Day16_Image_to_Array_Numpy

June 4, 2025

Summary: Image Processing with NumPy (Array) and PIL

In this exercise, we explored how digital images can be processed using Python libraries like **PIL**, **NumPy**, and **Matplotlib**.

What We Did & Why

1. **Created a Simple Matrix for Visualization** We started by generating a 5×5 matrix filled with the value 255, which simulates a small white pixel grid. This helped us understand how brightness and pixel intensity are stored numerically.
2. **Loaded Real Images** We imported two image files using the PIL library. This step allowed us to handle actual image data, converting them into objects that can be manipulated in Python.
3. **Converted Images to NumPy Arrays** To work directly with pixel values, we transformed the images into NumPy arrays. Each pixel was represented as a combination of three values — Red, Green, and Blue — forming a 3D array.
4. **Copied the Arrays for Safe Editing** We created duplicates of the image arrays to preserve the original versions, ensuring that our experiments wouldn't overwrite the original image data.
5. **Verified Image Equality** We confirmed that the copies were exactly the same as the originals by comparing each corresponding pixel value.
6. **Visualized the Original and Copied Images** Using `matplotlib`, we displayed the images to confirm they loaded and were converted properly.
7. **Explored Color Channels** We isolated and visualized the **Red**, **Green**, and **Blue** channels separately. This helped us see how each color contributes to the full image. Each channel showed different brightness patterns based on how much of that color was present in each pixel.
8. **Applied Color Maps for Better Understanding** To enhance visibility, we used color maps (like greyscale and yellow-green) when visualizing individual channels. This made it easier to interpret brightness and color intensities.

Why This Is Useful

- **Image Understanding**: You learned that images are just numerical data structured as matrices.
- **Color Decomposition**: Breaking down images into R, G, B channels helps in color correction, filtering, and analysis.
- **Data Manipulation**: You practiced matrix slicing — a powerful technique used in computer vision and deep learning.
- **Foundation for Advanced Tasks**: These steps are the foundation for more complex image processing tasks like filtering, masking, blending, object detection, and more.

In Python, images can be opened using the PIL (Python Imaging Library) and converted into NumPy arrays. This makes them easy to manipulate using matrix operations.Each image is represented as a 3D NumPy array:

(height, width, color_channels)

The color channels are:

0 = Red,

1 = Green,

2 = Blue

This means we can directly slice and view or modify individual channels.

[1]:
```
import numpy as np # Import necessary libraries
```

[2]:
```
# Create a 5x5 matrix filled with 1s and multiply by 255
# This is like simulating a white pixel grid because 255 is the max brightness␣
↪in an 8-bit image

ones_arr = np.ones((5,5),dtype = int)
ones_arr
```

[2]:
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

[3]:
```
ones_arr * 255
```

[3]:
```
array([[255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255]])
```

[4]:
```
import matplotlib.pyplot as plt
```

[5]:
```
from PIL import Image # python imaging library
```

```
[11]: # Load two images using PIL
      Image_Color_bubbles = Image.open(r'C:\Users\aksha\Downloads\Image1.jfif')
```

```
[12]: Image_Color_bubbles
```

[12]:

```
[13]: Image_cat = Image.open(r'C:\Users\aksha\Downloads\cat.jfif') #raw string
```

```
[14]: Image_cat
```

[14]:

```
[15]: type(Image_Color_bubbles)
```

```
[15]: PIL.JpegImagePlugin.JpegImageFile
```

```
[16]: type(Image_cat)
```

```
[16]: PIL.JpegImagePlugin.JpegImageFile
```

```
[17]: # Convert images to NumPy arrays to work with them numerically
      Image1 = np.asarray(Image_Color_bubbles)
      Image1
```

```
[17]: array([[[254, 126,  27],
              [237, 112,  32],
              [206,  91,  36],
              …,
              [197,  57,  66],
              [192,  56,  66],
              [184,  54,  64]],

             [[244, 116,  29],
              [223, 102,  31],
              [187,  82,  34],
              …,
              [205,  62,  64],
              [199,  60,  63],
              [192,  59,  64]],

             [[226, 101,  34],
              [199,  88,  32],
              [159,  71,  33],
              …,
              [210,  63,  53],
              [205,  62,  54],
              [199,  60,  55]],

             …,

             [[253, 138,   9],
              [249, 145,   0],
              [253, 150,   0],
              …,
              [ 90,  55,  36],
              [ 91,  56,  36],
              [ 94,  58,  34]],

             [[255, 139,   0],
              [255, 146,   6],
              [255, 146,  10],
              …,
              [ 91,  54,  36],
              [ 91,  54,  36],
              [ 92,  55,  37]],
             [[250, 133,   0],
              [254, 141,   3],
              [255, 143,   7],
              …,
              [ 91,  54,  36],
              [ 91,  54,  36],
              [ 92,  55,  37]]], dtype=uint8)
```

```
[18]: Image2 = np.asarray(Image_cat)
      Image2
```

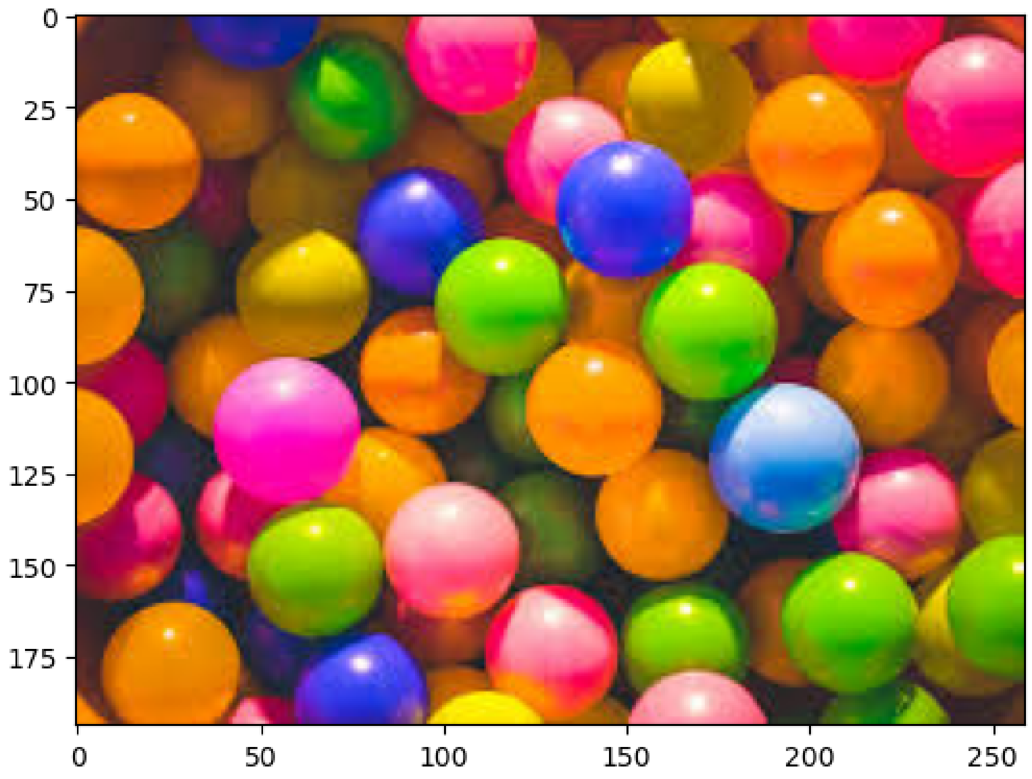
```
[18]: array([[[ 91,  91, 103],
              [ 58,  58,  70],
              [ 58,  58,  70],
              …,
              [ 59,  56,  63],
              [ 59,  56,  63],
              [ 60,  57,  64]],

             [[ 77,  77,  89],
              [ 71,  71,  83],
              [ 90,  90, 102],
              …,
              [ 60,  57,  64],
              [ 60,  57,  64],
              [ 61,  58,  65]],

             [[ 63,  63,  75],
              [ 86,  86,  98],
              [ 90,  90, 102],
              …,
              [ 61,  58,  65],
              [ 62,  59,  66],
              [ 62,  59,  66]],

             …,

             [[183, 184, 176],
              [184, 185, 177],
              [186, 187, 179],
              …,
              [152, 155, 160],
              [151, 154, 159],
              [150, 153, 158]],

             [[186, 187, 179],
              [184, 185, 177],
              [182, 183, 175],
              …,
              [152, 155, 160],
              [151, 154, 159],
              [149, 152, 157]],

             [[188, 189, 181],
              [185, 186, 178],
              [180, 181, 173],
              …,
              [152, 155, 160],
              [150, 153, 158],
              [148, 151, 156]]], dtype=uint8)
```

[20]:
```
# Check image types (should be NumPy arrays now)
print(type(Image1))
print(type(Image2))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

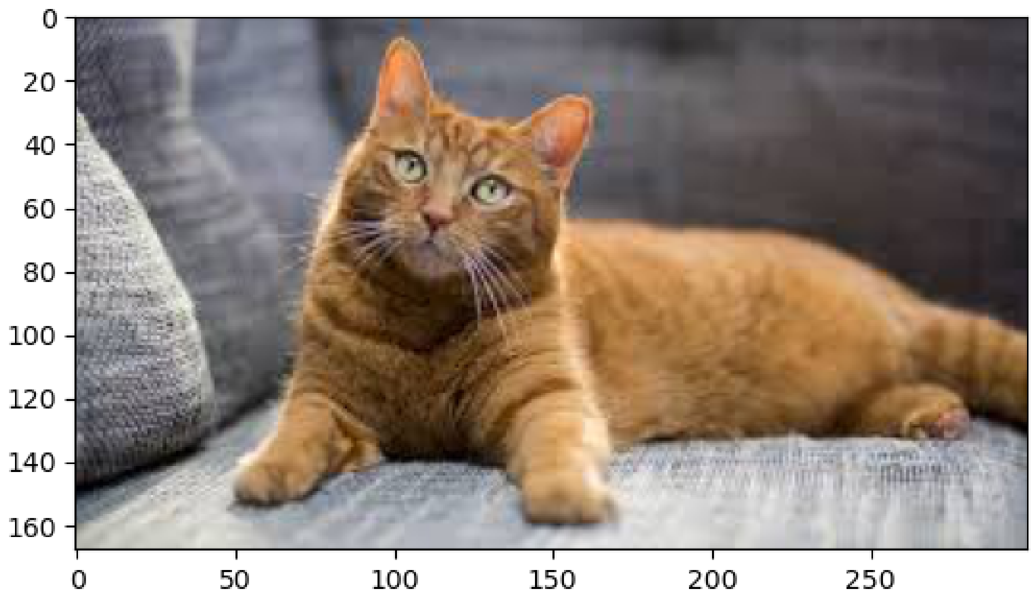
[21]:
```
# Show the original images using matplotlib
plt.imshow(Image1)
```

[21]:
```
<matplotlib.image.AxesImage at 0x27e48747d40>
```



```
[22]: plt.imshow(Image2)
```

```
[22]: <matplotlib.image.AxesImage at 0x27e48747350>
```



[23]:
```
# Get the shape of images
# Shape will be (height, width, 3) → for RGB
Image1.shape
```

[23]:
```
(194, 259, 3)
```

[24]:
```
Image2.shape
```

[24]:
```
(168, 300, 3)
```

[25]:
```
# Make safe copies before modifying
Image1_red = Image1.copy()
Image1_red
```

[25]:
```
array([[[254, 126,  27],
        [237, 112,  32],
        [206,  91,  36],
        …,
        [197,  57,  66],
        [192,  56,  66],
        [184,  54,  64]],

       [[244, 116,  29],
        [223, 102,  31],
        [187,  82,  34],
        …,
        [205,  62,  64],
        [199,  60,  63],
        [192,  59,  64]],

       [[226, 101,  34],
        [199,  88,  32],
        [159,  71,  33],
        …,
        [210,  63,  53],
        [205,  62,  54],
        [199,  60,  55]],

       …,

       [[253, 138,   9],
        [249, 145,   0],
        [253, 150,   0],
        …,
        [ 90,  55,  36],
        [ 91,  56,  36],
        [ 94,  58,  34]],

       [[255, 139,   0],
        [255, 146,   6],
        [255, 146,  10],
        …,
        [ 91,  54,  36],
        [ 91,  54,  36],
        [ 92,  55,  37]],

       [[250, 133,   0],
        [254, 141,   3],
        [255, 143,   7],
        …,
        [ 91,  54,  36],
        [ 91,  54,  36],
        [ 92,  55,  37]]], dtype=uint8)
```

```
[26]: Image2_red = Image2.copy()
      Image2_red
```

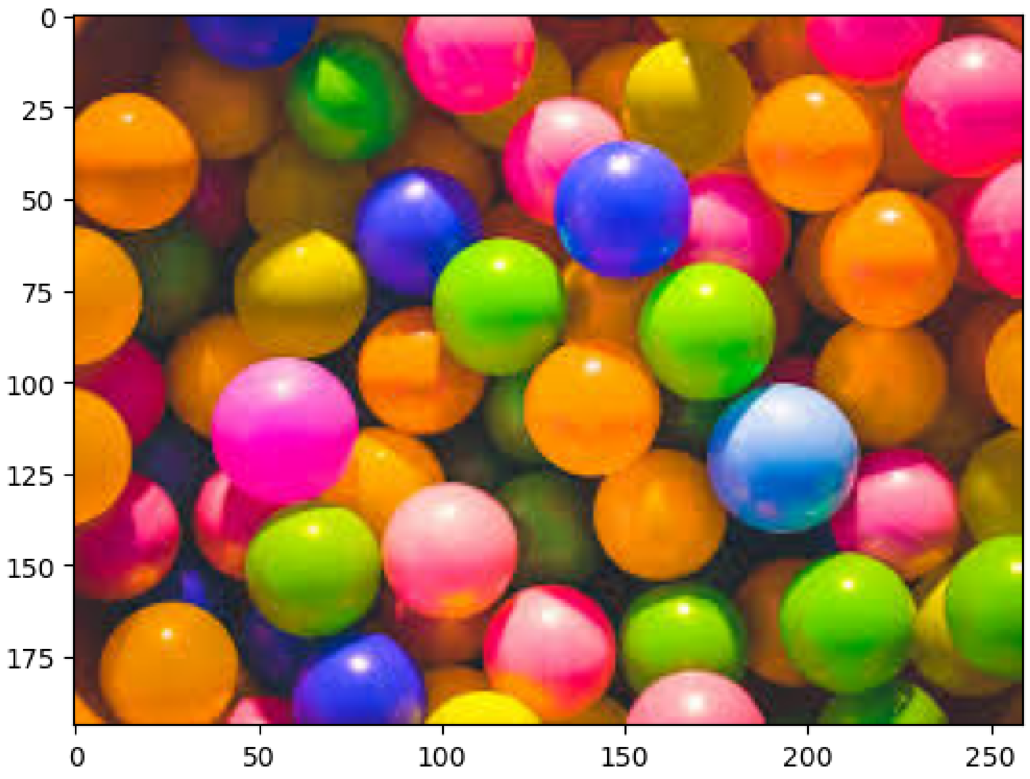
```
[26]: array([[[ 91,  91, 103],
         [ 58,  58,  70],
         [ 58,  58,  70],
         …,
         [ 59,  56,  63],
         [ 59,  56,  63],
         [ 60,  57,  64]],

        [[ 77,  77,  89],
         [ 71,  71,  83],
         [ 90,  90, 102],
         …,
         [ 60,  57,  64],
         [ 60,  57,  64],
         [ 61,  58,  65]],

        [[ 63,  63,  75],
         [ 86,  86,  98],
         [ 90,  90, 102],
         …,
         [ 61,  58,  65],
         [ 62,  59,  66],
         [ 62,  59,  66]],

       …,
        [[183, 184, 176],
         [184, 185, 177],
         [186, 187, 179],
         …,
         [152, 155, 160],
         [151, 154, 159],
         [150, 153, 158]],

        [[186, 187, 179],
         [184, 185, 177],
         [182, 183, 175],
         …,
         [152, 155, 160],
         [151, 154, 159],
         [149, 152, 157]],

        [[188, 189, 181],
         [185, 186, 178],
         [180, 181, 173],
         …,
         [152, 155, 160],
         [150, 153, 158],
         [148, 151, 156]]], dtype=uint8)
```

[27]:
```
# Confirm copies are identical using array comparison
# This checks if all pixel values are exactly the same

Image1 == Image1_red
```

[27]:
```
array([[[ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True],
        …,
        [ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True]],

       [[ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True],
        …,
        [ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True]],

       [[ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True],
        …,
        [ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True]],

       …,

       [[ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True],
        …,
        [ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True]],

       [[ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True],
        …,
        [ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True]],

       [[ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True],
        …,
        [ True,  True,  True],
        [ True,  True,  True],
        [ True,  True,  True]]])
```

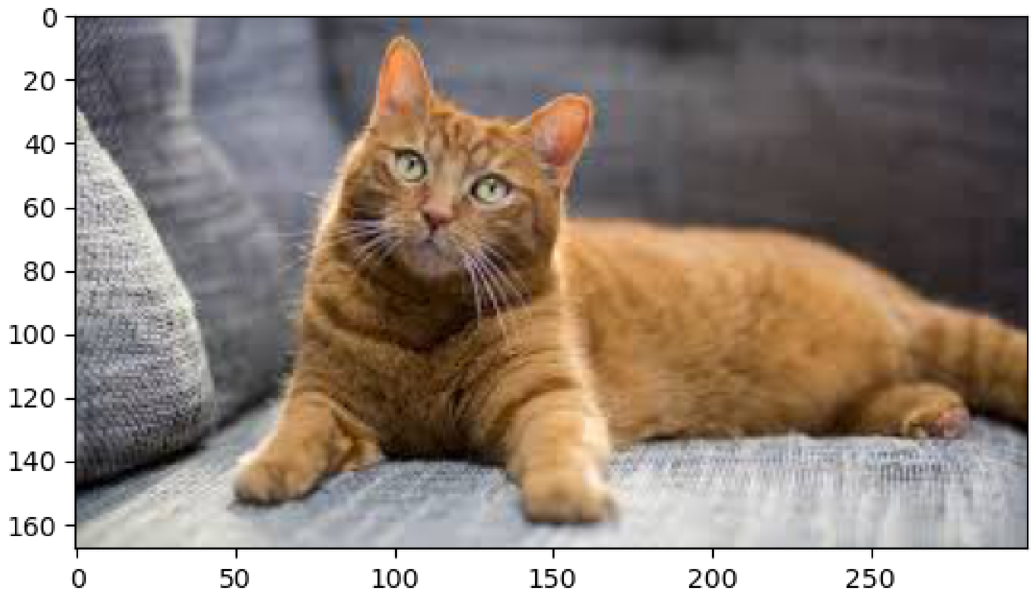
[28]:
```
# Show copies (should be identical to original)
plt.imshow(Image1_red)
```

[28]:
```
<matplotlib.image.AxesImage at 0x27e48184bc0>
```



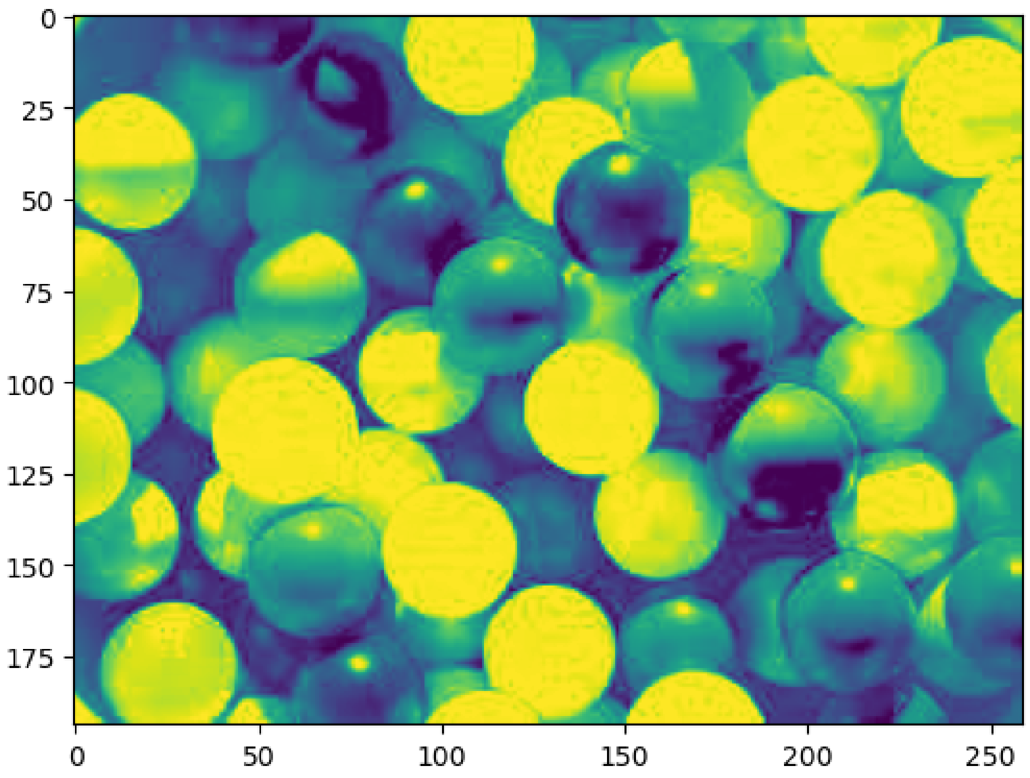
[29]:
```
plt.imshow(Image2_red)
```

[29]:
```
<matplotlib.image.AxesImage at 0x27e487aed50>
```



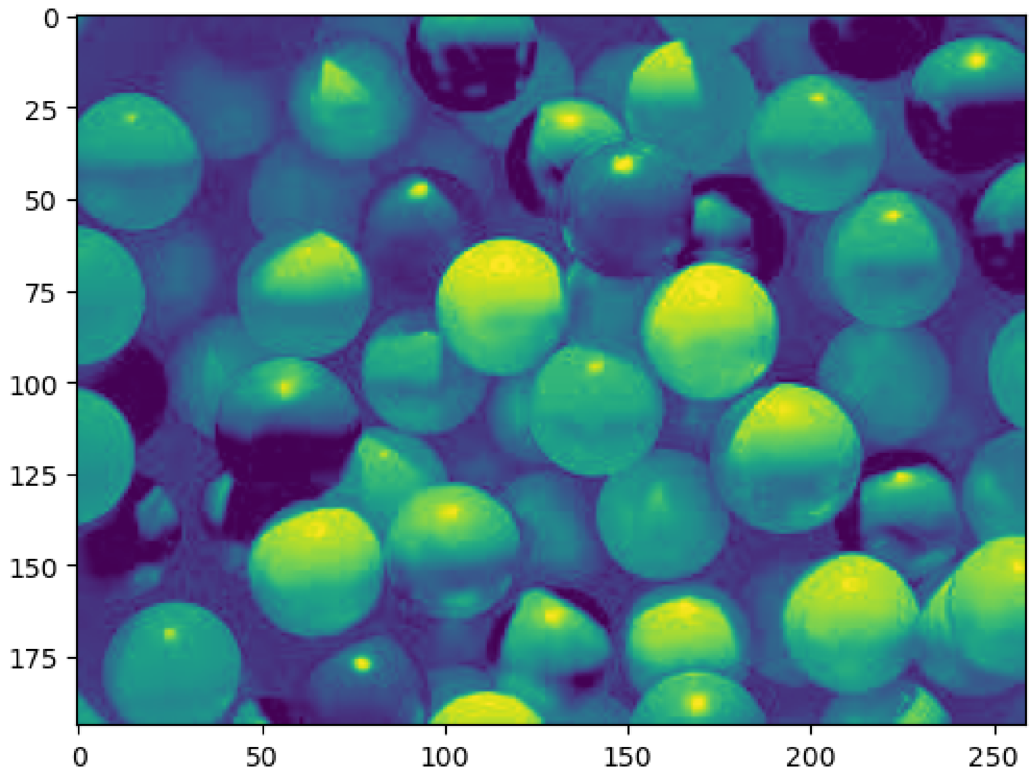
[30]:
```
# Show only the Red channel (channel index 0)
# change color
# R-0 G-1 B-2
plt.imshow(Image1_red[:,:,0])
```

[30]:
```
<matplotlib.image.AxesImage at 0x27e48472c00>
```



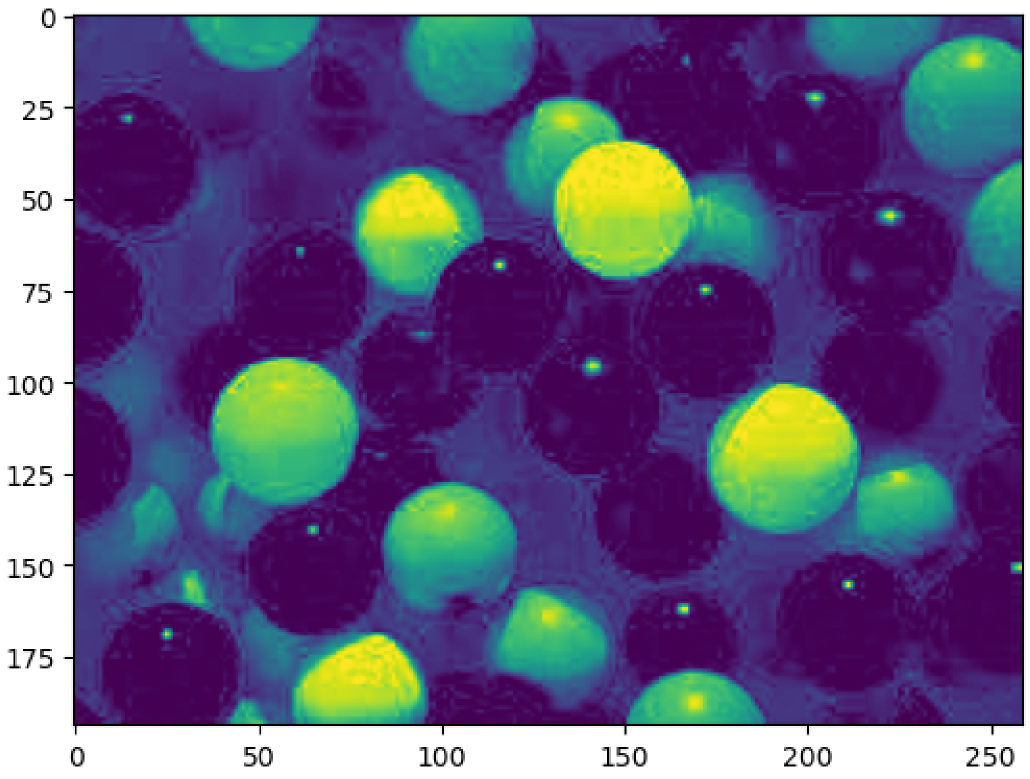
[31]:
```
# Show only the Green channel (channel index 1)
# change color
# R-0 G-1 B-2
plt.imshow(Image1_red[:,:,1])
```

[31]:
```
<matplotlib.image.AxesImage at 0x27e48622d20>
```



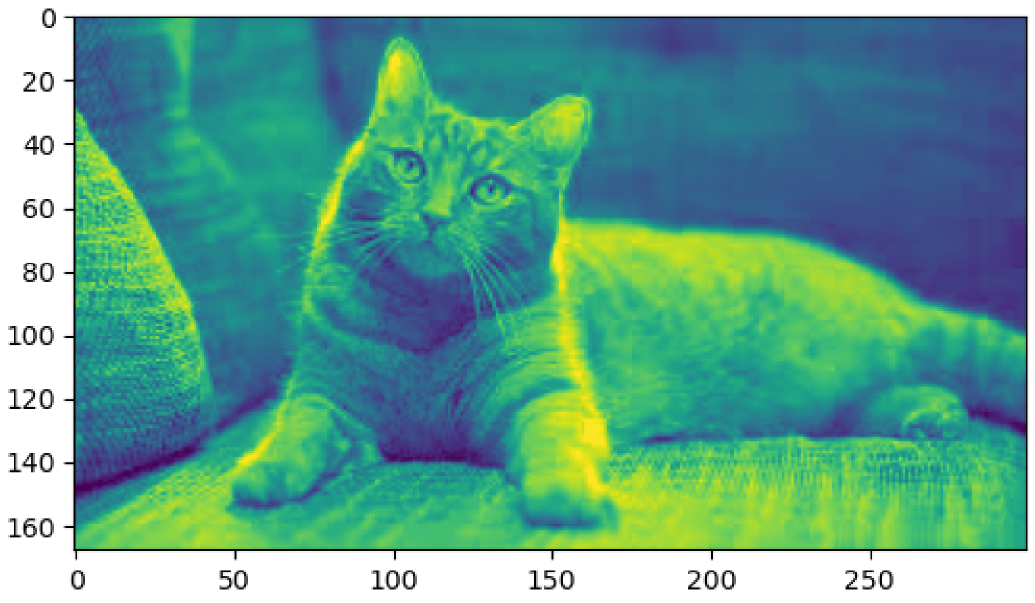
[32]:
```
# Show only the Blue channel (channel index 2)
# change color
# R-0 G-1 B-2
plt.imshow(Image1_red[:,:,2])
```

[32]:
```
<matplotlib.image.AxesImage at 0x27e489484d0>
```



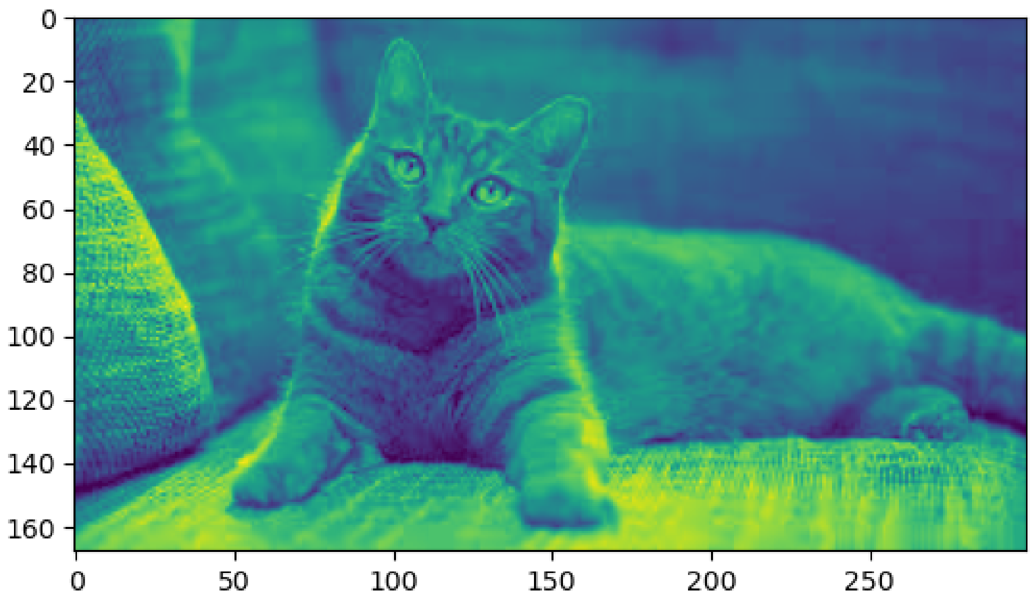
[33]:
```
# change color
# R-0 G-1 B-2
plt.imshow(Image2_red[:,:,0])
```

[33]:
```
<matplotlib.image.AxesImage at 0x27e48684800>
```



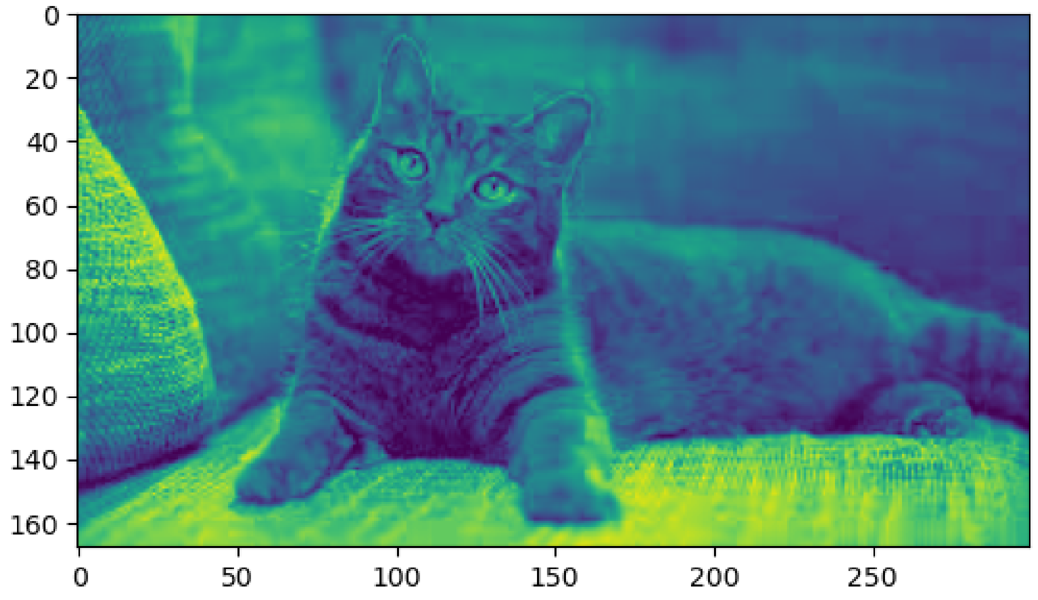
```
[35]: # change color
      # R-0 G-1 B-2
      plt.imshow(Image2_red[:,:,1])
```

```
[35]: <matplotlib.image.AxesImage at 0x27e48f1cda0>
```



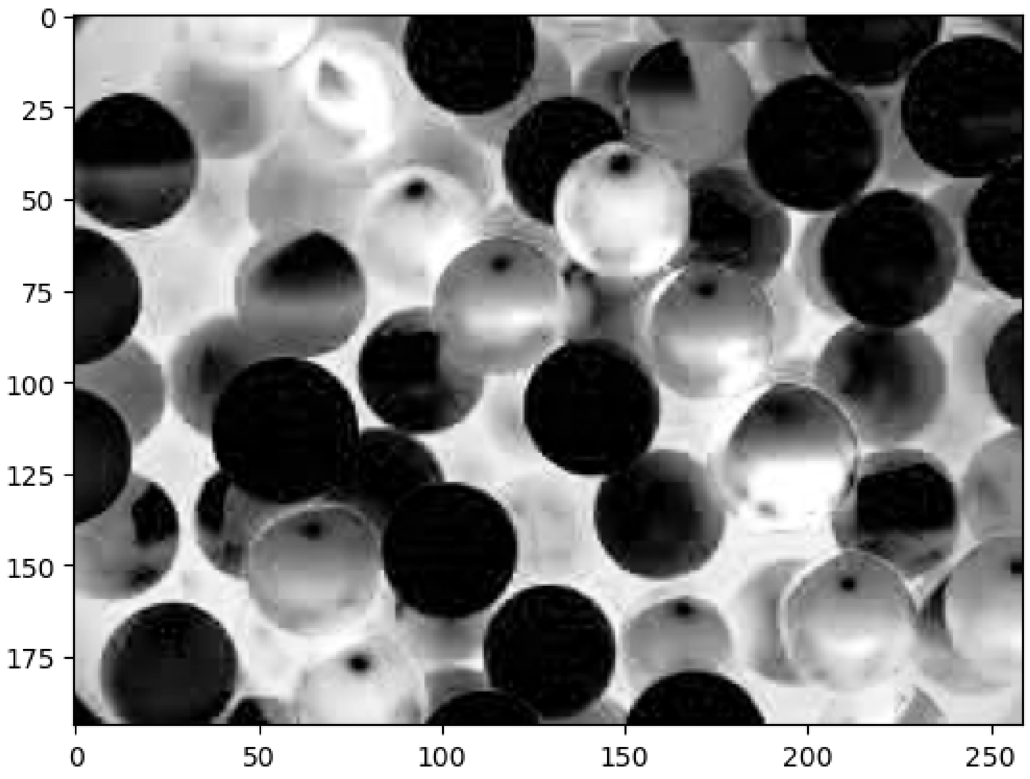
[36]:
```
# change color
# R-0 G-1 B-2
plt.imshow(Image2_red[:,:,2])
```

[36]:
```
<matplotlib.image.AxesImage at 0x27e48320140>
```



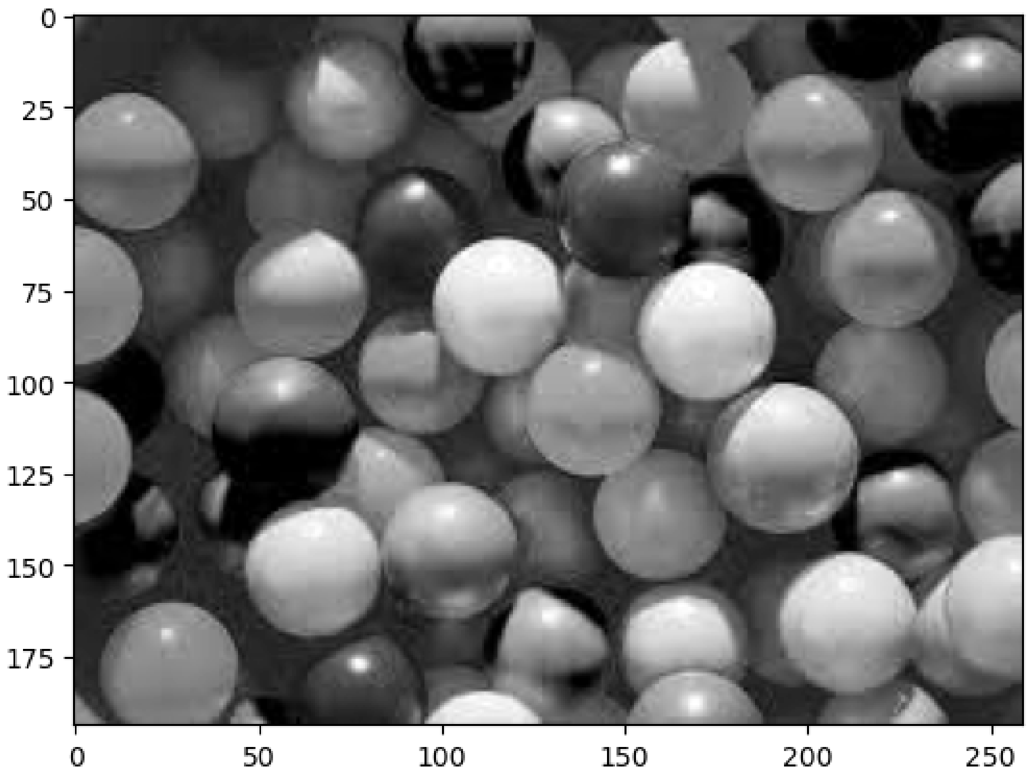
[37]:
```
# Visualize red channel using grayscale color map (makes it easier to␣
 ↪understand brightness)
plt.imshow(Image1_red[:,:,0], cmap = 'Greys')
```

[37]:
```
<matplotlib.image.AxesImage at 0x27e483b6570>
```



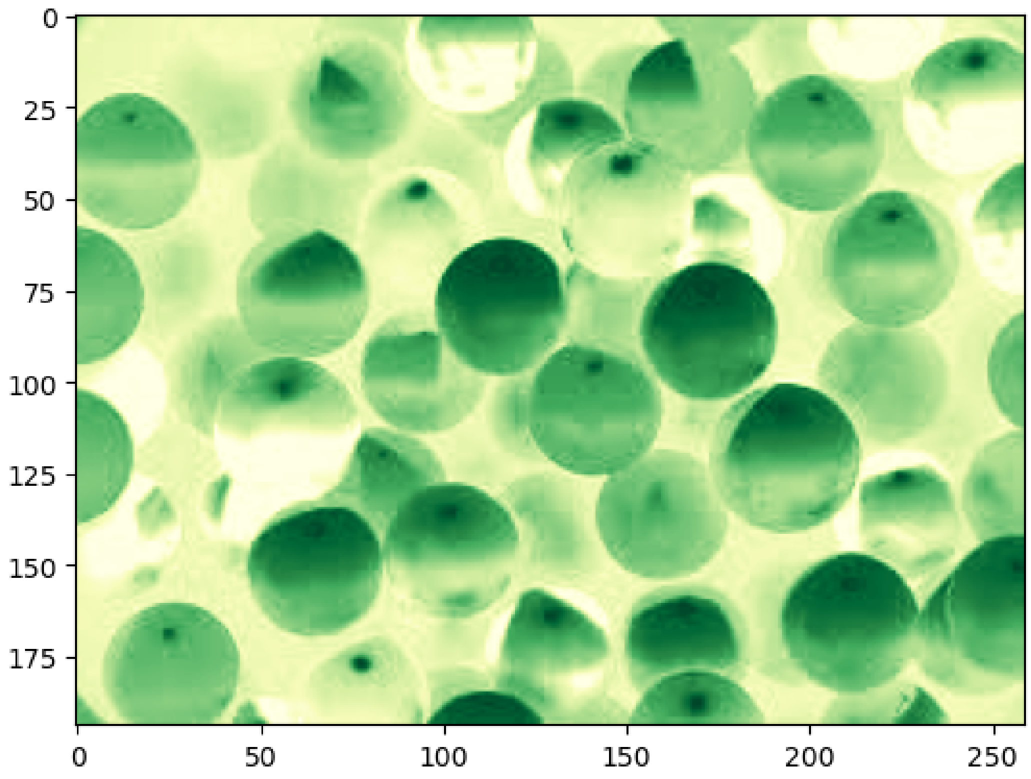
```
[38]: plt.imshow(Image1_red[:,:,1], cmap = 'grey')
```

```
[38]: <matplotlib.image.AxesImage at 0x27e47d534a0>
```



```
[39]: # Try a different colormap for fun (e.g., Yellow-Green)
      plt.imshow(Image1_red[:,:,1], cmap = 'YlGn')
```

```
[39]: <matplotlib.image.AxesImage at 0x27e471033b0>
```



```
[41]: Image1_red[:,:,0]
```

```
[41]: array([[254, 237, 206, …, 197, 192, 184],
             [244, 223, 187, …, 205, 199, 192],
             [226, 199, 159, …, 210, 205, 199],
             …,
             [253, 249, 253, …,  90,  91,  94],
             [255, 255, 255, …,  91,  91,  92],
             [250, 254, 255, …,  91,  91,  92]], dtype=uint8)
```

```
[42]: Image1_red[:,:,1]
```

```
[42]: array([[126, 112,  91, …,  57,  56,  54],
             [116, 102,  82, …,  62,  60,  59],
             [101,  88,  71, …,  63,  62,  60],
             …,
             [138, 145, 150, …,  55,  56,  58],
             [139, 146, 146, …,  54,  54,  55],
             [133, 141, 143, …,  54,  54,  55]], dtype=uint8)
```

```
[ ]:
```