Problem Set 1: Images as Functions

(really, arrays or matrices of numbers)

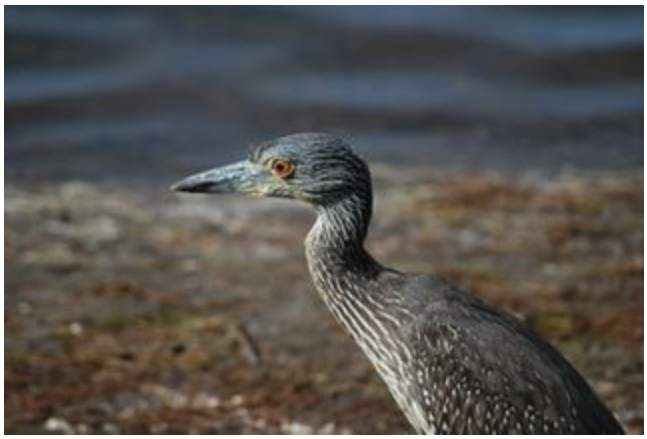
Christopher Owens

gtg819x

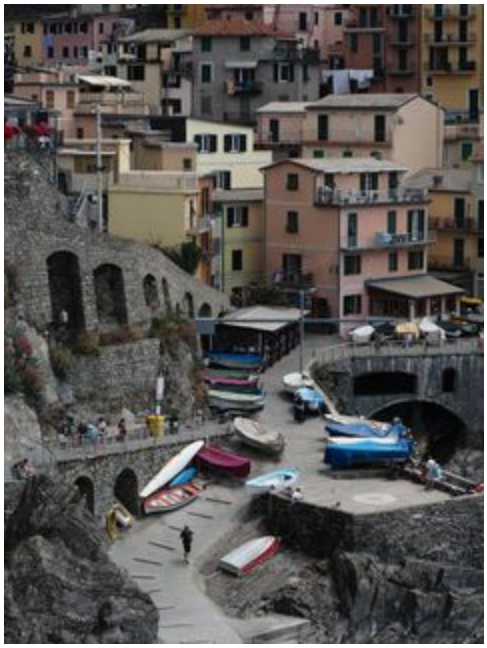
901821872

Questions

1. Input images
    a. Find two interesting images to use.

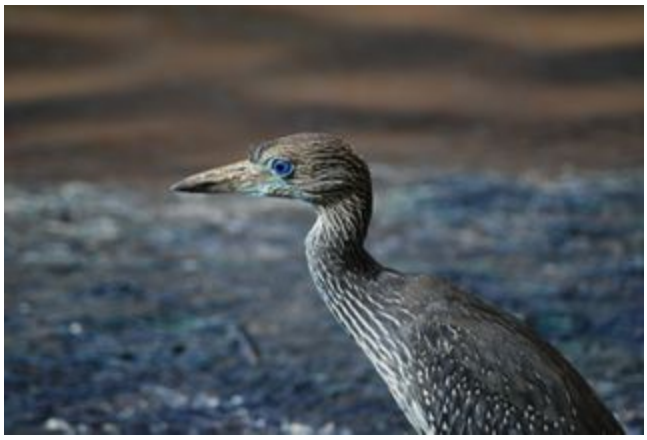


ps1-1-a-1



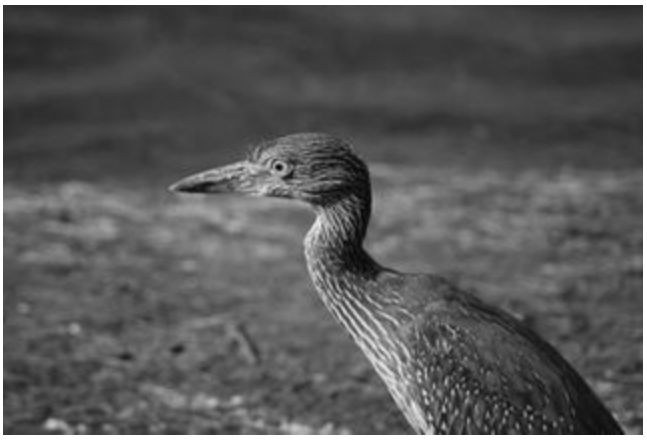
ps1-1-a-2

2. Color planes
    a. Swap the red and blue pixels of image 1



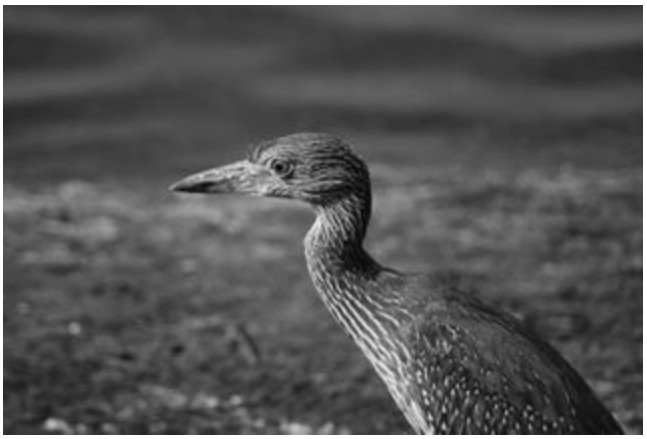
ps1-2-a-1

b. Create a monochrome image by selecting the green channel of image 1



ps1-2-b-1

c. Create a monochrome image by selecting the red channel of image 1



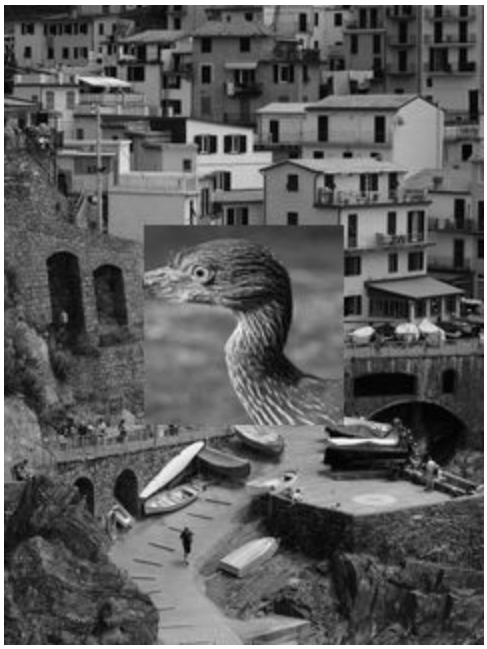
ps1-2-c-1

d. Which looks more like what you'd expect a monochrome image to look like? Would you expect a computer vision algorithm to work on one better than the other?

   i. ps1-2-b-1 has more detail than ps1-2-c-1, though both look similar to what I'd expect a monochrome image to look like. I would expect a computer vision algorithm to work better on ps1-2-b-1 that ps1-2-c-1 due to the it having more pronounced details.

3. Replacement of pixels

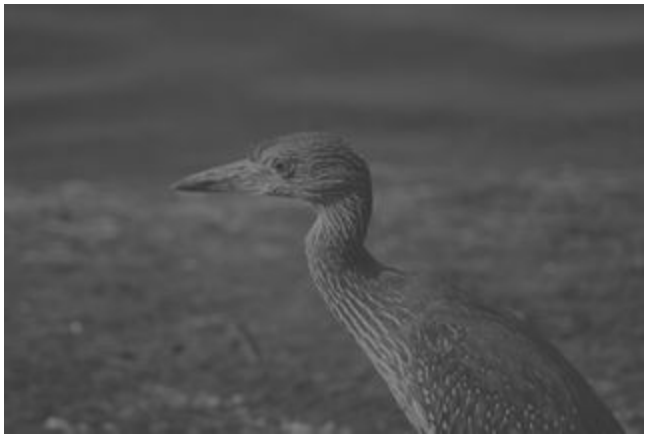
   a. Take the center square region of 100x100 pixels of monochrome version of image 1 and insert them into the center of a monochrome version of image 2



ps1-3-a-1

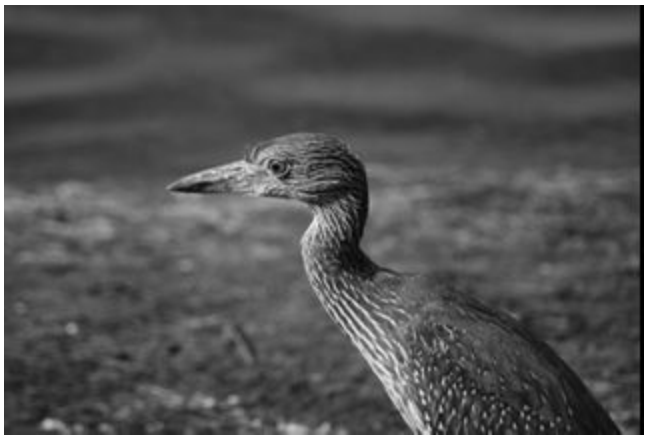
4. Arithmetic and Geometric operations
    a. What is the min and max of the pixel values of img1_green? What is the mean? What is the standard deviation? And how did you compute these?
        i. min: 2
        ii. max: 244
        iii. mean: 80.7904360465
        iv. stand dev: 26.8386556593
        v. I computed them using the respective numpy methods
            1. min = np.min(image)
            2. max = np.max(image)
            3. mean = np.mean(image)
            4. stand_dev = np.std(image)

    b. Subtract the mean from all pixels, then divide by standard deviation, then multiply by 10 (if your image is 0 to 255) or by 0.05 (if your image ranges from 0.0 to 1.0). Now add the mean back in.



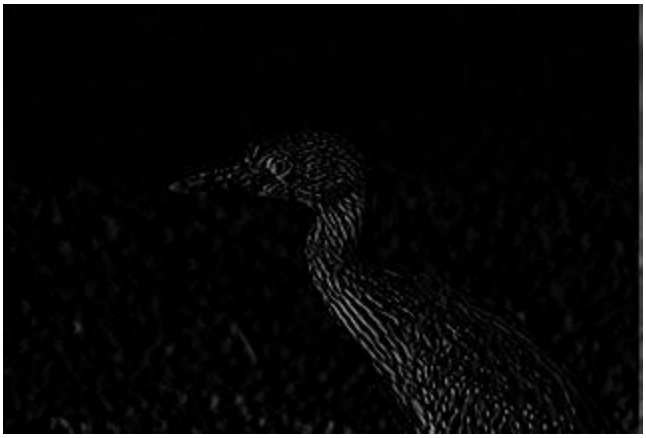
ps1-4-b-1

c. Shift img1_green to the left by 2 pixels.



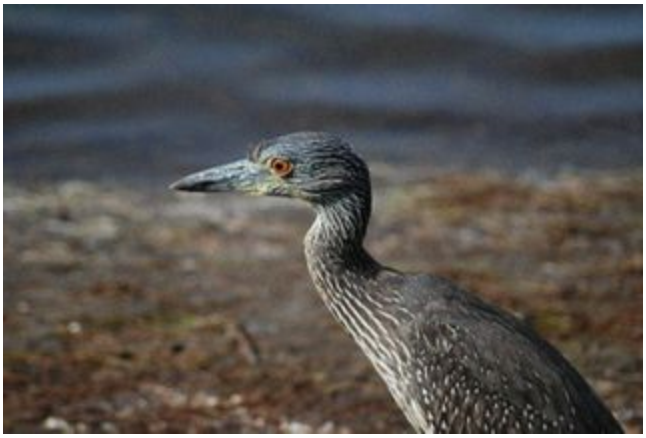
ps1-4-c-1

d. Subtract the shifted version of img1_green from the original img1_green, and save the difference image.
What do negative pixel values mean anyways?

i. 

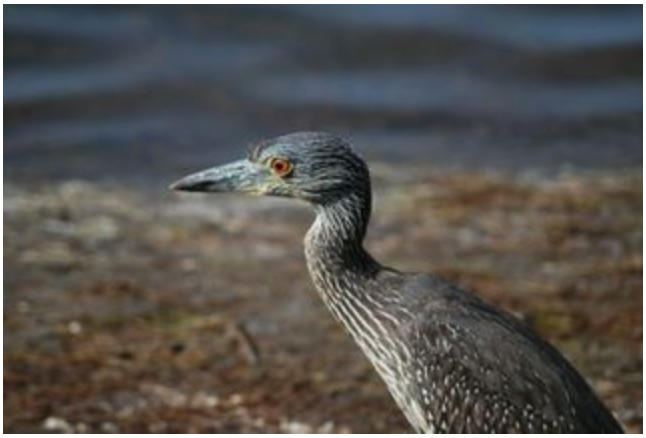
ps1-4-d-1

ii. Negative pixels imply an intensity level less than the valid range. This would mean we either didn't match our range up to reality, or we applied math to our image function pushing the values outside of the valid range. In this case, we've run into the later meaning the values are invalid. So, we 'round' them up into the valid range.

5. Noise
    a. Take the original colored image (image 1) and start adding Gaussian noise to the pixels in the green channel. Increase sigma until the noise is somewhat visible. What is the value of sigma you had to use?
        i. 

        ps1-5-a-1

        ii. Sigma = 4

    b. Now, instead add that amount of noise to the blue channel. Which looks better? Why?
        i. 

        ps1-5-b-1

        ii. The blue channel looks better. It looks like there is much less noise. I assume this is due to the large amount of blue in the image. Since the blue intensities are so much larger than the green intensities, the percentage of change is much smaller for the blue channel than the green channel.