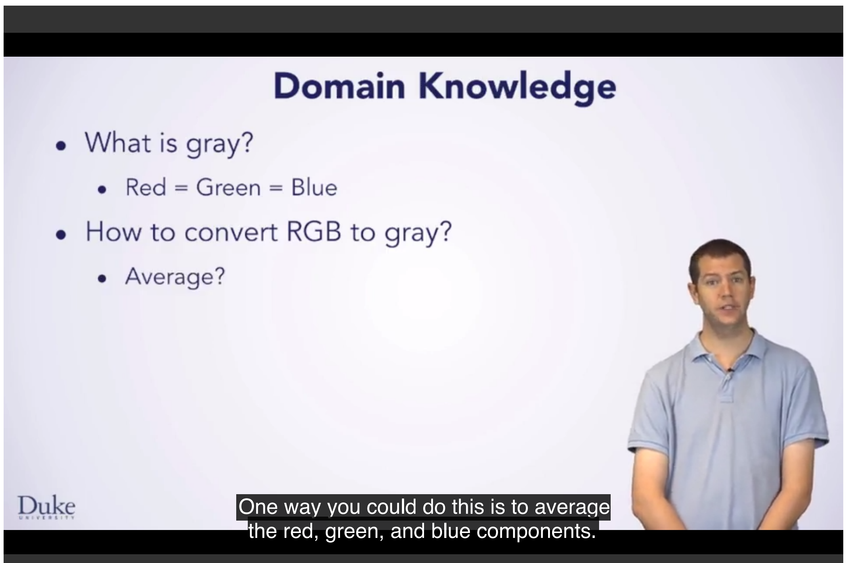
Walkthrough

Batch Grayscale Images

Batch Grayscale: Converting Many Files 2 min
Grayscale Algorithm: Seven Step Approach 4 min
Image Iterable in BlueJ: Grayscale 5 min
Batch Processing Grayscale 3 min
Saving Images with New Names 11 min
Batch Grayscale Summary: Converting Many Files 1 min
Programming Exercise: Batch Grayscale and Image Inversion 10 min
Quiz: Batch Grayscale Images 6 questions



Grayscale Algorithm: Seven Step Approach

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript English

0:03
Welcome back. You're currently learning how to convert a bunch of images to gray-scale. An important component of that process is the ability to convert one image to gray-scale. In particular, you want to take a color image and produce an image that looks like it, but only uses shades of gray, not any colors. As with all programming problems, the way you want to approach this problem is with the 7 step approach you learned previously. The first step is to work an instance by hand. As always, we must work with a small, manageable problem size. Here we picked a 2x2 image to work with. We're going to want to make a 2x2 image for the output, but how do you figure out what shade of gray to use for this pixel? You need the domain knowledge before you can proceed. In this particular problem, the knowledge you need is about colors or graphics.

0:53
The first thing we need to know is, what precisely is a shade of gray? A color is a shade of gray if its red, blue, and green components are all the same. However, this knowledge by itself is not sufficient to tell you how to come up with the shade of gray for a particular color. Just that the result needs to have their red, blue, and green all the same.

1:14
One way you could do this is to average the red, green, and blue components. Or you might decide you want a weighted average, because the human eye does not perceive all colors in the same way. There could be more complex alternatives. However, just taking the average works pretty well and is simple.

1:30
Now you have the domain knowledge required to do this problem yourself. You can look at the RGB for a pixel, compute the average, and color in the output pixel appropriately. Then you would go through looking at the RGB values for each input pixel, computing their average and coloring in the output pixel accordingly. Once you have colored in all of the pixels, you've worked an instance of the problem yourself and are done with Step 1.

1:57
The next thing you need to do, is write down exactly what you just did. I started with the image that I wanted, which we called inImage. Then, I made another image of the same size, which we called outImage. I computed (255+0+0)/3, which is 83, and I made the first pixel of outImage have red, green, and blue values at 83.

2:22
And then I went through each other pixel computing the average of the R, G, and B and coloring the output pixel accordingly.

2:31
One we finished this we had these ten steps, that we used to solve this particular instance of the problem. Now, you are ready to move to Step 3, and look for patterns and repetition. You can see that we are doing very similar things to each pixel, but they are not quite the same. We need to find the patterns in the numbers to generalize these steps to any image. Let us look at the particular numbers we need to generalize. Why did we use 255 here and 0 here. These numbers were all the corresponding pixel in inImage's red component. What about these numbers? Similarly, these were the green component of the corresponding pixel in inImage.

3:16
Lastly, these numbers were the blue component of the corresponding pixel.

3:21
Next, you should give a name to the result of this math. It won't always be these particular numbers, and you will want to be able to refer to it precisely. We'll call it average.

3:32
Okay, now that we have thought that through, you can write the general algorithm. Notice how we thought about what we do for each pixel and wrote down general steps. Now we can write this in terms of steps to do to each pixel in the output image. It will work for any size image, with any colors.

3:50
The last thing you should do before you write code is to test your generalized algorithm out on another small input. Here is a small image and the RGB values for each pixel. Take a moment to execute the algorithm and see if you get the correct answer.

4:05
Yes the answer is right, so you are ready to implement it in code.

Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt
Lecture Slides pdf

Would you like to help us translate the transcript and subtitles into additional languages?