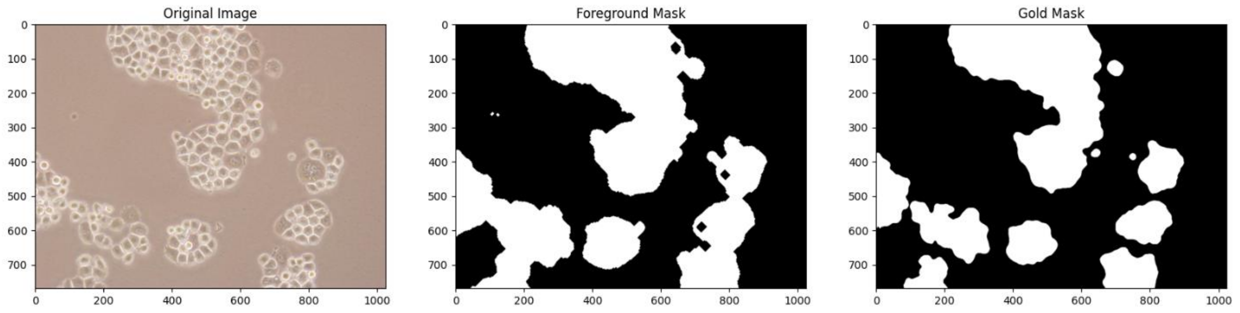
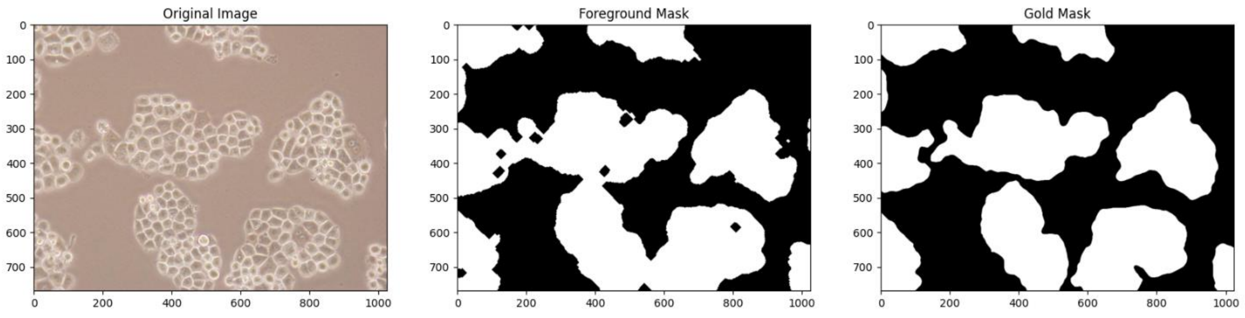
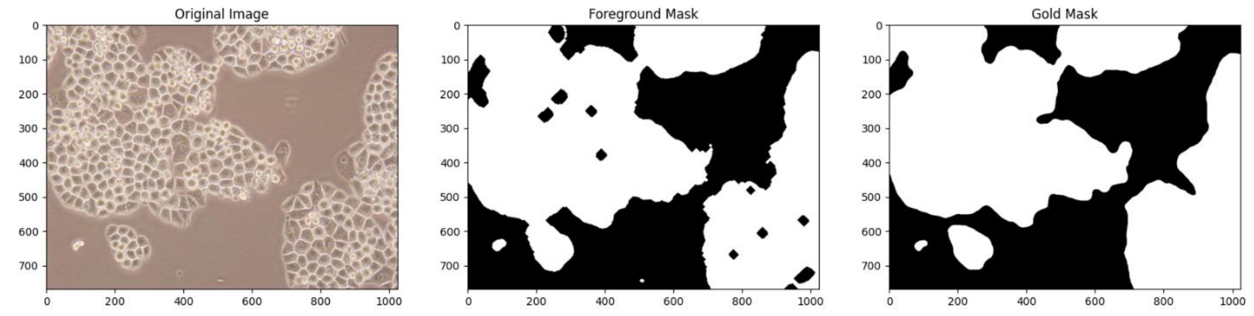
COMP 448/548 – MEDICAL IMAGE ANALYSIS

HOMEWORK #1

1) **Obtain Foreground Mask:** In this part of the assignment, I chose to modularize the functions to simply the test phase. For instance, in the “image_filtering” function, I experimented with various values for the second parameter “kernel_size”. A kernel size of 5x5 gave the best results in terms of evaluation metrics. In my application, the first function “image_filtering” takes two parameters, and an image and size of kernel. This function was used for filtering operations. It first applies Gaussian blur to the input image to reduce noise and smooth out the details. Then, it extracts the red channel from the blurred image because the edge values are much higher than the background. Next, it applies a Laplacian filter to the red channel. This filter highlights regions of rapid intensity change, which can help emphasize edges and details in the image. After applying the Laplacian filter, the function takes the absolute value of the obtained image. It combines the sharp and Laplacian images by adding two times the absolute Laplacian image to sharpen the details. The obtained image is then clipped to the range of 0-255 and converted to unsigned 8-bit integer format. Finally, it calculates the mean value of the obtained image to be used in “generate_fg_mask” function as a threshold parameter. The second function “generate_fg_mask” takes the output image from the “image_filtering” function and a “white_limit” value as input. It thresholds the input image using the white limit, which effectively separates foreground objects from the background. After thresholding, the function applies morphological opening and closing operations to the thresholded image. These operations help to smooth the boundaries of the foreground objects and remove small noise regions. The function then converts the foreground mask values to binary, where foreground pixels are represented by 1 and background pixels by 0. The resulting binary mask is returned as the output of the function.

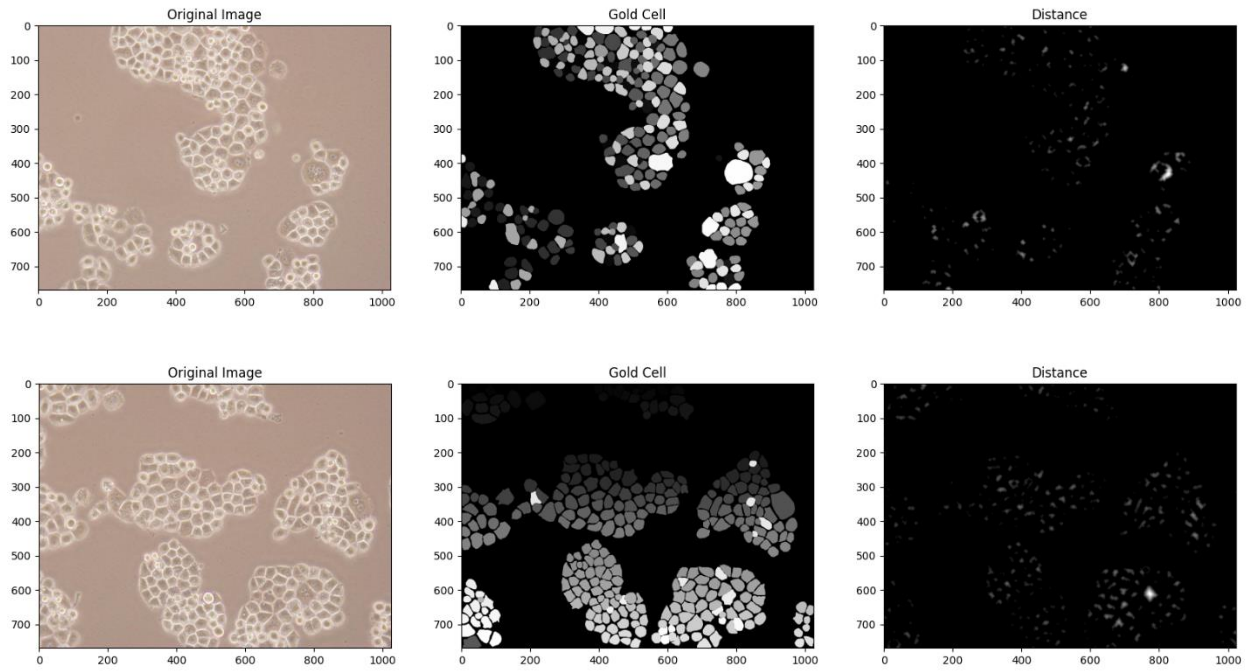


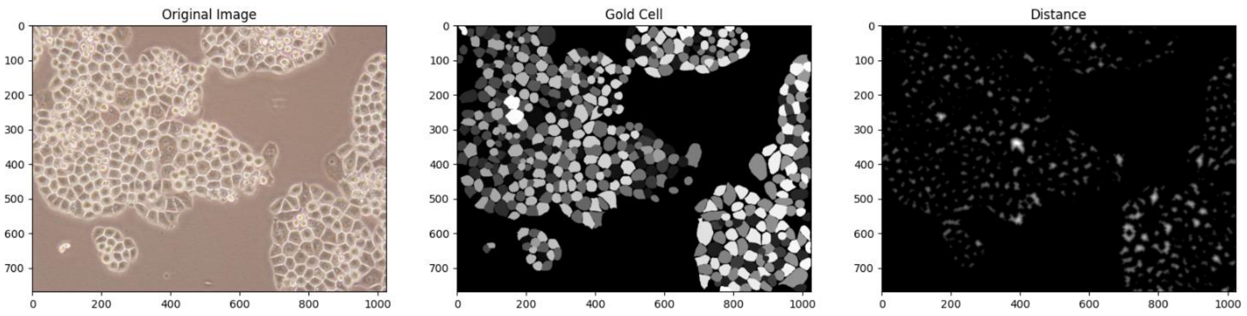




IMAGES	PRECISION	RECALL	F1 SCORE
IM 1	0.80	0.99	0.88
IM 2	0.88	0.97	0.93
IM 3	0.95	0.96	0.96

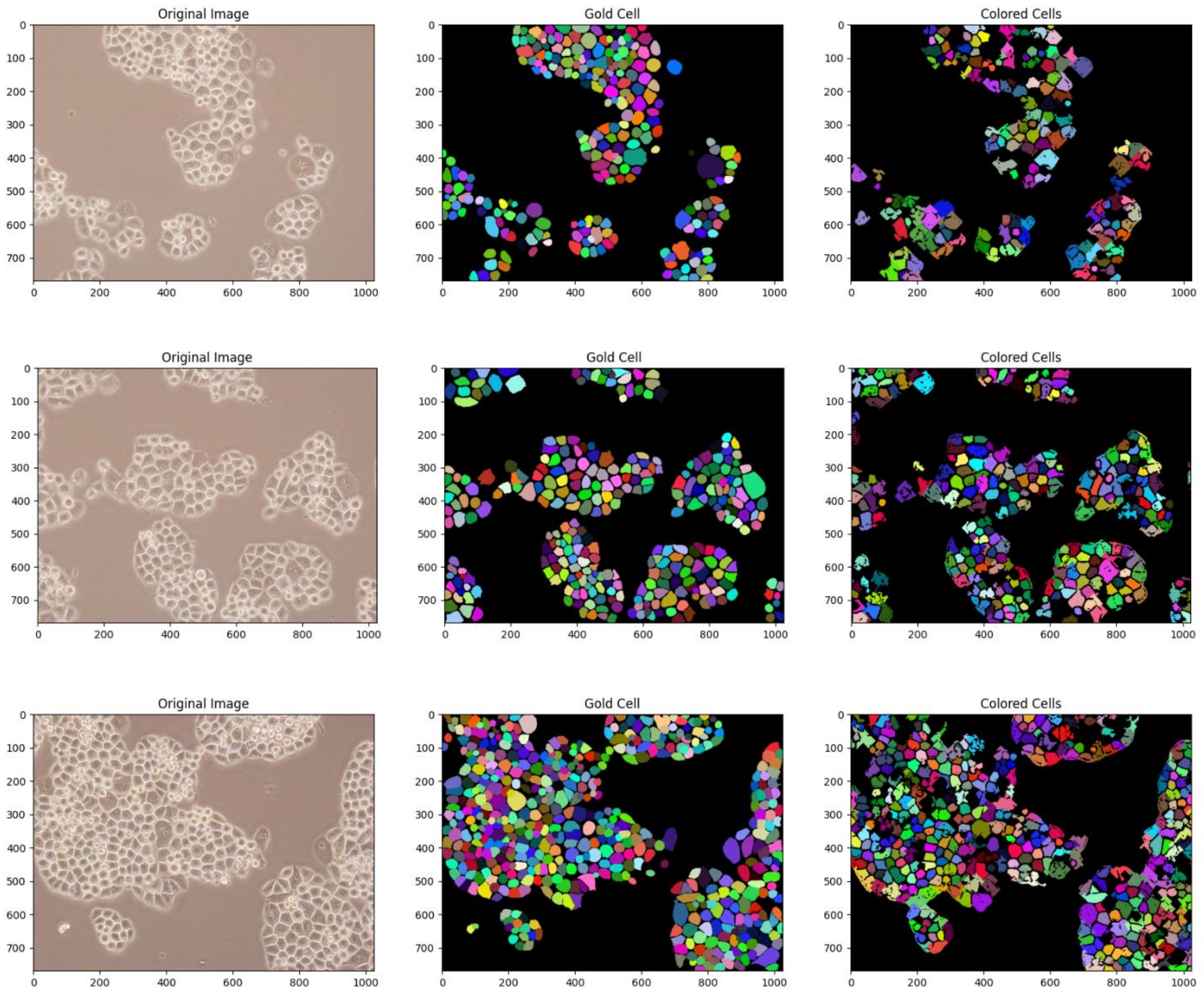
2) **Find Cell Locations:** In the first part of the assignment, I decided to the parameters for feature extraction. That is why I did not modularize the function and completed all the implementation in one function. The function begins by thresholding the input mask. It sets all pixels with a value of 1 to 255, effectively converting the mask to a binary representation where foreground pixels are set to 255. The input image is then blurred using a Gaussian filter with a kernel size of 5x5. The blurred image is then bitwise AND-ed with itself using the thresholded mask. This operation effectively retains only the regions of interest (cells) from the blurred image while suppressing background regions. The red channel of the masked image is extracted. It's assumed that the cell boundaries are best visible in the red channel. A Laplacian filter is applied to the red channel. This filter highlights regions of rapid intensity change, such as edges or boundaries. The resulting Laplacian image is normalized to the range [0, 255] and inverted. Values above 140 are set to 127, and values below 110 are also set to 127. This likely aims to enhance the contrast of the boundaries. Another Gaussian blur is applied to the normalized Laplacian image to further smooth it out. This step helps to remove noise and minor fluctuations in intensity. The red channel is converted to a floating-point representation, serving as the "sharp" image. The sharp image is combined with the Laplacian image to enhance the cell boundaries. This combination likely emphasizes edges and details in the image. The resulting combined image is normalized to the range [0, 255] and converted to an unsigned 8-bit integer format (uint8). For each cell in the input list of cells, a region growing algorithm (not defined in the provided code) is applied to identify the boundaries of the cell. The resulting regions are stored in the indices array. Finally, the identified regions are assigned random colors and stored in the “colored_region” array, which is then returned along with the “indices”.





IMAGES	PRECISION	RECALL	F1 SCORE
IM 1	0.97	0.65	0.78
IM 2	0.95	0.89	0.92
IM 3	0.93	0.77	0.84

3) **Find Cell Boundaries:** The function first converts the mask values to 255 where the value is 1, effectively creating a binary mask. This is done with the line mask[mask == 1] = 255. It applies Gaussian blur to the input image using a kernel size of (5, 5). This helps to smooth out noise and minor variations in intensity. The blurred image is then masked using the thresholded mask. A Laplacian filter is applied to the red channel. This filter highlights regions of rapid intensity change, such as edges or boundaries. The resulting Laplacian image is normalized to the range [0, 255] and inverted. Values above 140 or below 110 are set to 127. This likely aims to enhance the contrast of the boundaries. Another Gaussian blur is applied to the normalized Laplacian image to further smooth it out. This step helps to remove noise and minor fluctuations in intensity. The resulting combined image is clipped to the range [0, 255] and converted to an unsigned 8-bit integer format (uint8). For each cell in the input list of cells, a region growing algorithm (assumed to be defined elsewhere) is applied to identify the boundaries of the cell. The resulting regions are stored in the indices array. Finally, the identified regions are assigned random colors and stored in the "colored_region" array, which is then returned along with the "indices".



IMAGES (Ratio: 0.5)	PRECISION	RECALL	F1 SCORE
IM 1	0.48	0.51	0.48
IM 2	0.57	0.58	0.57
IM 3	0.37	0.54	0.44

IMAGES (Ratio: 0.75)	PRECISION	RECALL	F1 SCORE
IM 1	0.13	0.15	0.14
IM 2	0.21	0.21	0.22
IM 3	0.03	0.04	0.02

IMAGES (Ratio: 0.9)	PRECISION	RECALL	F1 SCORE
IM 1	0.00	0.00	0.00
IM 2	0.006	0.003	0.004
IM 3	0.00	0.00	0.00