How to Access and Change Color Channels using PIL?

Asked 11 months ago Active 6 months ago Viewed 5k times

4

1

I'm trying to access the RGB color channels of an image using `PIL` , and then change the color intensity of the color channel of the entire image at once.

When I say RGB color channels, here is an online example.

I don't understand if this has to be done on a pixel by pixel basis or not.

I image the logic of the code would look like this:

```
import PIL
from PIL import Image
image=Image.open("my_pic.gif")
image=image.convert('RGB')
# made up function
channel_values = image.get_channel_values()
#channel_values = i.e. (50, 100, 50)
# do some math function to transform channel_values
channel_values = (95,125,75)
image.update_channel_value('R',channel_values)
display(image.getchannel('R'))
```

This answer is the only one that comes close, but it is way too complicated for what I'm trying to do.

I've searched the PIL docs, etc. for a couple of hours but can't seem to get anywhere.

Here's how far I've gotten:

```
import PIL
from PIL import Image
image=Image.open("my_pic.gif")
image=image.convert('RGB')
display(image.getchannel('R'))
```

The problem is the `image.getchannel()` only returns a grey/black & white image.

I not only want to access the color channel value, I want to change it too.

python-imaging-library Edit tags

edited Dec 13 '19 at 11:13

asked Dec 13 '19 at 10:24

Python_Learner_DK

1,165 1 16 33

Can you please further elaborate on what do you mean with "change the intensity"? What do you actually want to do? Manipulate single pixels? Manipulate all pixels at once? – HansHirse Dec 13 '19 at 10:29

@HansHirse, clarified Q per your request. This is for the entire image/all pixels, not just a single pixel. – Python_Learner_DK Dec 13 '19 at 10:32

It's still not clear, HOW you want to change the intensity. Do you want to add value `2` ? Do you want to add some "pattern"? Do you need to pay attention to clipping? Maybe, just add some sample input image and the desired output. – HansHirse Dec 13 '19 at 10:39

@HansHirse updated Q again per your feedback, thank you for helping me ask better questions. – Python_Learner_DK Dec 13 '19 at 11:13

If your image is 20px wide and 10px tall, it will have 200 pixels, i.e. 20x10. Each pixel will have a red, a green and a blue value. So you will have 200 red values, 200 green values and 200 blue values. If you multiply all the red values by 1.1, the reds will be come 10% more prominent in the image. If you multiply all the green values by 0.8, all the greens will be come 20% less prominent. What sort of thing did you want to do? – Mark Setchell Dec 13 '19 at 11:20

1 Answer

Active | Oldest | Votes

6

If, for example, you want to multiply all pixels in the red channel by 1.2, and all the green pixels by 0.9, you have several options....

Split the image into Red, Green and Blue channels, upscale the Red channel and recombine:

```
from PIL import Image
im = Image.open('image.jpg').convert('RGB')

# Split into 3 channels
r, g, b = im.split()

# Increase Reds
r = r.point(lambda i: i * 1.2)

# Decrease Greens
g = g.point(lambda i: i * 0.9)

# Recombine back to RGB image
result = Image.merge('RGB', (r, g, b))

result.save('result.png')
```

See *"Processing Individual Bands"* here.

Or, use a matrix to transform the channels:

```
from PIL import Image

# Open image
im = Image.open('image.jpg')

# Make transform matrix, to multiply R by 1.1, G by 0.9 and leave B unchanged
# newRed   = 1.1*oldRed  +  0*oldGreen    +  0*oldBlue  + constant
# newGreen = 0*oldRed    +  0.9*OldGreen  +  0*OldBlue  + constant
# newBlue  = 0*oldRed    +  0*OldGreen    +  1*OldBlue  + constant
Matrix = ( 1.1,   0,  0, 0,
           0,   0.9,  0, 0,
           0,     0,  1, 0)

# Apply transform and save
```

```
im = im.convert("RGB", Matrix)
im.save('result.png')
```

Or, convert to Numpy array, do some processing and convert back to PIL Image:

```
from PIL import Image

# Open image
im = Image.open('image.jpg')

# Make into Numpy array of floats
na = np.array(im).astype(np.float)

# Multiply all red values by 1.1
na[...,0] *= 1.1

# Reduce green values
na[...,1] *= 0.9

# You may want to use "np.clip()" here to ensure you don't exceed 255

# Convert Numpy array back to PIL Image and save
pi = Image.fromarray(na.astype(np.uint8))
pi.save('result.png')
```

This option has the added benefit that the Numpy arrays can be intermixed and processed with **OpenCV**, **scikit-image**, **vips**, **wand** and other libraries to get MUCH more sophisticated processing done - morphology, granulometry, video processing, SIFT, object tracking...

edited Apr 26 at 8:38

answered Dec 13 '19 at 13:46

Mark Setchell
133k 18 153 275