Accessing individual pixels

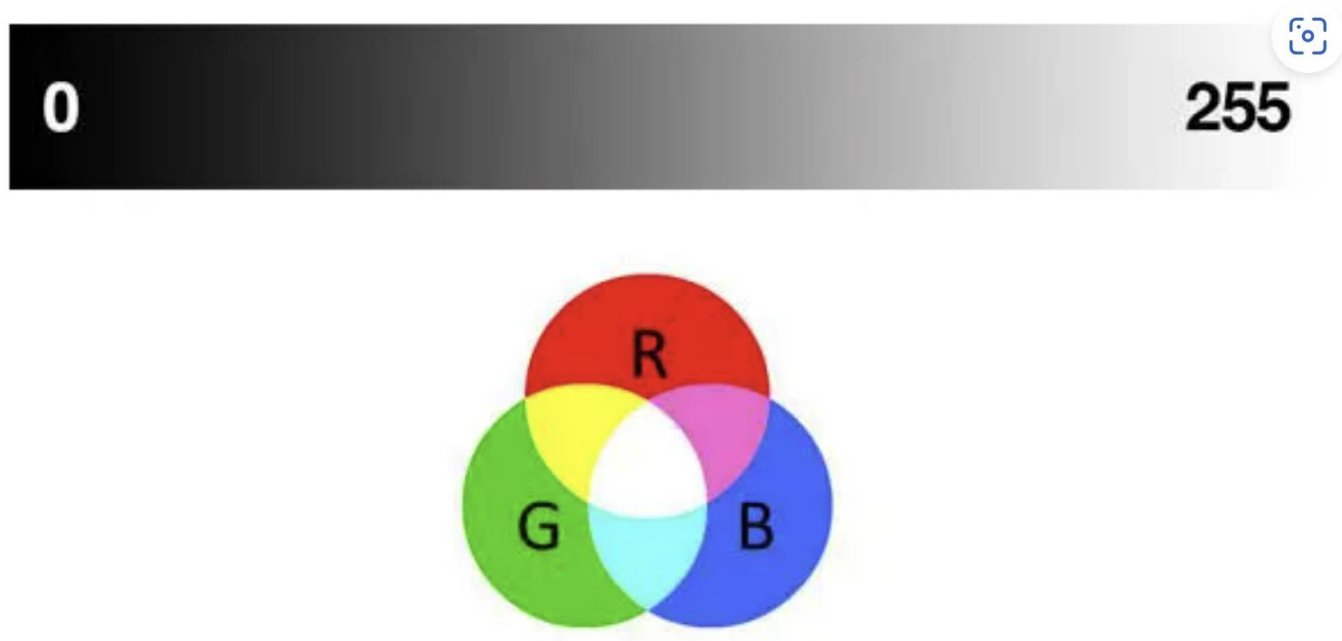

Figure 2: *Top*: grayscale gradient where brighter pixels are closer to 255 and darker pixels are closer to 0. *Bottom*: RGB venn diagram where brighter pixels are closer to the center.

> *What is a pixel?*

All images consist of pixels which are the raw building blocks of images. Images are made of pixels in a grid. A 640 x 480 image has 640 columns (the width) and 480 rows (the height). There are `640 * 480 = 307200` pixels in an image with those dimensions.

Each pixel in a grayscale image has a value representing the shade of gray. In OpenCV, there are 256 shades of gray — from 0 to 255. So a grayscale image would have a grayscale value associated with each pixel.

Pixels in a color image have additional information. There are several color spaces that you'll soon become familiar with as you learn about image processing. For simplicity let's only consider the RGB color space.

In OpenCV color images in the RGB (Red, Green, Blue) color space have a 3-tuple associated with each pixel: `(B, G, R)` .

Notice the ordering is BGR rather than RGB. This is because when OpenCV was first being developed many years ago the standard was BGR ordering. Over the years, the standard has now become RGB but OpenCV still maintains this “legacy” BGR ordering to ensure no existing code breaks.

Each value in the BGR 3-tuple has a range of `[0, 255]` . How many color possibilities are there for each pixel in an RGB image in OpenCV? That’s easy: `256 * 256 * 256 = 16777216` .

Resizing images

Resizing images is important for a number of reasons. First, you might want to resize a large image to fit on your screen. Image processing is also faster on smaller images because there are fewer pixels to process. In the case of deep learning, we often resize images, ignoring aspect ratio, so that the volume fits into a network which requires that an image be square and of a certain dimension.

```
# resize the image to 200x200px, ignoring aspect ratio
resized = cv2.resize(image, (200, 200))
cv2.imshow("Fixed Resizing", resized)
cv2.waitKey(0)
```