Getting black plots with plt.imshow after multiplying image array by a scalar

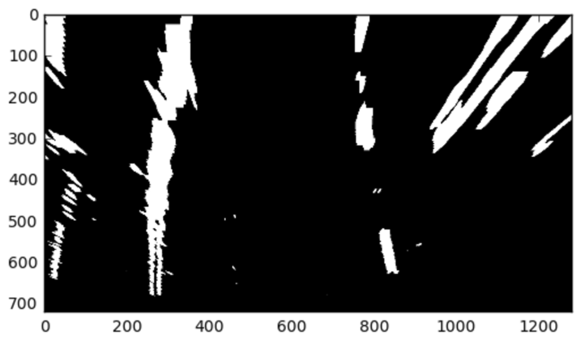
So I am a bit confused as to why this is happening.

I have a binary image:

```
In [80]: plt.imshow(binary_warped, cmap = 'gray')
         print(binary_warped[0:1])
         binary_warped.shape

         [[ 1.  1.  1. ...,  0.  0.  1.]]

Out[80]: (720, 1280)
```

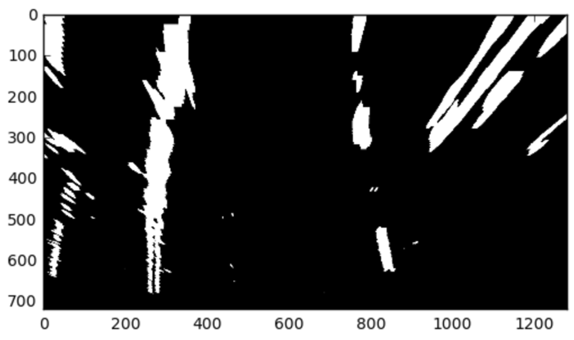


Now I want to convert this binary image into RGB space, so therefore I use the `dstack` function to concatenate the 3rd axis

```
In [82]: out_image = np.dstack((binary_warped, binary_warped, binary_warped))
         print(out_image.shape)
         print(out_image[0:1])
         plt.imshow(out_image)

         (720, 1280, 3)
         [[[ 1.  1.  1.]
           [ 1.  1.  1.]
           [ 1.  1.  1.]
           ...,
           [ 0.  0.  0.]
           [ 0.  0.  0.]
           [ 1.  1.  1.]]]

Out[82]: <matplotlib.image.AxesImage at 0x14a0fb550>
```

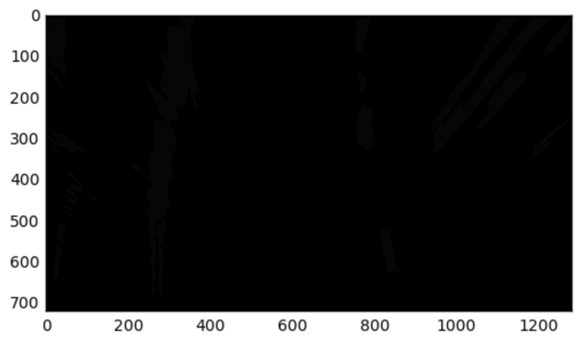


Everything works fine so far, but now I have to multiply the `out_image` array by `255` to reflect white in RGB space, and this is where the problem occurs everything turns black

```
In [83]: out_image = np.dstack((binary_warped, binary_warped, binary_warped)) * 255
         print(out_image.shape)
         print(out_image[0:1])
         plt.imshow(out_image)

         (720, 1280, 3)
         [[[ 255.  255.  255.]
           [ 255.  255.  255.]
           [ 255.  255.  255.]
           ...,
           [   0.    0.    0.]
           [   0.    0.    0.]
           [ 255.  255.  255.]]]

Out[83]: <matplotlib.image.AxesImage at 0x1522b9a58>
```



But if I plot another random image, everything is fine so what is happening here, I've also played around with the `cmap` but regardless of what kind of `cmap` I use it always turns out to be black when multiplied by `255`

Any ideas?

python matplotlib

asked 12 hours ago
YellowPillow
260 3 16

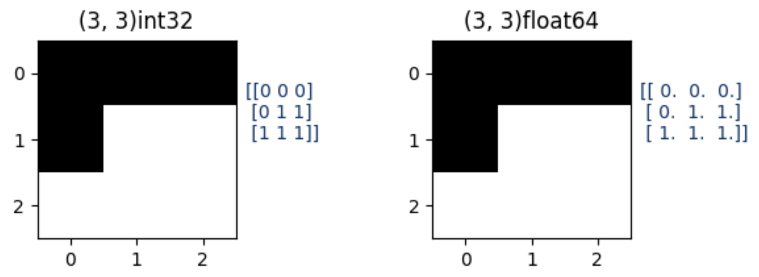
When I use `plt.imshow(...)` for RGB images, I've used float values between 0 and 1 in each of the RGB channels. In fact, its given me errors when I don't do it that way. It seems to be plotting it that way in your second image with black and white from 0 to 1. If you adjust one of the other channels I would imagine that you would see the other colors. So, they question comes down to whether or not you need to have it mapped to 8-bit integers or not. – tmwilson26 11 hours ago

1 Answer

The solution for the problem in the question would be not to multiply the array with `255`.
The other option is to reduce the datatype of the image to unsigned int8, `out_image = out_image.astype(np.uint8)`

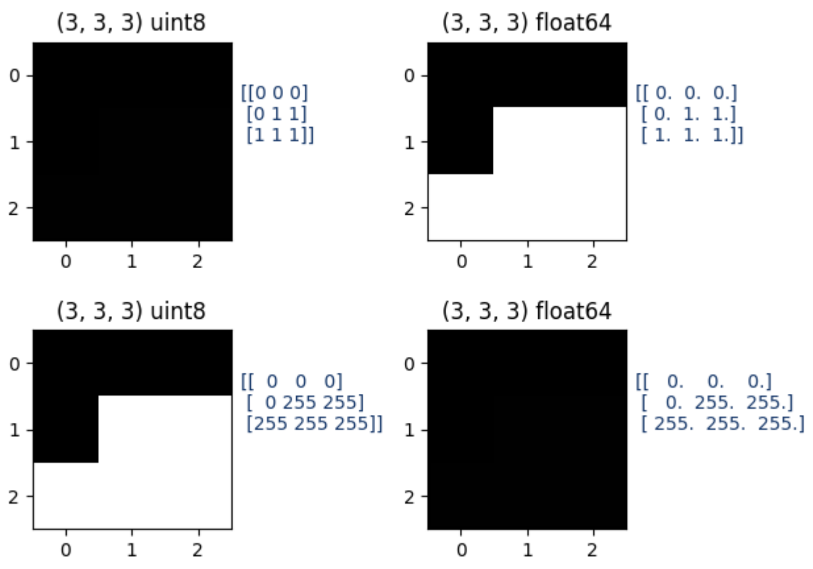
Let me explain why:

A single channel image can have arbitrary values and datatype. The color will be determined by the colormap to be used, and if required, normalized to a certain range.



In contrast, 3 channel RGB arrays are assumed by `imshow` to be in two ranges, `[0., 1.]` or `[0,255]`. ("3-dimensional arrays must be of dtype unsigned byte, unsigned short, float32 or float64").
The range to use will be selected by the datatype of the array:

1. A **float** array should be in the **`[0., 1.]`** range,
2. an **integer** array should be in the range **`[0,255]`**. Also note that integer arrays must be of datatype int8 and not int32.



As can be seen in the RGB case, an integer array in the range `[0,1]` stays black, as well as a float array of range `[0., 255.]` .

edited 7 hours ago

answered 11 hours ago
ImportanceOfBeingErnest
7,064 1 5 37

Did you mean `as.type(numpy.uint8)` ? Because casting it to `int8` didn't work for me – YellowPillow 7 hours ago

Sorry, my mistake, I meant `.astype(np.uint8)` . I corrected the answer text. – ImportanceOfBeingErnest 7 hours ago

Thanks you saved my day :D – YellowPillow 6 hours ago