Questions

1. Input images

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 do not exceed 512 pixels.

Output: Store the two images as ps0-1-a-1.png and ps0-1-a-2.png inside the output folder

2. Color planes

Swap the red and blue pixels of image 1

Output: Store as ps0-2-a-1.png in the output folder

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

Output: ps0-2-b-1.png

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

Output: ps0-2-c-1.png

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?

Output: Text response in report ps0_report.pdf

===> green monochrome picture. The red one seem to not have enough details on face

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

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

Output: Store the new image created as ps0-3-a-1.png

4. Arithmetic and Geometric operations

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?

Output: ==> use deviation = sqrt(quare_sum/N - mean*mean) N= width*height

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.

Output: ps0-4-b-1.png

Shift img1_green to the left by 2 pixels.

Output: ps0-4-c-1.png

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

Output: ps0-4-d-1.png (make sure that the values are legal when you write the image so that you can see all relative differences), text response: What do negative pixel values mean anyways?

====> negative pixels will eventually end up being high postive in uint_8 representation. So they look all washout to me

5. Noise

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.

Output: ps0-5-a-1.png, text response: What is the value of sigma you had to use?

====> When sigma start to be around 1/8 mean I can see visible noise

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

Output: ps0-5-b-1.png

Which looks better? Why?

Output: Text response

====> blue channel noise is less sensitive to my eyes.