Baku Higher Oil School

Image Processing

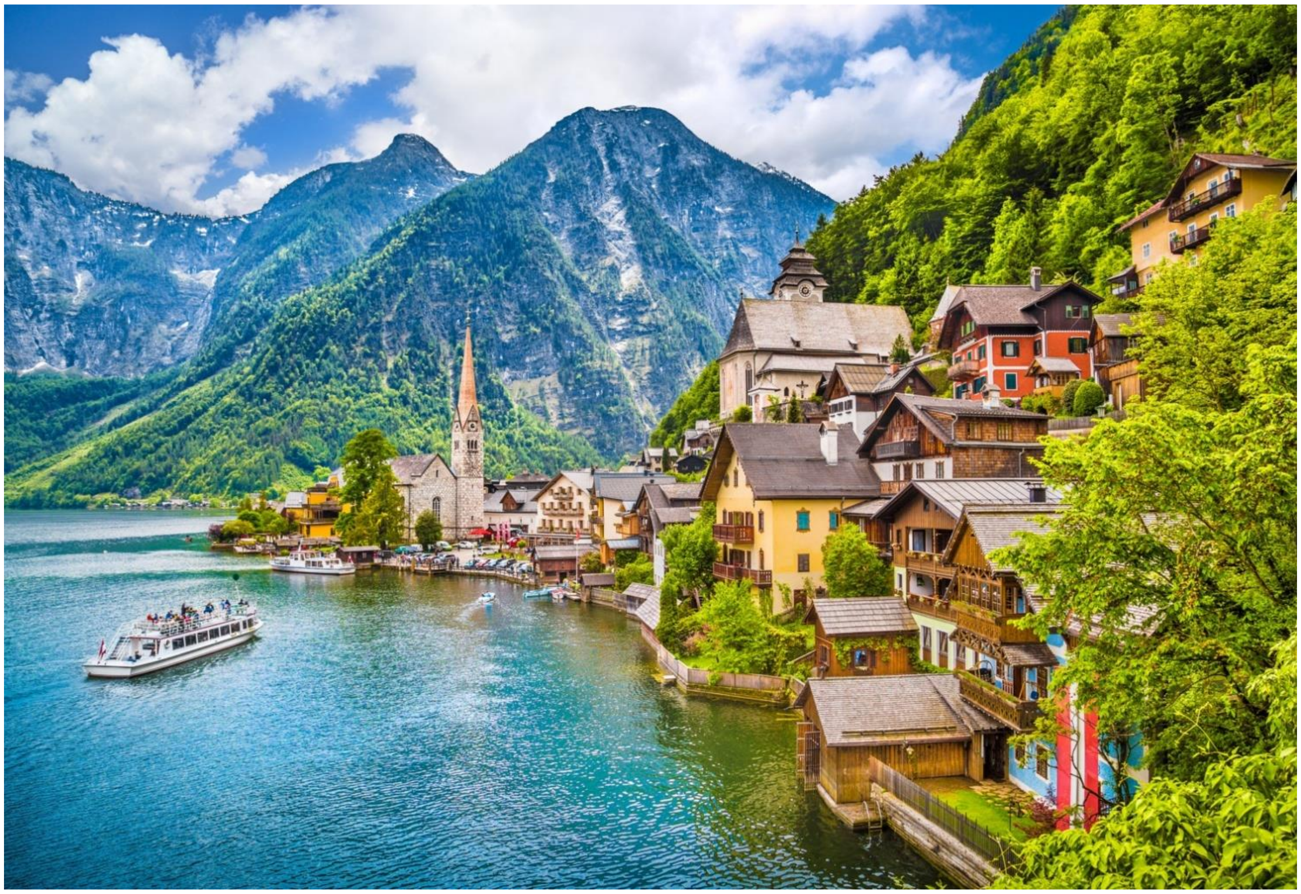
Student's full name:	Rinat Mahmudov
Group number:	PAE 22.2
Report submitted on:	09.27.2025
Instructor name:	PhD, Associate Professor Leyla Muradkhanli

Lab work 2. Working with Images in Python with OpenCV

1. Reading and Displaying the Image

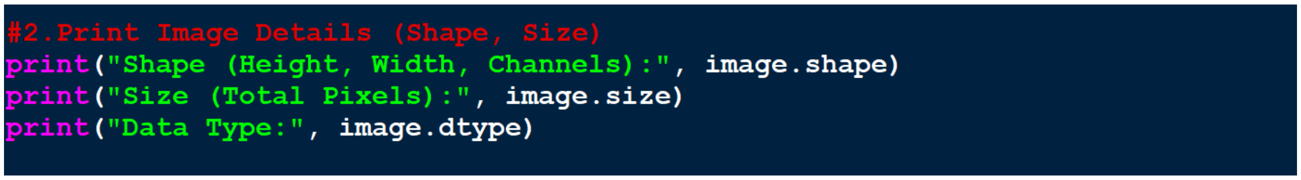
The image file is loaded into the program using cv2.imread() and displayed on the screen with cv2.imshow(). The cv2.waitKey(0) function keeps the window open until a key is pressed. Finally, cv2.destroyAllWindows() closes the window.

```
#1.Read and Display an Image
import cv2

# Load the image
image = cv2.imread('landscape.jpg')

# Display the image
cv2.imshow('Original Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

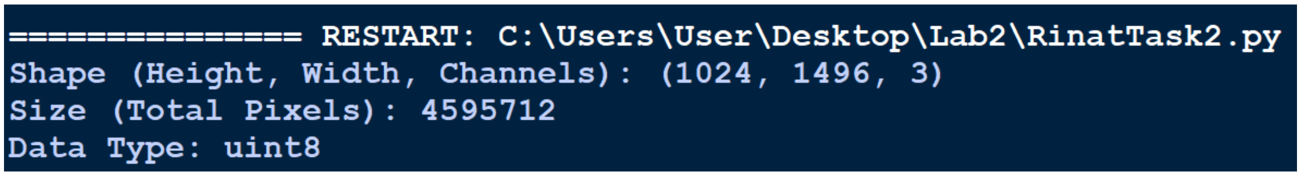


2. Image Details

image.shape shows the height, width, and number of channels in the image. image.size returns the total number of pixels, while image.dtype indicates the data type of the pixel values. These details provide basic technical information about the image.



```
#2.Print Image Details (Shape, Size)
print("Shape (Height, Width, Channels):", image.shape)
print("Size (Total Pixels):", image.size)
print("Data Type:", image.dtype)
```



```
=============== RESTART: C:\Users\User\Desktop\Lab2\RinatTask2.py
Shape (Height, Width, Channels): (1024, 1496, 3)
Size (Total Pixels): 4595712
Data Type: uint8
```

3. Converting the Image to Grayscale

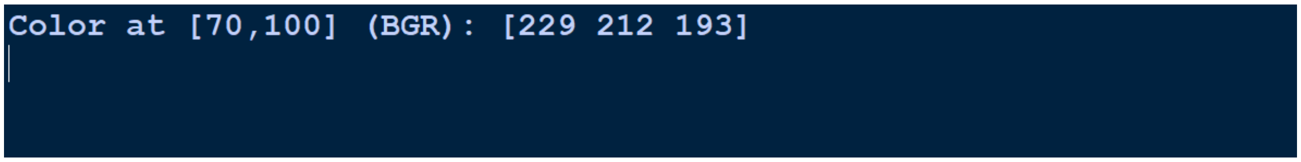
The cv2.cvtColor() function is used to convert the color image into grayscale. This operation reduces the color information, making the image simpler. Grayscale images are often used in processing tasks since they make calculations faster and easier.

```
#3.Convert the Image to Grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Grayscale Image', gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. Finding the Color of a Specific Pixel

The color of the pixel at coordinates [70, 100] is retrieved. The result is displayed in BGR (Blue, Green, Red) format. This operation is useful for analyzing the color of specific regions in an image.

```
#4.Find Color of Pixel at [70, 100]
pixel_color = image[70, 100]
print("Color at [70,100] (BGR):", pixel_color)
```



```
Color at [70,100] (BGR): [229 212 193]
```

5. Modifying a Pixel’s Color

The pixel at [200, 150] is changed to red ([0, 0, 255]). This demonstrates how a single pixel’s color can be manipulated directly in the image.

```
#5.Modify Pixel at [200, 150] to Red (BGR: [0, 0, 255])
image[200, 150] = [0, 0, 255]
```

6. Modifying a Block of Pixels

The block of pixels within coordinates [100:300, 150:400] is filled with green. This approach allows for quick modification of larger areas in the image.

```
# 6. Change Pixel Block [100:300, 150:400] to Green
image[100:300, 150:400] = [0, 255, 0]

cv2.imshow('Modified Image', image)
cv2.imwrite('output_modified.jpg', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

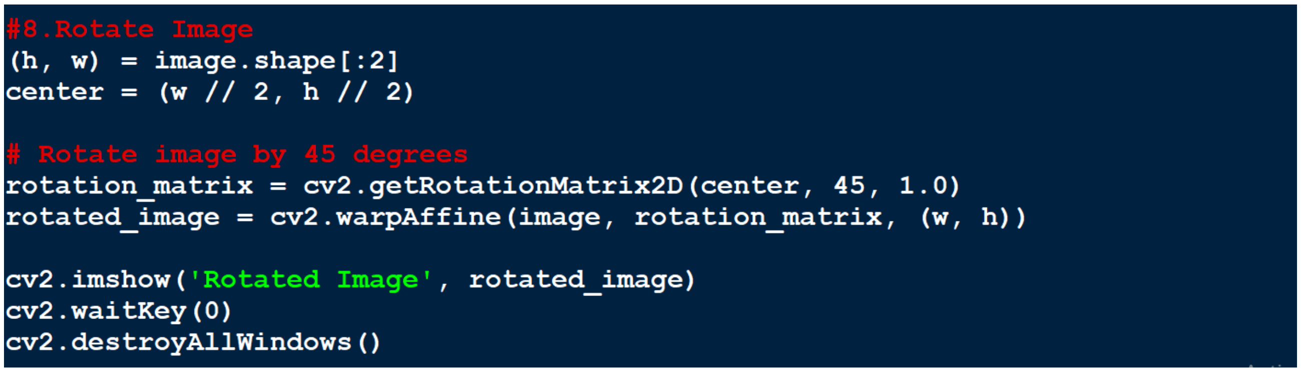
7. Resizing the Image

The cv2.resize() function resizes the image to 300x300 dimensions. Resizing is commonly used to standardize image sizes or to reduce data size.

```
#7.Resize Image
resized_image = cv2.resize(image, (300, 300))
cv2.imshow('Resized Image', resized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8. Rotating the Image

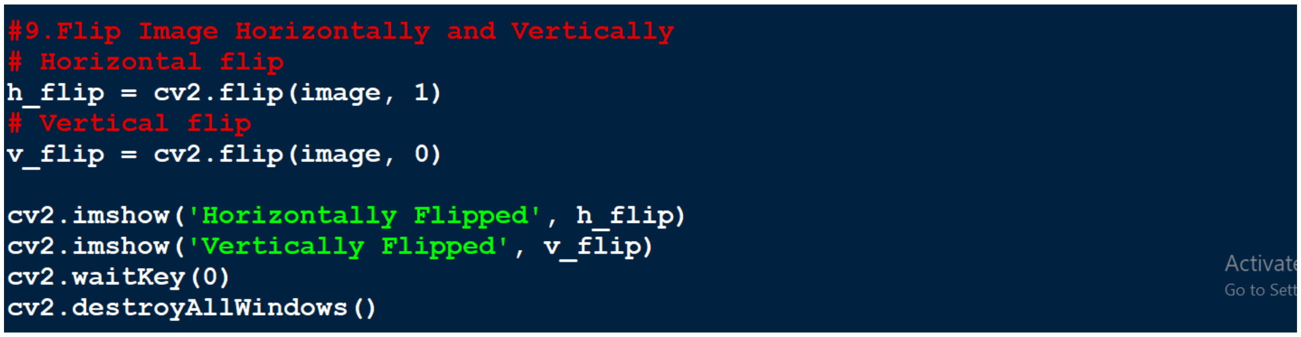
The center of the image is calculated first, and then cv2.getRotationMatrix2D() generates a rotation matrix for a 45° angle. Finally, cv2.warpAffine() applies this matrix to rotate the image.



```
#8.Rotate Image
(h, w) = image.shape[:2]
center = (w // 2, h // 2)

# Rotate image by 45 degrees
rotation_matrix = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated_image = cv2.warpAffine(image, rotation_matrix, (w, h))

cv2.imshow('Rotated Image', rotated_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

9. Flipping the Image

The cv2.flip() function flips the image horizontally (mirror effect) and vertically. Flipping is useful for creating symmetry or applying different visual effects to images.



```
#9.Flip Image Horizontally and Vertically
# Horizontal flip
h_flip = cv2.flip(image, 1)
# Vertical flip
v_flip = cv2.flip(image, 0)

cv2.imshow('Horizontally Flipped', h_flip)
cv2.imshow('Vertically Flipped', v_flip)
cv2.waitKey(0)
cv2.destroyAllWindows()
```