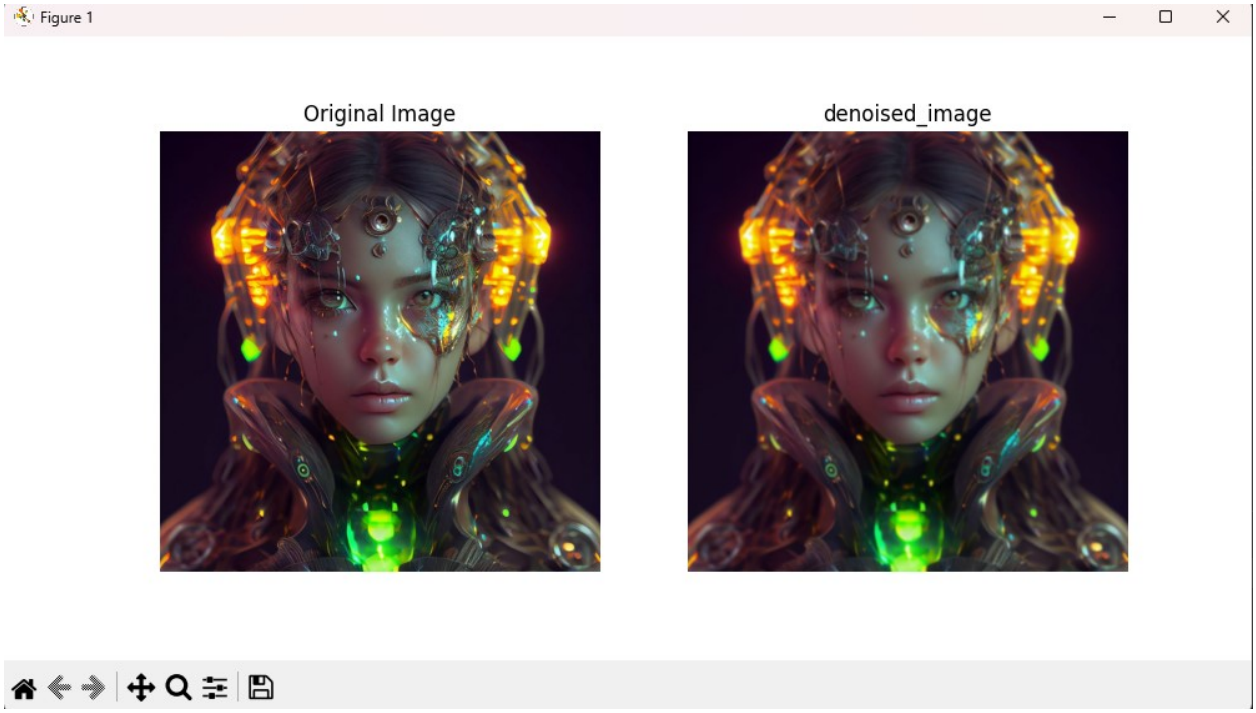

**RESULT:- The program is successfully executed.**

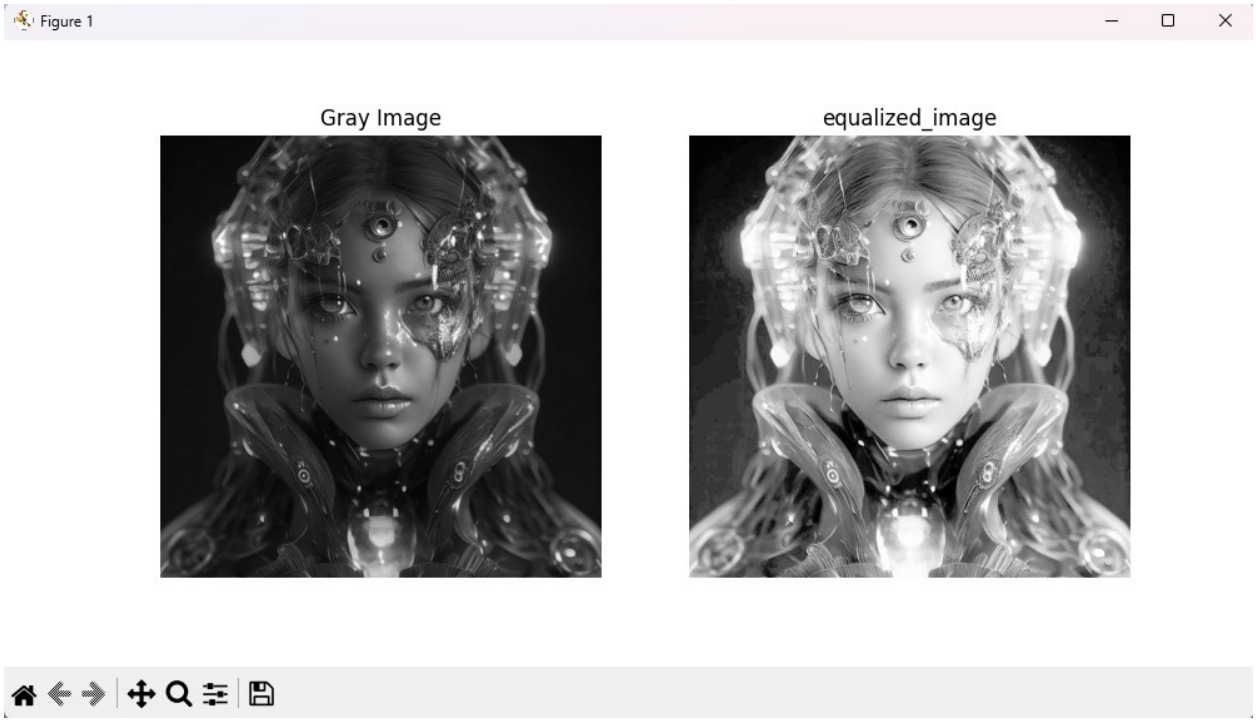
# 3. Image Denoising

```
# import necessary libraries
import cv2
import matplotlib.pyplot as plt

# Load an image
image = cv2.imread('1.jpg')

# Convert the image from BGR (OpenCV format) to RGB (Matplotlib format)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Apply Gaussian blur to denoise
denoised_image = cv2.GaussianBlur(image_rgb, (11, 11), 0)


# Display the original and resized images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title('denoised_image')
plt.imshow(denoised_image)
plt.axis('off')
plt.show()

# Convert to grayscale
gray_image = cv2.cvtColor(image_rgb, cv2.COLOR_BGR2GRAY)

# Apply histogram equalization
equalized_image = cv2.equalizeHist(gray_image)

# Display the original and resized images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
```

```
plt.title('Gray Image')
plt.imshow(gray_image, cmap="gray")
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title('equalized_image')
plt.imshow(equalized_image, cmap="gray")
plt.axis('off')
plt.show()
```

**OUTPUT:-**






**RESULT:- The program is successfully executed.**