Reading, Saving, and Displaying an Image in Python

Bindeshwar Singh Kushwaha
Postnetwork Academy

Reading, Saving, and Displaying an Image Using Matplotlib

Code

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
im = mpimg.imread("brown_val1.jpg") # read the image from disk as a numpy ndarray
print(im.shape, im.dtype, type(im)) # print image shape, dtype, and type
plt.imshow(im) # display the image
plt.axis('off') # turn off the axis
plt.show() # show the image in a plot
```

Reading, Saving, and Displaying an Image Using Matplotlib

Code

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
im = mpimg.imread("brown_val1.jpg") # read the image from disk as a numpy ndarray
print(im.shape, im.dtype, type(im)) # print image shape, dtype, and type
plt.imshow(im) # display the image
plt.axis('off') # turn off the axis
plt.show() # show the image in a plot
```

Output

Explanation of Code

- `im = mpimg.imread("brown_val1.jpg")`
  This line reads an image file (`brown_val1.jpg`) from the disk and loads it as a NumPy ndarray, where each pixel is represented as a floating-point number between 0 and 1.

Explanation of Code

- `im = mpimg.imread("brown_val1.jpg")`
  This line reads an image file (`brown_val1.jpg`) from the disk and loads it as a NumPy ndarray, where each pixel is represented as a floating-point number between 0 and 1.
- `print(im.shape, im.dtype, type(im))`
  This line prints the shape (dimensions), data type, and the type of the object `im`. This helps us understand the structure and format of the image.

Explanation of Code

- `im = mpimg.imread("brown_val1.jpg")`
  This line reads an image file (`brown_val1.jpg`) from the disk and loads it as a NumPy ndarray, where each pixel is represented as a floating-point number between 0 and 1.
- `print(im.shape, im.dtype, type(im))`
  This line prints the shape (dimensions), data type, and the type of the object `im`. This helps us understand the structure and format of the image.
- `# (1600, 1600, 3) uint8 <class 'numpy.ndarray'>`
  This is an example of the output produced by the `print` statement, showing the image's shape and its data type. It means the image is 960 pixels high, 1280 pixels wide, and has 3 channels (RGB).

Explanation of Code

- `im = mpimg.imread("brown_val1.jpg")`
  This line reads an image file (`brown_val1.jpg`) from the disk and loads it as a NumPy ndarray, where each pixel is represented as a floating-point number between 0 and 1.
- `print(im.shape, im.dtype, type(im))`
  This line prints the shape (dimensions), data type, and the type of the object `im`. This helps us understand the structure and format of the image.
- `# (1600, 1600, 3) uint8 <class 'numpy.ndarray'>`
  This is an example of the output produced by the `print` statement, showing the image's shape and its data type. It means the image is 960 pixels high, 1280 pixels wide, and has 3 channels (RGB).
- `plt.imshow(im)`
  This function displays the image stored in `im` in the current plot window.

Explanation of Code

- `im = mpimg.imread("brown_val1.jpg")`
  This line reads an image file (`brown_val1.jpg`) from the disk and loads it as a NumPy ndarray, where each pixel is represented as a floating-point number between 0 and 1.
- `print(im.shape, im.dtype, type(im))`
  This line prints the shape (dimensions), data type, and the type of the object `im`. This helps us understand the structure and format of the image.
- `# (1600, 1600, 3) uint8 <class 'numpy.ndarray'>`
  This is an example of the output produced by the `print` statement, showing the image's shape and its data type. It means the image is 960 pixels high, 1280 pixels wide, and has 3 channels (RGB).
- `plt.imshow(im)`
  This function displays the image stored in `im` in the current plot window.
- `plt.axis('off')`
  This command turns off the axis, removing grid lines and labels, so the focus is entirely on the image itself.

Explanation of Code

- `im = mpimg.imread("brown_val1.jpg")`
  This line reads an image file (`brown_val1.jpg`) from the disk and loads it as a NumPy ndarray, where each pixel is represented as a floating-point number between 0 and 1.
- `print(im.shape, im.dtype, type(im))`
  This line prints the shape (dimensions), data type, and the type of the object `im`. This helps us understand the structure and format of the image.
- `# (1600, 1600, 3) uint8 <class 'numpy.ndarray'>`
  This is an example of the output produced by the `print` statement, showing the image's shape and its data type. It means the image is 960 pixels high, 1280 pixels wide, and has 3 channels (RGB).
- `plt.imshow(im)`
  This function displays the image stored in `im` in the current plot window.
- `plt.axis('off')`
  This command turns off the axis, removing grid lines and labels, so the focus is entirely on the image itself.
- `plt.show()`
  Displays the plot containing the image on the screen.

Reading and Displaying an Image Using skimage

Code

```
import skimage.io as io
im = io.imread("brown_val1.jpg") # read the image from disk as a numpy ndarray
io.imshow(im) # display the image
io.show() # show the image in a window
```

Reading and Displaying an Image Using skimage

Code

```
import skimage.io as io
im = io.imread("brown_val1.jpg") # read the image from disk as a numpy ndarray
io.imshow(im) # display the image
io.show() # show the image in a window
```

Output

Explanation of Code

- `im = io.imread("brown_val1.jpg")`
  This line uses the `io.imread` function from `skimage` to read an image file from the disk and store it in the variable `im`.

Explanation of Code

- `im = io.imread("brown_val1.jpg")`
  This line uses the `io.imread` function from `skimage` to read an image file from the disk and store it in the variable `im`.
- `io.imshow(im)`
  This function is used to prepare the image for display. It takes the image stored in `im` and gets it ready for rendering in the plot.

Explanation of Code

- `im = io.imread("brown_val1.jpg")`
  This line uses the `io.imread` function from `skimage` to read an image file from the disk and store it in the variable `im`.
- `io.imshow(im)`
  This function is used to prepare the image for display. It takes the image stored in `im` and gets it ready for rendering in the plot.
- `io.show()`
  Finally, the `io.show()` function displays the image in a window, allowing the user to view it.

Reading and Displaying an Image Using OpenCV

Code

```
import cv2
im = cv2.imread("brown_val1.jpg") # read the image from disk as a numpy ndarray
cv2.imshow("Displayed Image", im) # display the image in a window
cv2.waitKey(0) # wait for a key press indefinitely
cv2.destroyAllWindows() # close all OpenCV windows
```

Reading and Displaying an Image Using OpenCV

Code

```
import cv2
im = cv2.imread("brown_val1.jpg") # read the image from disk as a numpy ndarray
cv2.imshow("Displayed Image", im) # display the image in a window
cv2.waitKey(0) # wait for a key press indefinitely
cv2.destroyAllWindows() # close all OpenCV windows
```

Output

Explanation of Code

- `im = cv2.imread("brown_val1.jpg")`
  This line reads the image file from disk using OpenCV and stores it in the variable `im` as a NumPy array.

Explanation of Code

- `im = cv2.imread("brown_val1.jpg")`
  This line reads the image file from disk using OpenCV and stores it in the variable `im` as a NumPy array.
- `cv2.imshow("Displayed Image", im)`
  This function creates a window titled `"Displayed Image"` and displays the image stored in `im`.

Explanation of Code

- `im = cv2.imread("brown_val1.jpg")`
  This line reads the image file from disk using OpenCV and stores it in the variable `im` as a NumPy array.
- `cv2.imshow("Displayed Image", im)`
  This function creates a window titled `"Displayed Image"` and displays the image stored in `im`.
- `cv2.waitKey(0)`
  Waits indefinitely until a key is pressed before closing the window.

Explanation of Code

- `im = cv2.imread("brown_val1.jpg")`
  This line reads the image file from disk using OpenCV and stores it in the variable `im` as a NumPy array.
- `cv2.imshow("Displayed Image", im)`
  This function creates a window titled `"Displayed Image"` and displays the image stored in `im`.
- `cv2.waitKey(0)`
  Waits indefinitely until a key is pressed before closing the window.
- `cv2.destroyAllWindows()`
  Closes all OpenCV-created windows and releases resources.

Reach PostNetwork Academy

Website

www.postnetwork.co

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

LinkedIn Page

www.linkedin.com/company/postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

LinkedIn Page

www.linkedin.com/company/postnetworkacademy

GitHub Repositories

www.github.com/postnetworkacademy

Thank You!