**­EEL 4930/5934**

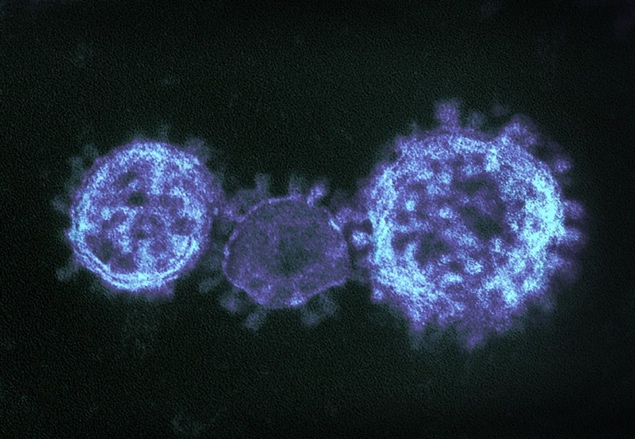
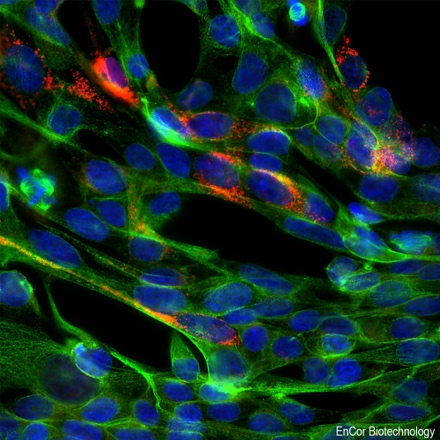
**Introduction to Biomedical Image Analysis**

**Assignment – 5**

**Due: 02/27/2024, Noon**

1. **Image Manipulation:** The field of computational image analysis continues to make major contributions to COVID-19 research and discovery. For each of the tasks below, produce a well-annotated, original MATLAB script to be turned in along with your assignment.

a.) A negative image is an image in which the lightest areas of the photographed subject appear darkest and the darkest areas appear lightest. Derivation of an image's negative requires inversion of each pixel's intensity value such that the darkest pixels (min zero) and lightest pixels (max 255) are inverted. Given this definition as well as the image analysis approach presented in Monday's lab session, generate the negative of "SarsCoV2.jpg" (below left), an image featuring viral particles captured with super-resolution microscopy. Note: MATLAB functions *rgb2gray()* and *imcomplement()* may not be used for this task. **(2 pts.)**



A close-up of several images of sars

Description automatically generated

b.) In lab we also learned how to rearrange a RGB image’s colors channels, and discussed the importance of catenating the channels in the correct (original) order when the color of each channel is representative of a particular cell structure. The above image on the right, titled "SarsCoV2\_IF.jpg", features human embryonic stem cells transfected with Sars-CoV-2 to study the effect of long-term viral infection on cell structure and function. The cell sample has been fluorescently stained for Sars-CoV-2 viral particles (red), cell cytoplasm (green), and cell nuclei (blue). Suppose that a computational scientist wants to split, normalize, and recombine the fluorescence image’s channels to produce a high-quality image for publication. Incorrect "stacking" of the channels would result in misrepresentation of their critical findings. Consider the possible mistakes the scientist could make in recombining the R, G, and B channels, and generate a RGB image for each potential combination. **(3 pts.)**

There are 5 possible mistakes the scientist could make, as there are 6 combinations of the letters R, G, and B where order matters (and only 1 of them is correct). Any one of these mistakes would result in the scientist misrepresenting findings – while the image may look okay with the switched channels, the analysis done likely depends on the colors being stained and labeled correctly.

A collage of different colored cells

Description automatically generated

1. **2-D Convolution:** 
   1. Using the provided *Two\_D\_Conv\_Example.m* code, generate a random input image and binary convolution kernel. Include the output images of both here. **(1 pt.)**A black and grey squares

      Description automatically generated
   2. Get the convolution kernel in terms of (-τ1, -τ2). **(1 pt.)**
   3. What should the final size of the convolved input image and kernel be? (X x Y) (Show work) **(1 pt.)**
   4. Show work necessary to derive the final convolved image. You may use Matlab to check your answer. **(2 pts.)**

A black and white pixelated square

Description automatically generated

1. **Convolutional Edge Filters:**
   1. Using 2-D convolution it is possible to locate areas with high gradients of intensity change which appear as edges in an image. Using *one\_sphere.m*, generate a 256x256 image with a sphere centered at (128,128), with a radius of 25, and intensity of 15. **(1 pt.)**
   2. Next, generate the following convolutional kernels in Matlab. Define two new variables that are the 2-D convolution of the sphere image with each of these kernels separately (H and V). Include the output images below. (Hint: use the ‘same’ shape argument in *conv2(A, B, shape).*) **(2 pts.)**

**A close-up of a grey square

Description automatically generated**

* 1. Now that we have the horizontal and vertical edges captured, we have to combine them to arrive at an edge mask for the sphere. To do that, run the following lines (substituting variable names for the H and V components as necessary.) What we are doing is taking the magnitude of the two components of the final answer and then showing the result with the lowest intensity pixels set to zero whereas the highest intensity pixels are set to 1. Include output image below. **(1 pt.)**

>> E = sqrt(H.\*H+V.\*V);

>> figure, imshow(E, []), title(‘Implementation of Sobel Edge Filter’)

A black square with white circles

Description automatically generated

* 1. Apply the same procedures as above with the image, ‘*cell.jpg,’* and include output below. **(1 pt.)**

A close-up of a black background

Description automatically generated

1. **Filter Concepts:**
   1. Explain the idea of “linearity” when applied to image filters. **(1 pt.)**

A linear image filter is one that obeys the mathematical properties of linearity – additivity and homogeneity. That is, if you have two images, a linear filter satisfies the following:

* + 1. The filter applied to the sum of the two images is the same as the filter applied to each image and then summed
    2. The filter applied to an image scaled up by a value “a” is the same as first applying the filter to the image and then scaling up by “a.”

It is worth noting that the more abstract properties here assume that the images’ pixels are not restricted by magnitude, like how a normal RGB image is restricted to a max value of 255 for any given pixel. However, with some (hopefully sufficient) handwaving, we are able to scale the images down to the max value through scalar multiplication by a fraction and preserve these properties while maintaining our restraint on the number of bits our image can hold.

* 1. What is an example of a non-linear image filter (Explain)? **(1 pt.)**

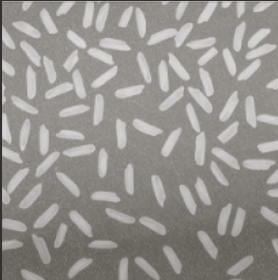
The 2x2 median filter is an example of a non-linear filter. Consider two 3x3 grayscale images:

And apply the 2x2 median filter to each image separately (with padding).

* 1. Is the Sobel edge filter, a high-pass or low-pass filter? Explain. **(1 pt.)**

**Total = 5+5+5+3 = 18 pts**

1. **Extra question:**



In the above image you can see several grains of rice on a blurry, unevenly illuminated gray background. Your goal is to perfectly segment each grain of rice in this image and then find any directional edges. This information could be useful in determining the overall orientation of the grains of rice on the imaging plane.

**Part I. Without Background Correction**

1. Use the *imhist()* command to generate a histogram of all the grayscale intensities in the image (use *imread(‘tophat.png’)* to read this built-in Matlab image). Copy the output figure below.
2. What are the peaks in this histogram? Which peak do you think corresponds to the grains of rice? Which peak(s) do you think belong to background?
3. Use the *im2bw()* command (without varying the threshold) and copy the output below. Is this a good segmentation?
4. Now use the *imbinarize()* command to threshold this image. How does this segmentation compare?

**Part II. Introducing Background Correction**

1. Describe a top-hat transform in your own words.
2. Create a block diagram detailing the steps to a top-hat transform.
3. Use the *imtophat()* command, with a ‘disk’ structuring element, on the image of the grains of rice. Include the output below. (Hint: the syntax should be like `*top\_hat\_filtered = imtophat(rice\_image, strel(‘disk’,12));*`)
4. Use the *imhist()* command to generate a histogram of the grayscale intensities in the altered image. How does this compare to the original histogram?
5. Enhance the contrast using *imadjust()* on the top-hat filtered image. Then, use *imbinarize()* to threshold the result. Include the binary image below.
6. How many grains of rice are in the image?

**Part III. Edge Detection**

1. Using your binary image of the rice, use 2D convolution to apply the following Sobel edge detection filter to your image. Include your output below.
2. What direction does this filter operate in? What would the filter kernel for edges in the opposite direction look like? Include your answer for the opposite filter kernel as well as the result of applying this kernel to your image.

**4 + 6 + 2 = 12**

**Bonus:**

Since you did so well developing your rice counting pipeline, your professor has agreed to grant you a reward. Being very clever, you pull out a chess board and ask your professor for one grain of rice on the first square followed by two grains of rice on the second square, four on the third, and so on. Your professor laughs at your silly prize suggestion and starts counting out the grains of rice you are owed. How many grains of rice does your professor now owe you? (Show work) How many bits are needed to represent this number?