INT3404E - Image Processing: HW 1

21021531 - Le The Quang

29/3/2024

1 Homework

Figure 1: Original Image

A picture or image can be represented as a NumPy array of "pixels", with dimensions H × W × C, where H is the height,W is the width, and C is the number of color channels. Figure 1 illustrates the coordinate system. The origin is at the top left corner and the first dimension indicates the Y (row) direction, while the second dimension indicates the X (column) dimension. Typically we will use an image with channels that give the Red, Green, and Blue "level" of each pixel,which is referred to in the short form

RGB. The value for each channel ranges from 0 (darkest) to 255 (lightest). However,when loading an image through Matplotlib,this range will be scaled from 0 (darkest) to 1 (brightest) instead, and will be a real number, rather than an integer.
You will write Python code to load an image perform several manipulations to the image and visualize their effects.
Report the result of the functions flip_image, rotate_image, grayscale_image.

2 Homework Report

2.1 Basic function

`load_image(image_path)` function: Load an image from file using OpenCV, use function *imread*.
`display_image(image, title="Image")` function: Display an image using matplotlib, use *plt.show()* to display the image.
`save_image(image, output_path)` function: Save an image to file using OpenCV, use function *imwrite*.
The result is displayed in Figure 1.

```
# Load an image from file as function
def load_image(image_path):
    return cv2.imread(image_path)

# Display an image as function
def display_image(image, title="Image"):
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.axis("off")
    plt.show()
# Save an image as function
def save_image(image, output_path):
    """
    Save an image to file using OpenCV
    """
    cv2.imwrite(output_path, image)
```

2.2 function `grayscale_image`

`grayscale_image(image)` function: Convert an image to grayscale. Convert the original image to a grayscale image.
In a grayscale image, the pixel value of the 3 channels will be the same for a particular X, Y coordinate. The equation for the pixel is given by:

$$p = 0.299R + 0.587G + 0.114B$$

Where the R, G, B are the values for each of the corresponding channels. We will do this by creating an array called img_gray with the same shape as img

The result is displayed in Figure 1.

Listing 1: Code of function grayscale_image

```
def grayscale_image(image):
  """
  Convert an image to grayscale. Convert the original image
  to a grayscale image. In a grayscale image, the pixel
 value of the
  3 channels will be the same for a particular X, Y
 coordinate. The equation for the pixel value
  [1] is given by:
      p = 0.299R + 0.587G + 0.114B
  Where the R, G, B are the values for each of the
 corresponding channels. We will do this by
  creating an array called img_gray with the same shape as
 img
  """
  # Extract R, G, B channels
  B = image[:, :, 0]
  G = image[:, :, 1]
  R = image[:, :, 2]

  # Calculate grayscale value with provided equation
  img_gray = 0.299 * R + 0.587 * G + 0.114 * B

  # Convert dtype to uint8
  img_gray = img_gray.astype(np.uint8)
  return img_gray
```

Figure 2: Gray-scale Image

2.3 function `flip_image`

`flip_image(image)` function: Flip an image horizontally using OpenCV.
Use function : cv2.flip(image, 1)
- image: The data of the input image
- flipCode = 1 Specify that the image will be flipped along the y-axis.

The result is displayed in Figure 3.

Listing 2: Code of function flip_image

```
def flip_image(image):
    """
    Flip an image horizontally using OpenCV
    """
    return cv2.flip(image, 1)
```

Figure 3: Flipped Gray-scale Image

2.4 function `rotate_image`

`rotate_image(image, angle)` function: Rotate an image using OpenCV. The angle is in degrees. The result is displayed in Figure 4.

Listing 3: Code of function rotate_image

```
def rotate_image(image, angle):
    """
    Rotate an image using OpenCV. The angle is in degrees
    """
    # Get height and width of image
    height, width = image.shape[:2]
    # Calculate rotation matrix
    rotation_matrix = cv2.getRotationMatrix2D((width / 2,
   height / 2), angle, 1)
    # Apply rotation to image
    rotated_image = cv2.warpAffine(image, rotation_matrix, (
   width, height))

    return rotated_image
```

`cv2.getRotationMatrix2D()` function takes two arguments: the rotation center (in this case,calculated by taking half of the width and the height of the image), the rotation angle and scale factor(setted to 1 here, meaning no scaling of the image).

`cv2.warpAffine()` function takes three arguments: the original image, the rotation matrix, and the size of the output image (original width and height of the image). This function will apply the rotation defined by the rotation matrix `rotation_matrix`.

Figure 4: Rotated Gray-scale Image