DOCUMENTATION

SUBMITTED BY
SHUHAIB

Activity 1

Loading Different Image Formats for Computer Vision Tasks

we'll learn how to load images of various formats using different Python libraries, including OpenCV, PIL (Pillow), and imageio. These libraries provide robust methods for handling images, which are essential for computer vision tasks.

Requirements

- Personal computer/laptop
- Google Collab

Procedure

- Loading Images with OpenCV

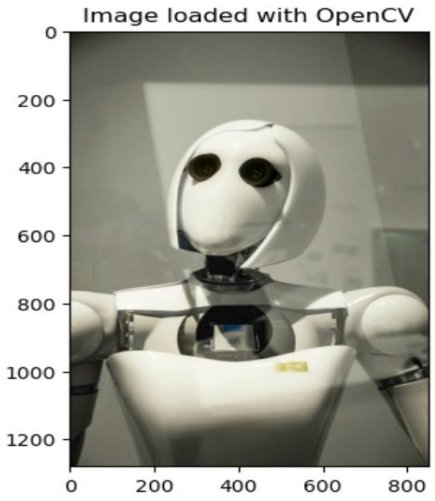
OpenCV is a powerful library for computer vision tasks. It reads images in BGR format by default.

```
import cv2
import matplotlib.pyplot as plt

# Load an image using OpenCV
image_path = "test1.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()
```



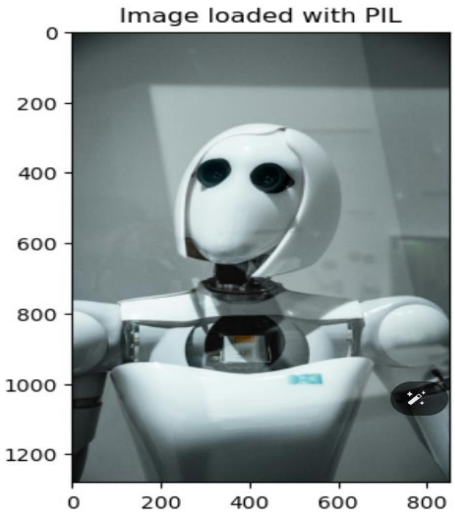
- Loading Images with PIL (Pillow)

PIL (Pillow) is a widely-used library for image processing in Python. It reads images in RGB format by default.

```
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()
```



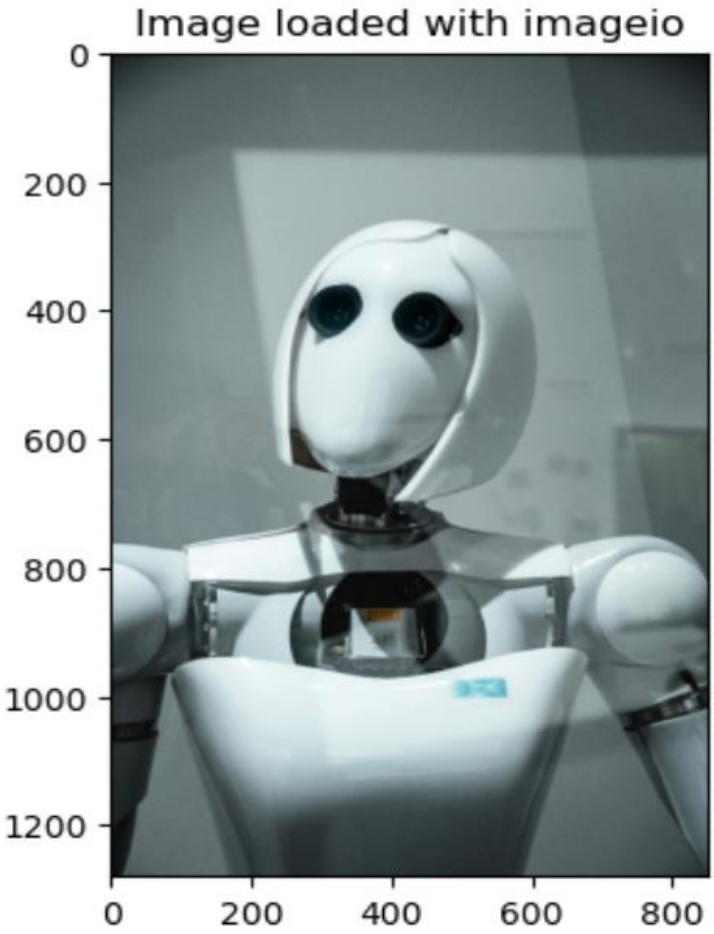
- Loading Images with imageio

imageio is another versatile library for reading and writing images in various formats.

```
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()
```



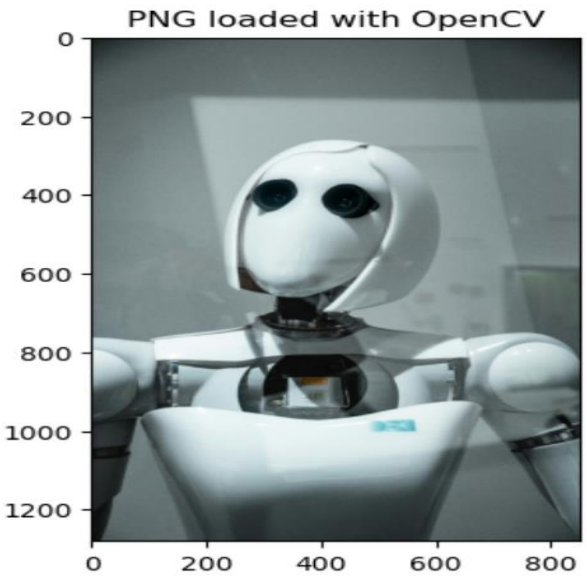
- Handling Different Image Formats

These libraries can handle various image formats such as JPEG, PNG, BMP, and more. Let's load images of different formats using each library.

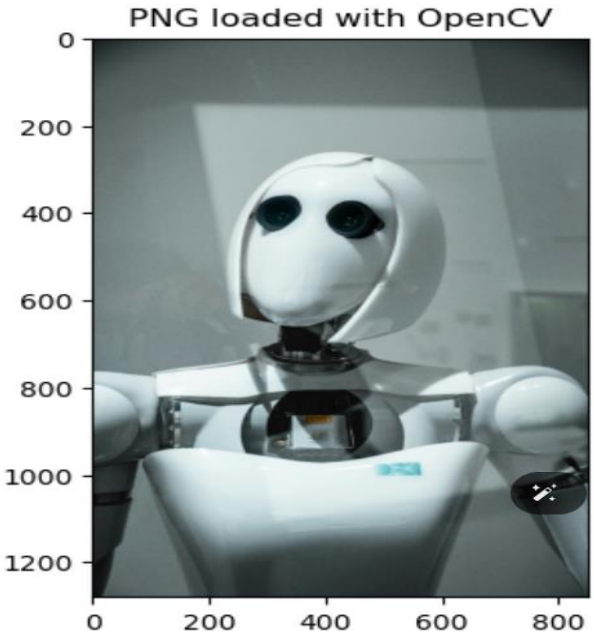
Example with PNG Image

[8]:
```
# PNG image path
image_path_png = "test1.jpg"
image_path_jpg = "test1.jpg"
```

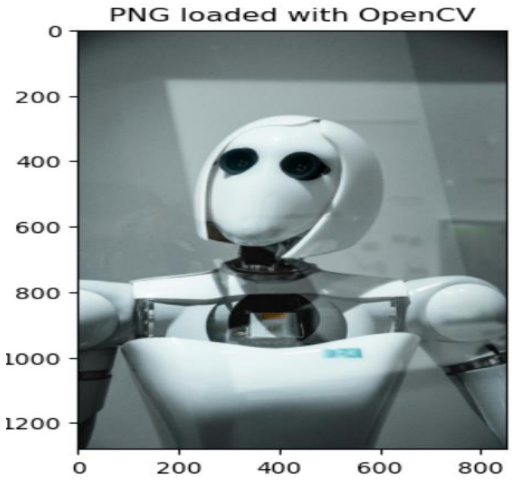
[10]:
```
# OpenCV
image_cv2_png = cv2.imread(image_path_png)
image_cv2_png_rgb = cv2.cvtColor(image_cv2_png, cv2.COLOR_BGR2RGB)
plt.imshow(image_cv2_png_rgb)
plt.title('PNG loaded with OpenCV')
plt.show()
```



[12]:
```
# PIL
image_pil_png = Image.open(image_path_png)
plt.imshow(image_cv2_png_rgb)
plt.title('PNG loaded with OpenCV')
plt.show()
```



[16]:
```
# imageio
image_imageio_png = imageio.imread(image_path_png)
plt.imshow(image_cv2_png_rgb)
plt.title('PNG loaded with OpenCV')
plt.show()
```



Activity 2

Image Preprocessing Techniques

Image Resizing, Cropping, and Rotation: Adjusts the size, shape, and orientation of images.

Requirements

- Personal computer/laptop
- Google Collab

Procedure

Load the necessary library

[2]:
```
# Load the necessary library
import cv2
import matplotlib.pyplot as plt
```

[4]:
```
# Load an image
image = cv2.imread('test2.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)
```

[7]:
```
# 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()
```

Original Image

cropped_image

[10]:
```
# 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()
```

Original Image

rotated_image

Activity 3

Image Preprocessing Techniques

- Image Denoising and Smoothing: Reduces noise and smoothens images to improve quality.
- Histogram Equalization and Contrast Enhancement: Enhances the contrast and brightness of images for better visibility.

- Image Denoising and Smoothing : These techniques reduce noise and smooth the image to enhance the quality.
- Denoising : Denoising removes unwanted noise from images.

Requirements

- Personal computer/laptop
- Google Collab

Procedure

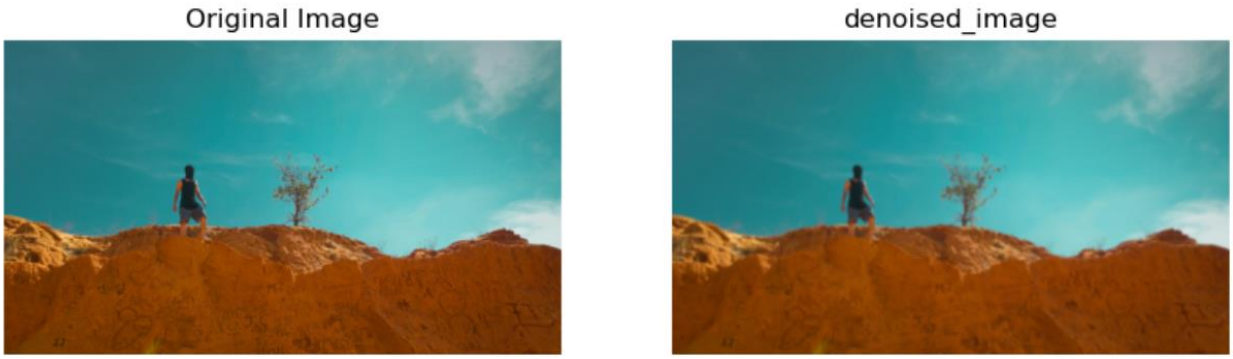
[2]:
```
# import necessary libraries
import cv2
import matplotlib.pyplot as plt
```

[4]:
```python
# Load an image
image = cv2.imread('test3.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()
```



- Histogram Equalization and Contrast Enhancement These techniques improve the contrast and brightness of images.

- Histogram Equalization
  Histogram equalization enhances the contrast of an image by spreading out the most frequent intensity values.

[6]:
```python
# 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()
```