Lab Documentation(Computer Vision)

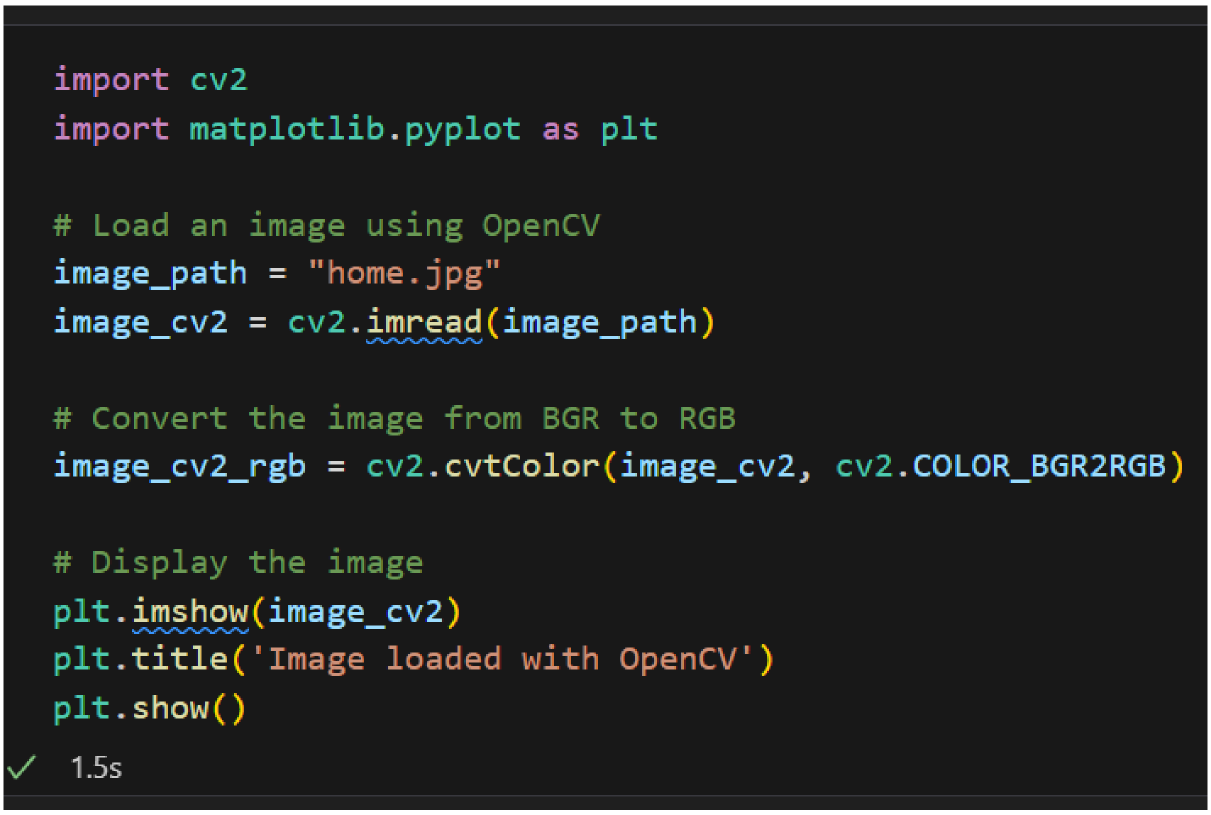
Akash T
NSTI | CALICUT

<u>Lab -1</u>

Loading Different Image Formats for Computer Vision Tasks

<u>Procedure</u>

<u>Loading Images with OpenCV</u>

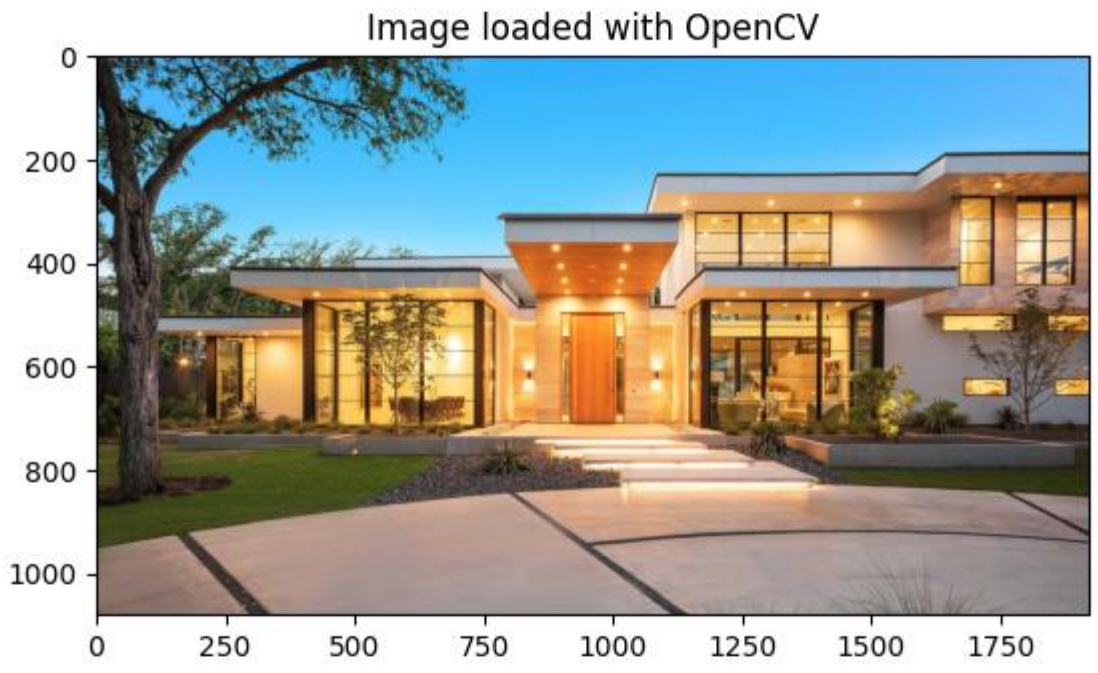


```python
import cv2
import matplotlib.pyplot as plt

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

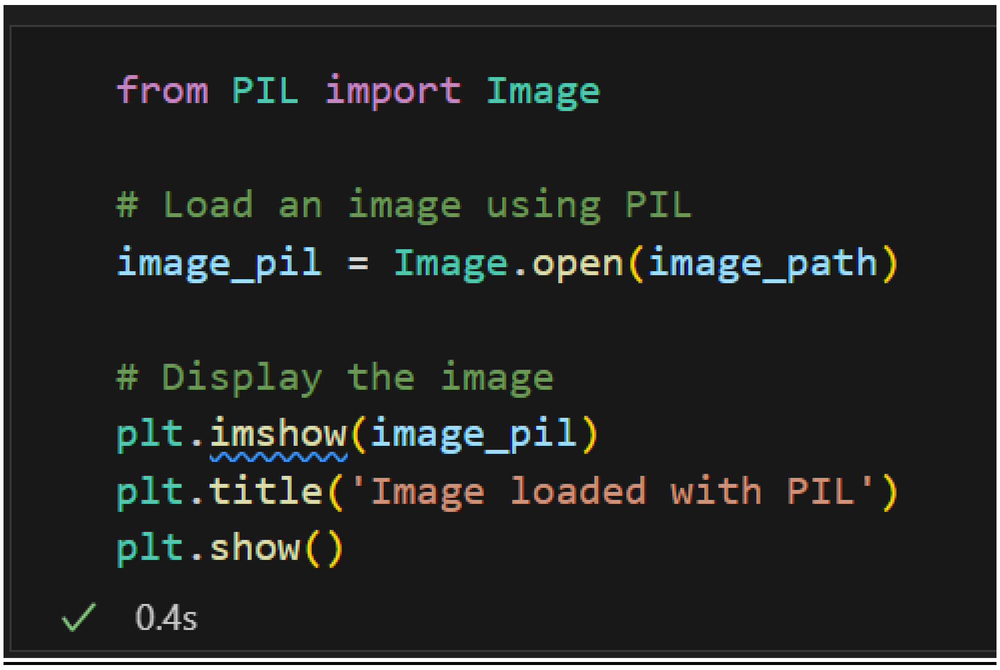
```
✓ 1.5s
```

- cv2: This is the OpenCV library, which is used for image processing tasks.
- matplotlib.pyplot: This is a plotting library used for creating static, animated, and interactive visualizations in Python.
- image_path: A string variable holding the path to the image file.
- cv2.imread(image_path): Reads the image from the specified path and returns it as a NumPy array in BGR (Blue, Green, Red) format.
- cv2.cvtColor(image_cv2, cv2.COLOR_BGR2RGB): Converts the image from BGR color space (used by OpenCV) to RGB color space (used by Matplotlib). This step is needed because Matplotlib expects images in RGB format for accurate color representation.
- plt.imshow(image_cv2): Displays the image using Matplotlib.
- plt.title('Image loaded with OpenCV'): Sets the title of the image plot.

- plt.show(): Renders the image on the screen.



Loading Images with PIL (Pillow)



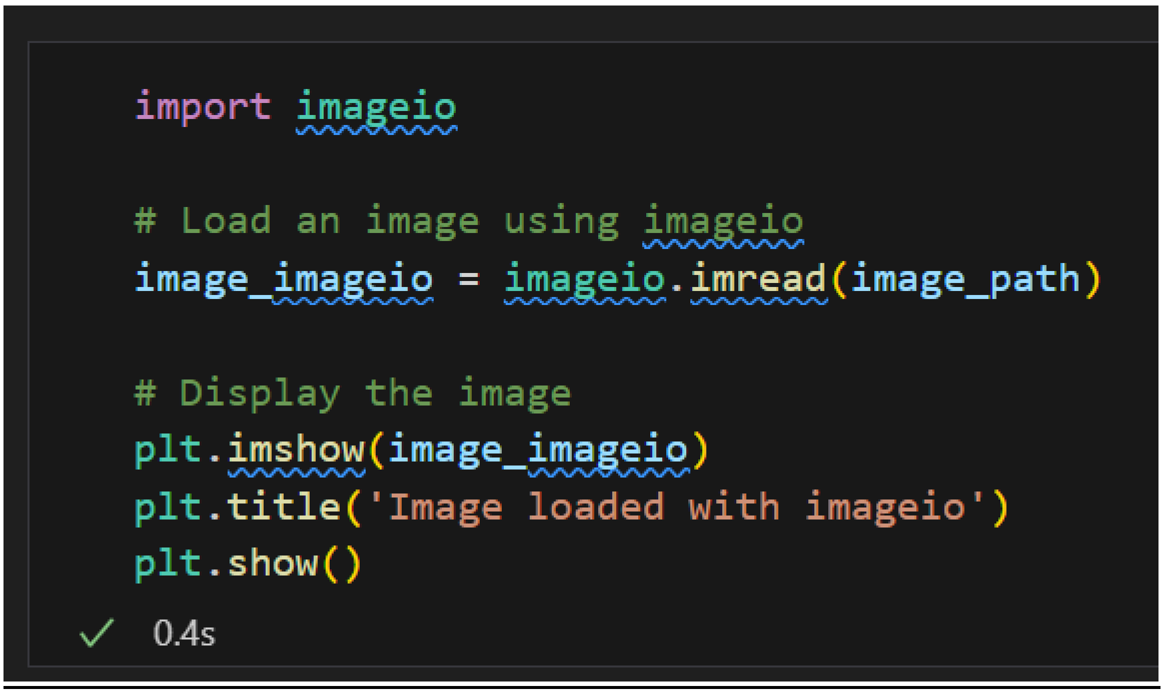
```
from PIL import Image

# 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()
```

✓ 0.4s

- Image.open(image_path): Opens the image file specified by image_path and returns a PIL Image object. The Image object supports various image operations and maintains color space information correctly.

- plt.imshow(image_pil): Displays the image using Matplotlib. PIL images are automatically converted to a format that Matplotlib can handle.
- plt.title('Image loaded with PIL'): Sets the title of the image plot.
- plt.show(): Renders and displays the image on the screen.

Loading Images with imageio



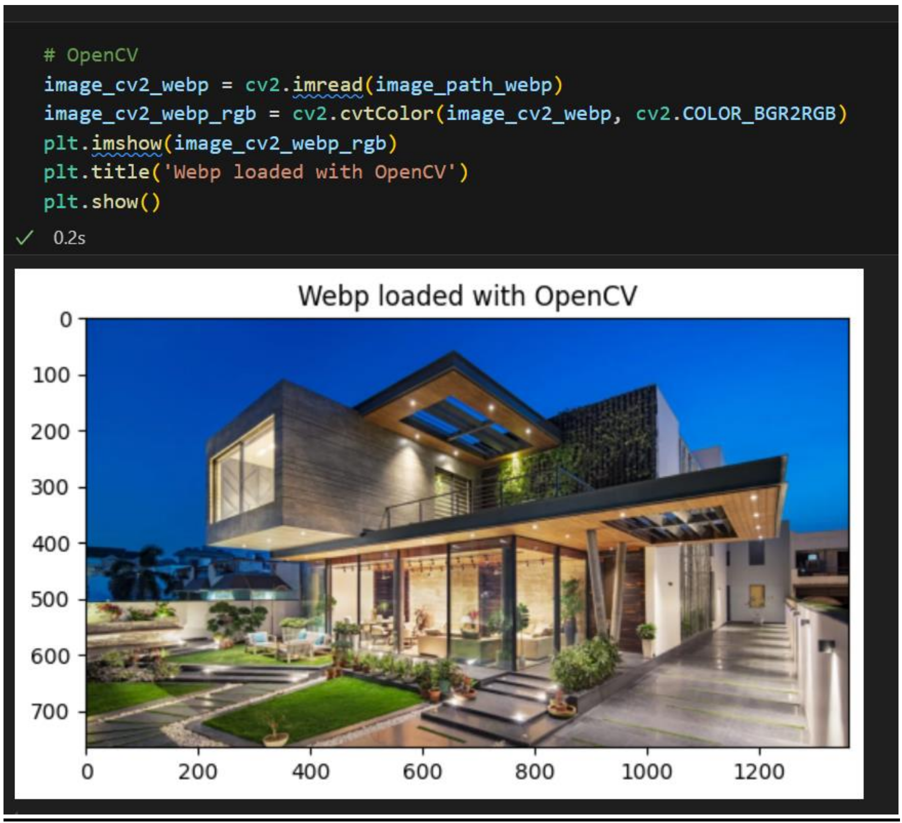
```
import imageio

# 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()
```

✓ 0.4s

- imageio: A library for reading and writing image data. It supports various image formats and is straightforward to use for loading images.
- matplotlib.pyplot: Used for plotting and displaying images.
- imageio.imread(image_path): Reads the image from the specified path and returns it as a NumPy array. imageio handles color space internally, so no explicit conversion is needed for displaying the image.
- plt.imshow(image_imageio): Displays the image using Matplotlib. Since imageio loads images in a format compatible with Matplotlib, this function will render the image correctly.
- plt.title('Image loaded with imageio'): Sets the title of the image plot.
- plt.show(): Renders and displays the image on the screen.

Handling Different Image Formats(webp)

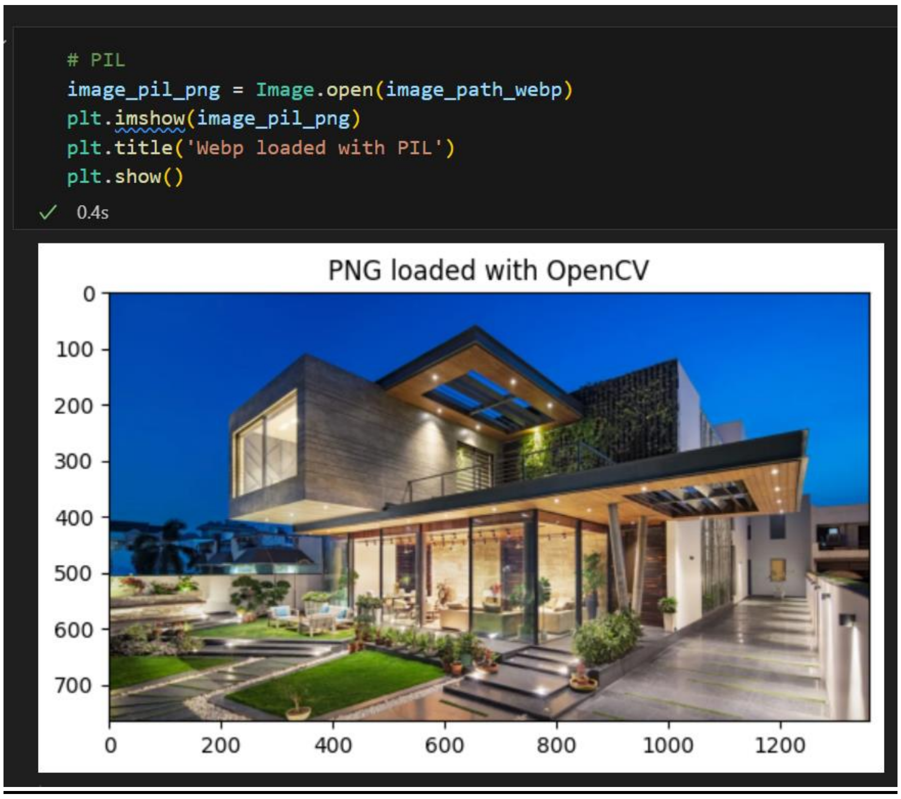
Opencv



```
# OpenCV
image_cv2_webp = cv2.imread(image_path_webp)
image_cv2_webp_rgb = cv2.cvtColor(image_cv2_webp, cv2.COLOR_BGR2RGB)
plt.imshow(image_cv2_webp_rgb)
plt.title('Webp loaded with OpenCV')
plt.show()
```

✓ 0.2s

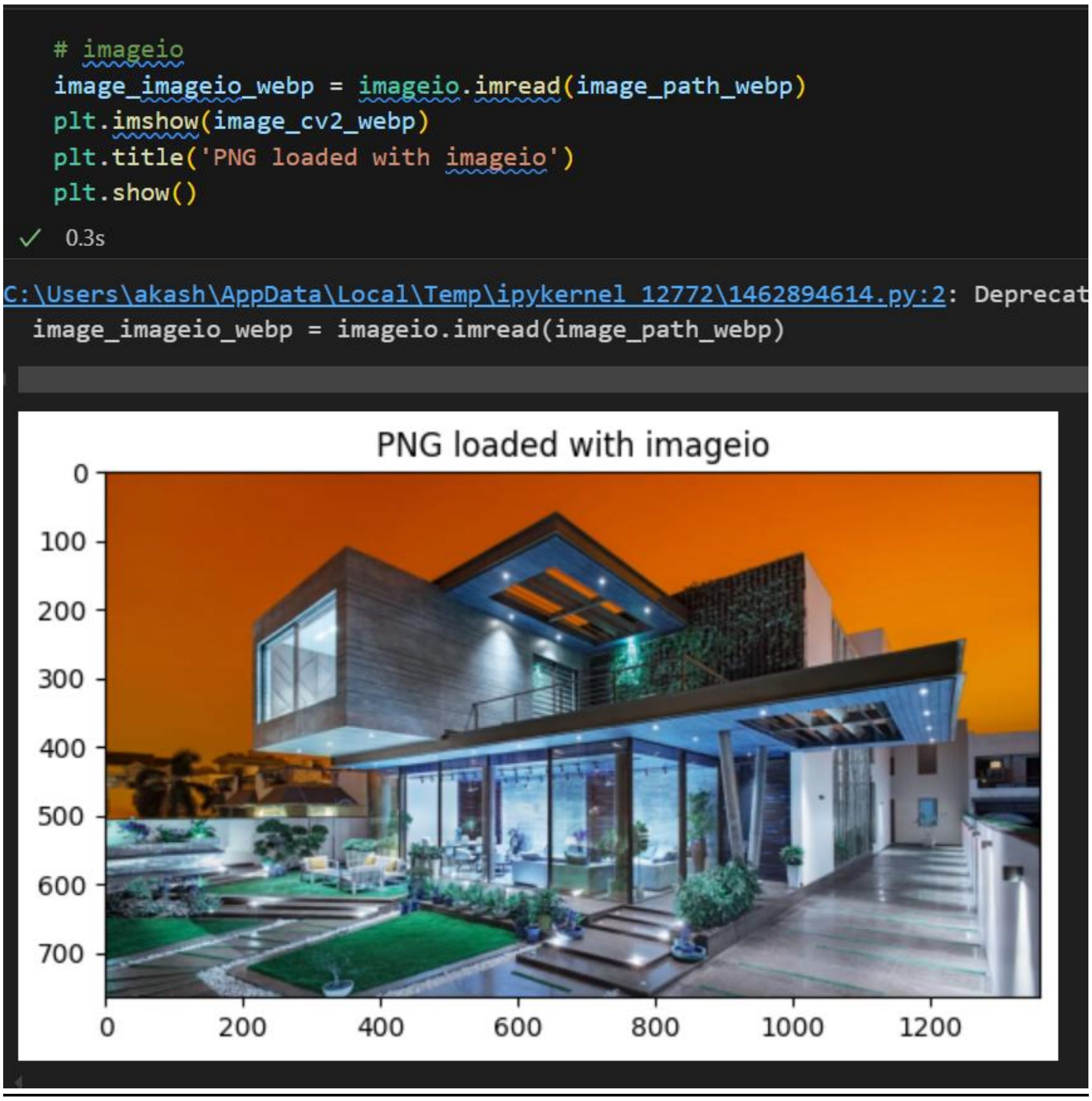
PIL



```
# PIL
image_pil_png = Image.open(image_path_webp)
plt.imshow(image_pil_png)
plt.title('Webp loaded with PIL')
plt.show()
```

✓ 0.4s

Imageio



```
# imageio
image_imageio_webp = imageio.imread(image_path_webp)
plt.imshow(image_cv2_webp)
plt.title('PNG loaded with imageio')
plt.show()
```

✓ 0.3s

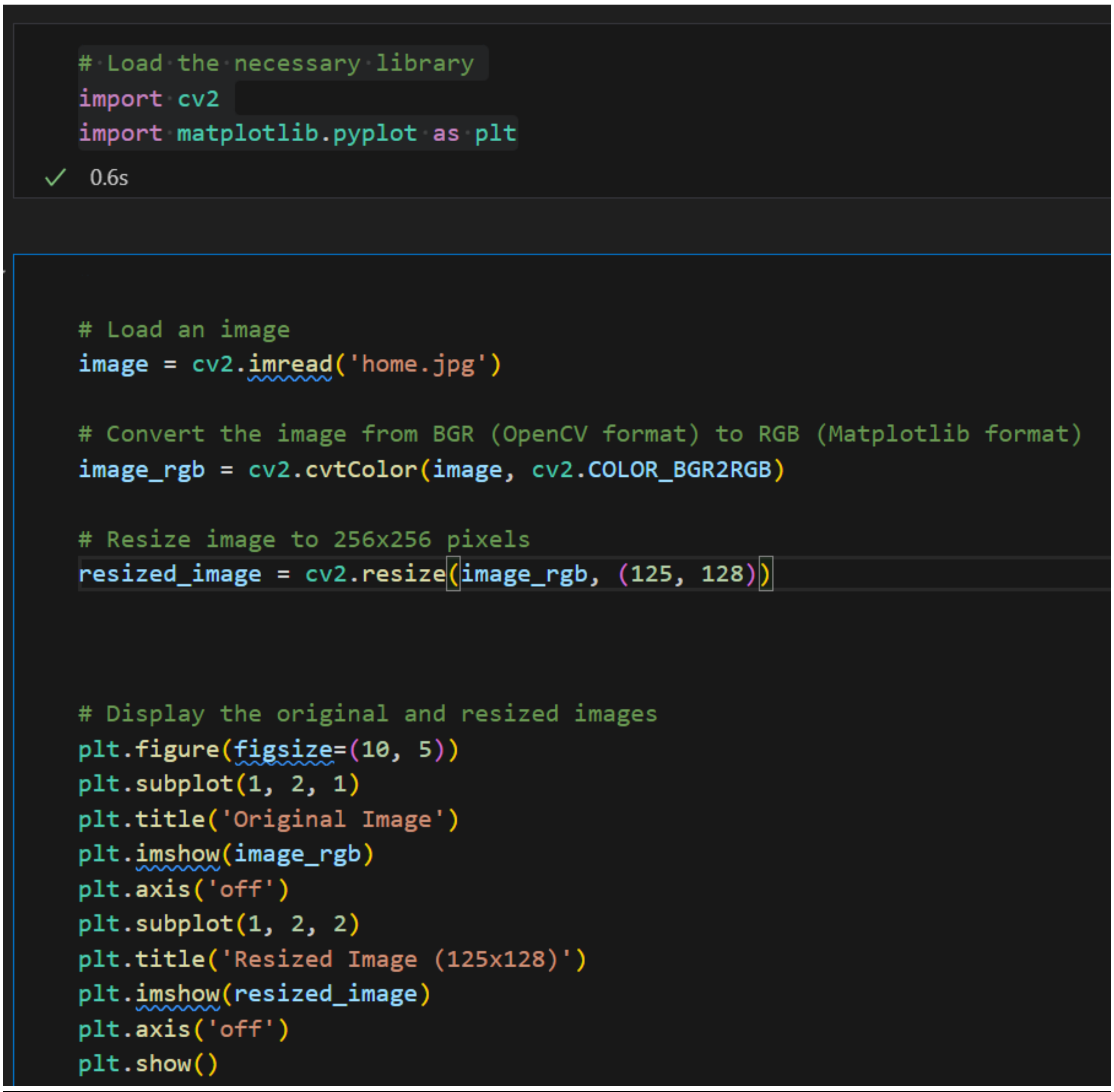
```
C:\Users\akash\AppData\Local\Temp\ipykernel_12772\1462894614.py:2: Deprecat
  image_imageio_webp = imageio.imread(image_path_webp)
```

Lab-2

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

Procedure

Image resized



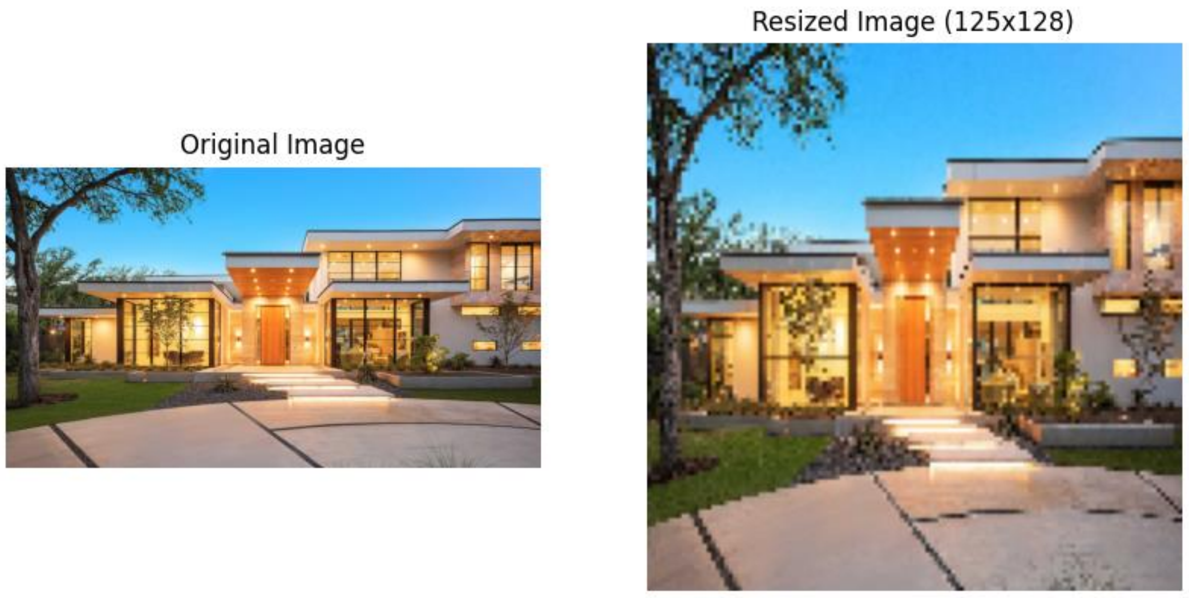
```
# Load the necessary library
import cv2
import matplotlib.pyplot as plt
```

✓ 0.6s

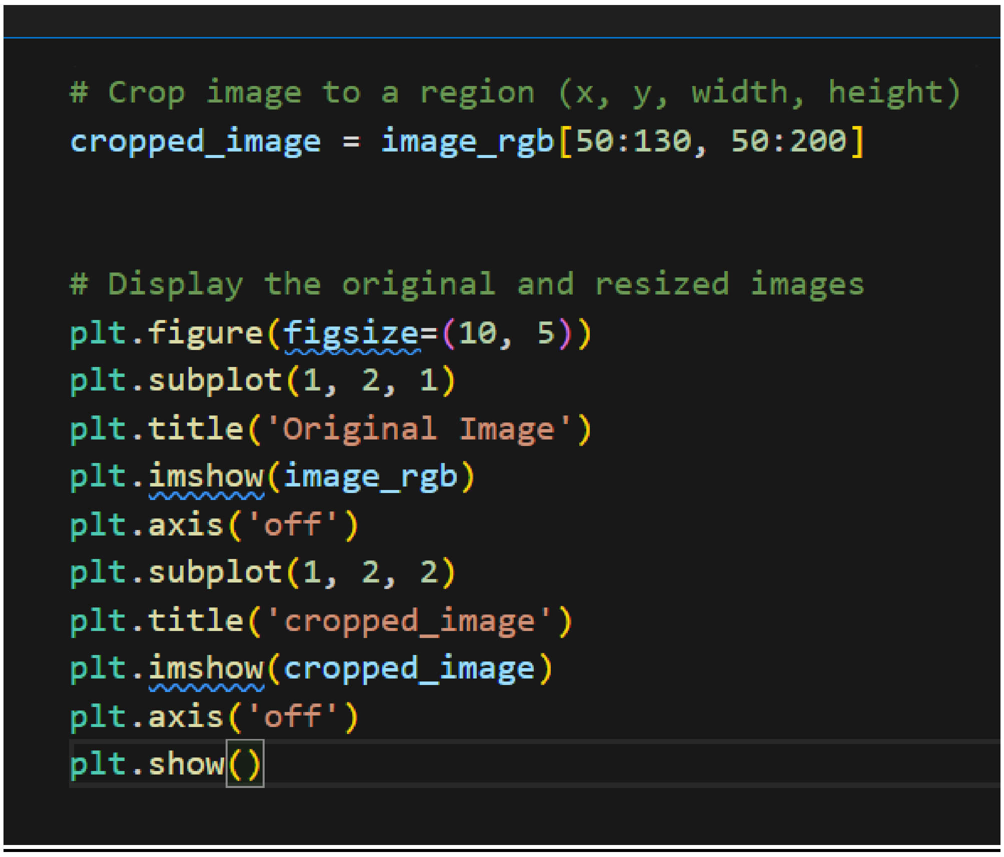
```
# Load an image
image = cv2.imread('home.jpg')

# 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()
```

- cv2: The OpenCV library, used for image processing tasks.
- matplotlib.pyplot: Used for plotting and displaying images.
- cv2.imread('home.jpg'): Reads the image file specified by 'home.jpg' and loads it into a NumPy array. The image is in BGR format by default.

- cv2.cvtColor(image, cv2.COLOR_BGR2RGB): Converts the image from BGR (used by OpenCV) to RGB (used by Matplotlib) to ensure correct color representation when displaying the image.
- cv2.resize(image_rgb, (125, 128)): Resizes the RGB image to 125x128 pixels. This method can resize images to any specified dimensions.
- plt.figure(figsize=(10, 5)): Creates a figure with a size of 10x5 inches.
- plt.subplot(1, 2, 1): Creates a subplot in a 1-row, 2-column grid, selecting the first subplot.
- plt.title('Original Image'): Sets the title for the original image subplot.
- plt.imshow(image_rgb): Displays the original RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.subplot(1, 2, 2): Creates the second subplot in the 1-row, 2-column grid.
- plt.title('Resized Image (125x128)'): Sets the title for the resized image subplot.
- plt.imshow(resized_image): Displays the resized RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.show(): Renders and displays the images on the screen.



Crop image

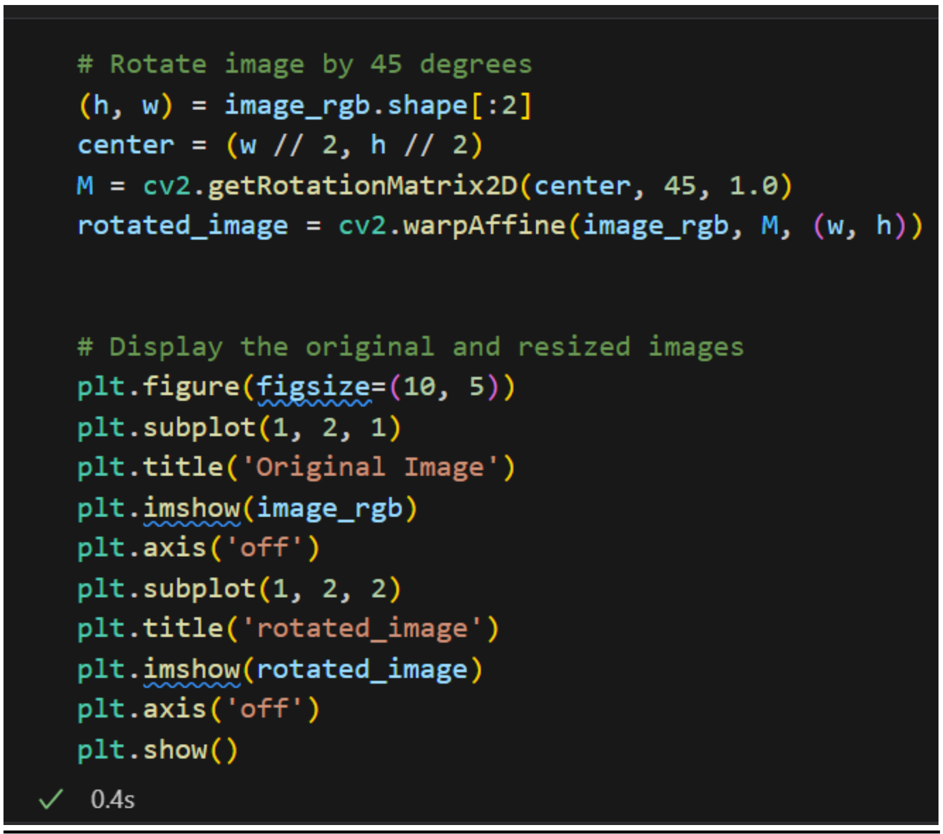


```
# Crop image to a region (x, y, width, height)
cropped_image = image_rgb[50:130, 50:200]


# 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()
```

- image_rgb[50:130, 50:200]: Crops the image by selecting a rectangular region. The slicing syntax [y1:y2, x1:x2] is used, where (y1, x1) is the top-left corner and (y2, x2) is the bottom-right corner of the cropping rectangle. In this case:
  1. 50:130 specifies the rows (height), so it crops from row 50 to row 130.
  2. 50:200 specifies the columns (width), so it crops from column 50 to column 200.
- The resulting cropped_image contains only the selected region of the original image.
- plt.figure(figsize=(10, 5)): Creates a figure with a size of 10x5 inches.
- plt.subplot(1, 2, 1): Creates a subplot in a 1-row, 2-column grid, selecting the first subplot.
- plt.title('Original Image'): Sets the title for the original image subplot.
- plt.imshow(image_rgb): Displays the original RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.subplot(1, 2, 2): Creates the second subplot in the 1-row, 2-column grid.

- plt.title('Cropped Image'): Sets the title for the cropped image subplot.
- plt.imshow(cropped_image): Displays the cropped RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.show(): Renders and displays the images on the screen.

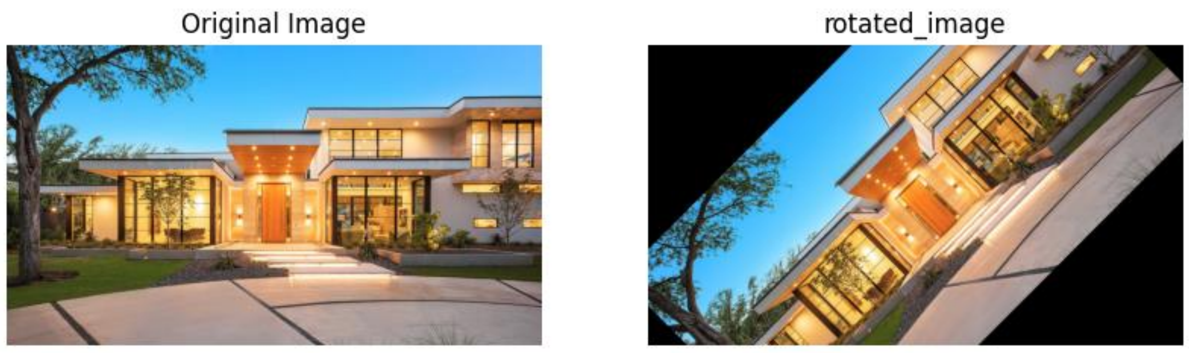
Rotate image



```
# 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()
```
✓ 0.4s

- (h, w) = image_rgb.shape[:2]: Retrieves the height (h) and width (w) of the image.
- center = (w // 2, h // 2): Calculates the center of the image, which is used as the point around which the image will be rotated.

- M = cv2.getRotationMatrix2D(center, 45, 1.0): Creates a rotation matrix M. The parameters are:
    1. center: The center of rotation.
    2. 45: The angle of rotation in degrees (clockwise).
    3. 1.0: The scaling factor (1.0 means no scaling).
- rotated_image = cv2.warpAffine(image_rgb, M, (w, h)): Applies the affine transformation (rotation) to the image using the rotation matrix M. The resulting rotated_image will be rotated by 45 degrees around its center.
- plt.figure(figsize=(10, 5)): Creates a figure with a size of 10x5 inches.
- plt.subplot(1, 2, 1): Creates a subplot in a 1-row, 2-column grid, selecting the first subplot.
- plt.title('Original Image'): Sets the title for the original image subplot.
- plt.imshow(image_rgb): Displays the original RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.subplot(1, 2, 2): Creates the second subplot in the 1-row, 2-column grid.
- plt.title('Rotated Image'): Sets the title for the rotated image subplot.
- plt.imshow(rotated_image): Displays the rotated RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.show(): Renders and displays the images on the screen.



Lab-3

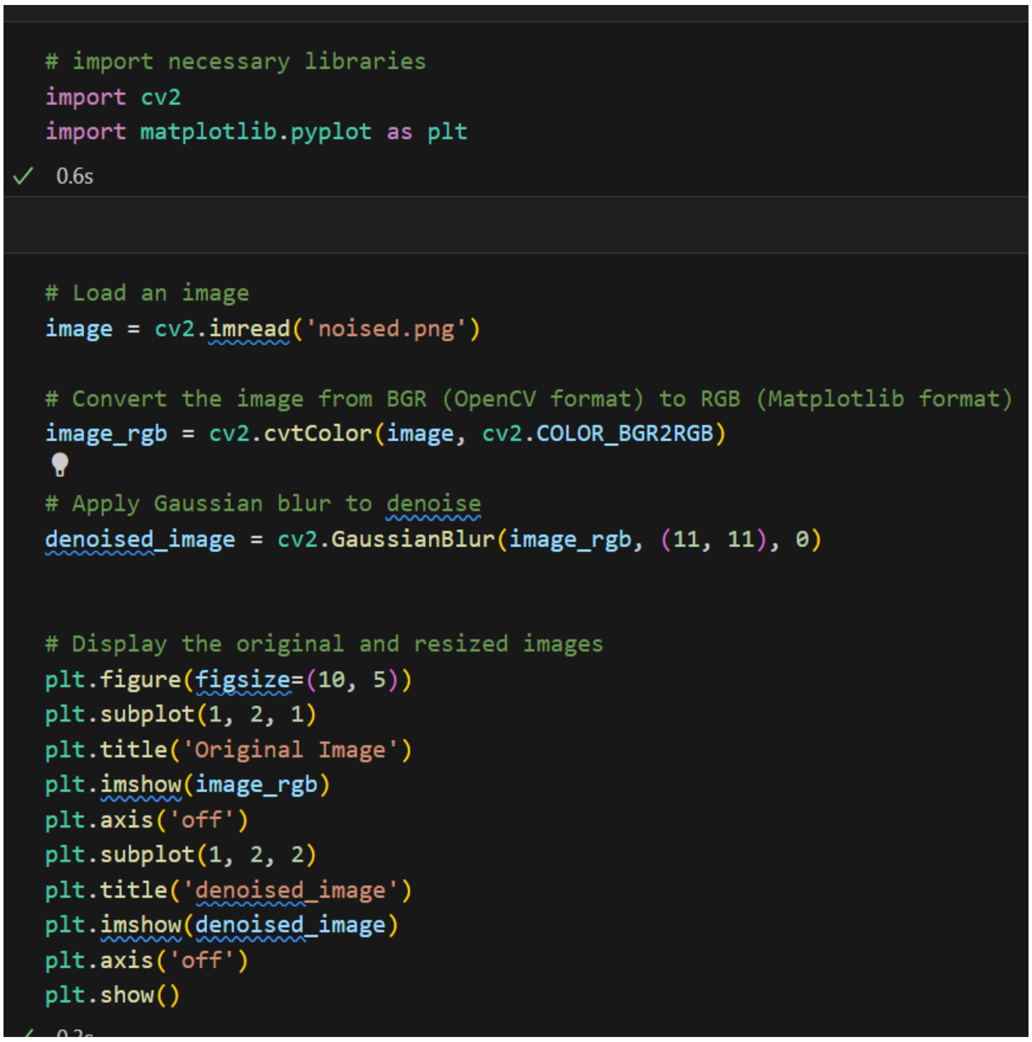
Image Preprocessing Techniques:

- 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.

Procedure

Denoising



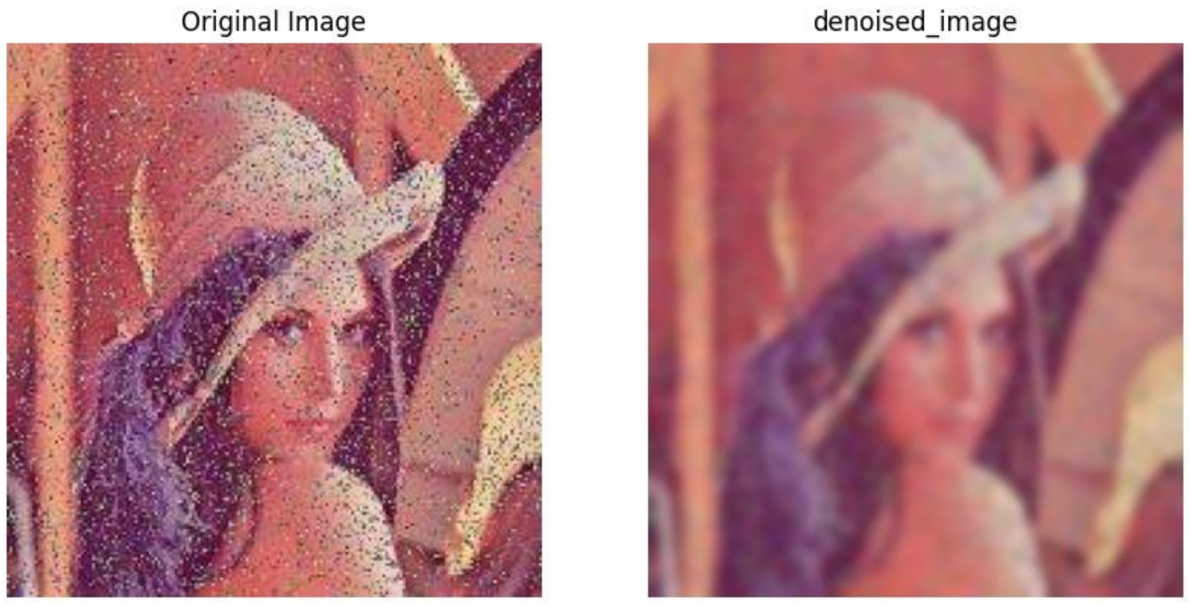
```
# import necessary libraries
import cv2
import matplotlib.pyplot as plt
```

```
✓ 0.6s
```

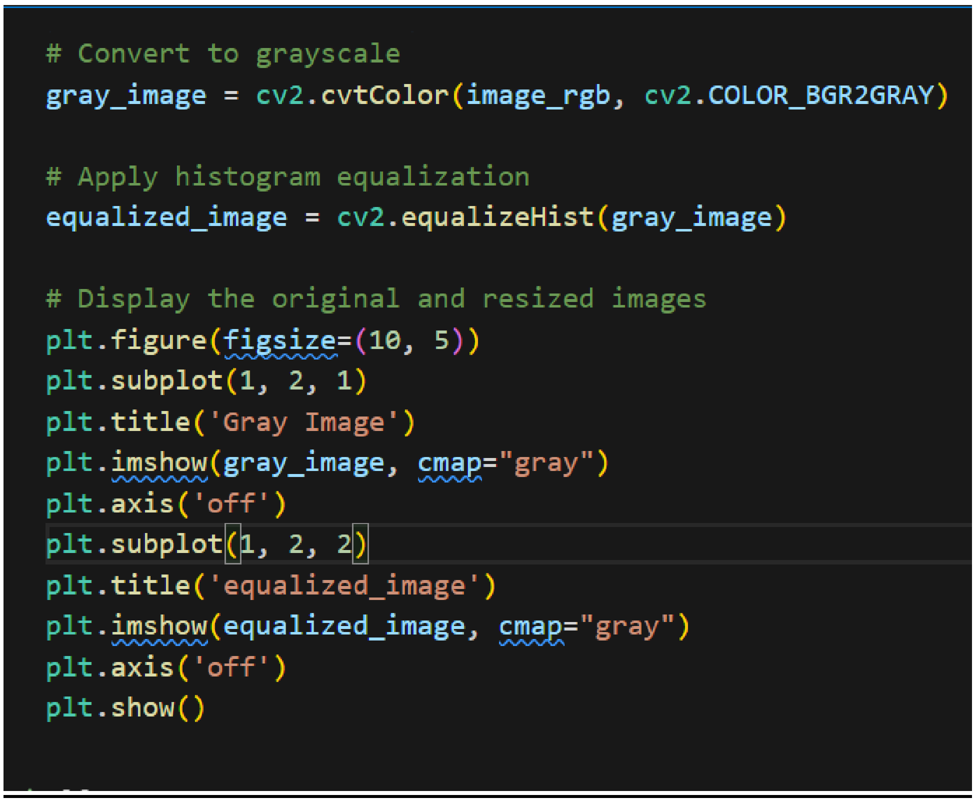
```
# Load an image
image = cv2.imread('noised.png')

# 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()
```

- cv2: The OpenCV library for image processing.
- matplotlib.pyplot: Used for plotting and displaying images.
- cv2.imread('noised.png'): Reads the image file named 'noised.png' and loads it into a NumPy array. The image is in BGR format by default.

- cv2.cvtColor(image, cv2.COLOR_BGR2RGB): Converts the image from BGR color space (used by OpenCV) to RGB color space (used by Matplotlib) to ensure correct color representation when displaying the image.
- cv2.GaussianBlur(image_rgb, (11, 11), 0): Applies Gaussian blur to the image to reduce noise.
- plt.figure(figsize=(10, 5)): Creates a figure with a size of 10x5 inches.
- plt.subplot(1, 2, 1): Creates a subplot in a 1-row, 2-column grid, selecting the first subplot.
- plt.title('Original Image'): Sets the title for the original image subplot.
- plt.imshow(image_rgb): Displays the original RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.subplot(1, 2, 2): Creates the second subplot in the 1-row, 2-column grid.
- plt.title('Denoised Image'): Sets the title for the denoised image subplot.
- plt.imshow(denoised_image): Displays the denoised RGB image.
- plt.axis('off'): Hides the axis lines and labels.
- plt.show(): Renders and displays the images on the screen.



Convert to Grayscale



```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()
```

- cv2.cvtColor(image_rgb, cv2.COLOR_BGR2GRAY): Converts the RGB image to a grayscale image. The result is a 2D array where each value represents the intensity of light, ranging from 0 (black) to 255 (white).
- cv2.equalizeHist(gray_image): Applies histogram equalization to the grayscale image. This technique improves the contrast of the image by spreading out the intensity values across the entire range of 0 to 255. It can enhance details in regions with poor contrast.
- plt.figure(figsize=(10, 5)): Creates a figure with a size of 10x5 inches.
- plt.subplot(1, 2, 1): Creates a subplot in a 1-row, 2-column grid, selecting the first subplot.
- plt.title('Gray Image'): Sets the title for the grayscale image subplot.
- plt.imshow(gray_image, cmap="gray"): Displays the grayscale image using a grayscale colormap.
- plt.axis('off'): Hides the axis lines and labels.
- plt.subplot(1, 2, 2): Creates the second subplot in the 1-row, 2-column grid.
- plt.title('Equalized Image'): Sets the title for the equalized image subplot.
- plt.imshow(equalized_image, cmap="gray"): Displays the histogram-equalized grayscale image using a grayscale colormap.

- plt.axis('off'): Hides the axis lines and labels.
- plt.show(): Renders and displays the images on the screen.