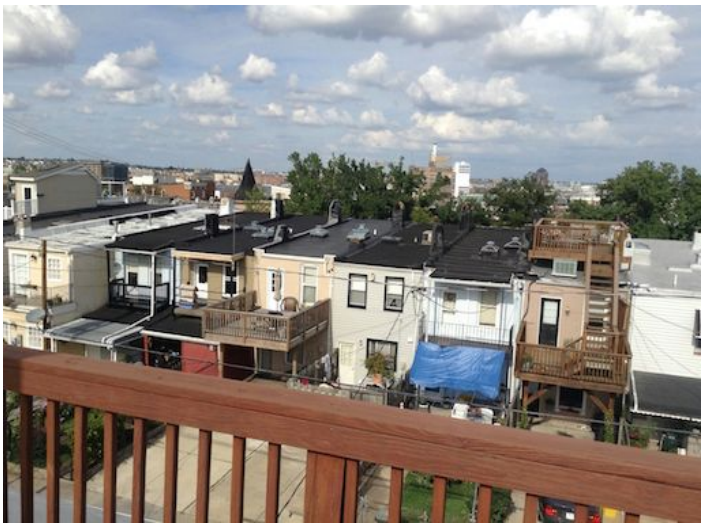
Problem Set 1: Images as Functions

(really, arrays or matrices of numbers)

Questions

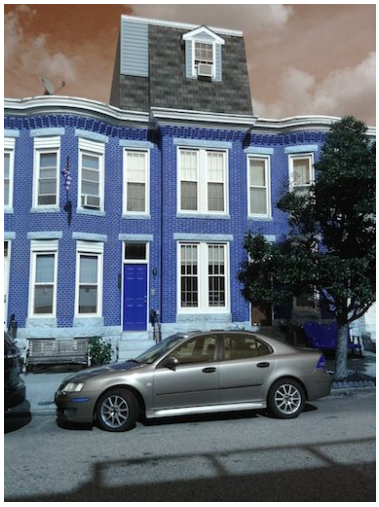
1. Input images
    a. Find two interesting images to use. They should be color, rectangular in shape (NOT square). Pick one that is wide and one tall.
    You might find some classic vision examples here. Or take your own. Make sure the image width or height does not exceed 512 pixels.
    Output: Store the two images as ps1-1-a-1.png and ps1-1-a-2.png inside the output folder

ps1-1-a-1.png



ps1-1-a-2.png

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



ps1-2-a-1.png

b. Create a monochrome image (img1_green) by selecting the green channel of image 1

ps1-2-b-1.png

c. Create a monochrome image (img1_red) by selecting the red channel of image 1

ps1-2-c-1.png

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?

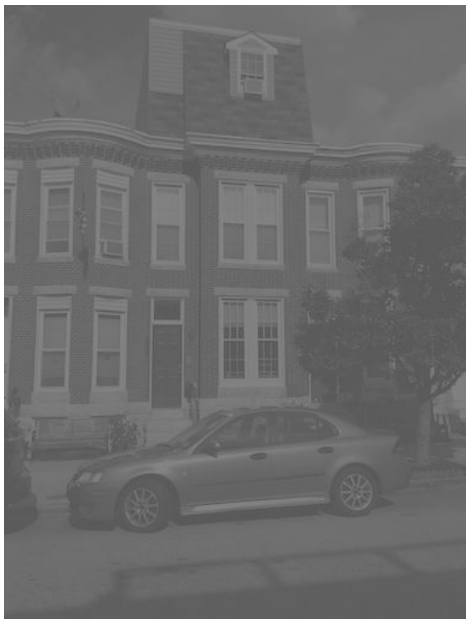
   i. I would expect the green channel to be the monochrome image. It has more contrast, and seems to show more details of the brick, doors and other items in the scene. Also, since it

has more contrast and detail, I would expect a computer vision algorithm could get more information out of the scene and work better

3. Replacement of pixels (Note: For this, use the *better* channel from 2-b/2-c as monochrome versions.)
    a. Take the ~~inner~~ 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.png

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 = 0 max = 255 mean = 107.11 std = 60.52
        ii. I used the numpy commands for each (respectively, np.min, np.max, np.mean, np.std)
    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.

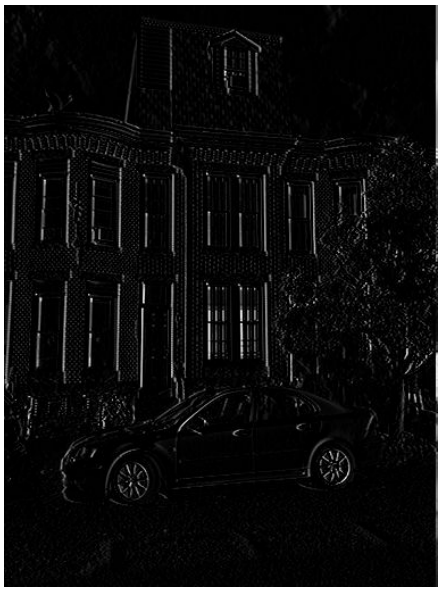


ps1-4-b-1.png

c. Shift img1_green to the left by 2 pixels.

ps1-4-c-1.png

d. Subtract the shifted version of img1_green from the original img1_green, and save the difference image.



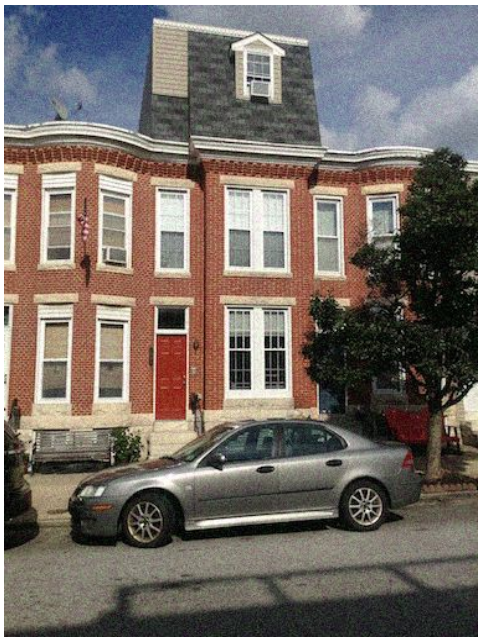
ps1-4-d-1.png

What do negative pixel values mean anyways?

The negative pixel values are a place of negative gradient. It means that (moving from left to right) the scene was a lower intensity and then moved to a higher intensity. Overall, this is an 'invalid' number given our range is 0-255. We could either scale this, or clip it to achieve different effects.

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.



ps1-5-a-1.png

What is the value of sigma you had to use?

i. I used 16 as my sigma. I could start to tell a difference at about 10, but 16 made it pretty pronounced

b. Now, instead add that amount of noise to the blue channel.

ps1-5-b-1.png

c. Which looks better? Why?

The blue channel noise definitely looks better. The difference is very subtle and is difficult to detect. My guess is that there isn't a whole lot of blue in the scene - the only place it is obvious that there is any noise is in the sky. Additionally, I would guess that the human eye is better at picking up some color channels than others.