

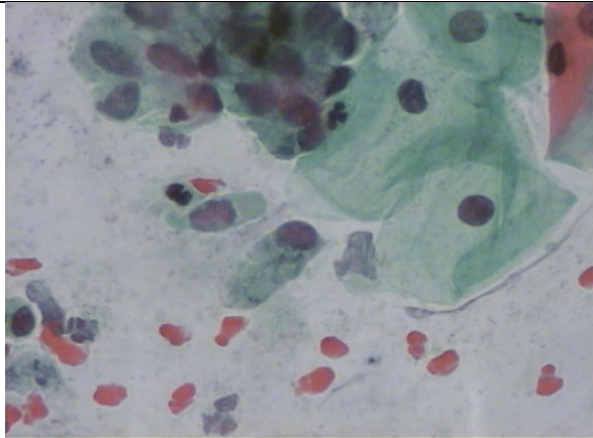
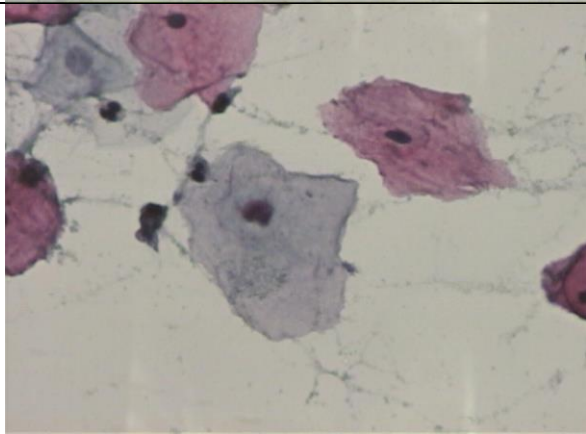
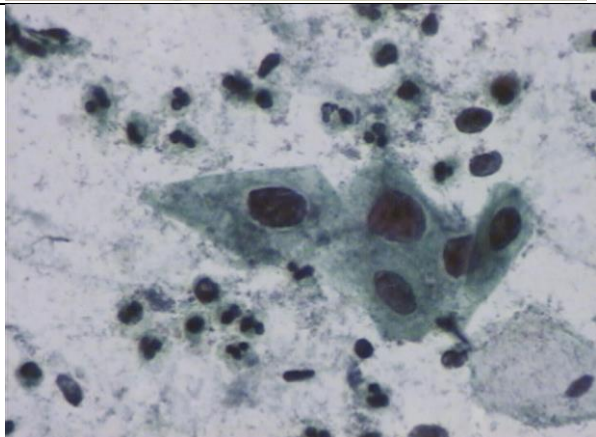
Github link: <https://github.com/BrandonM001/CMSC630Part1>

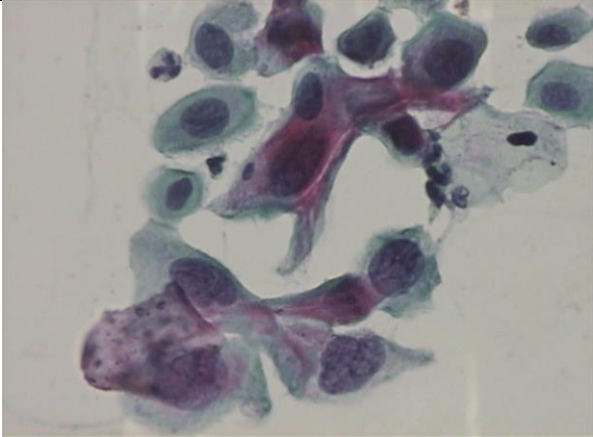
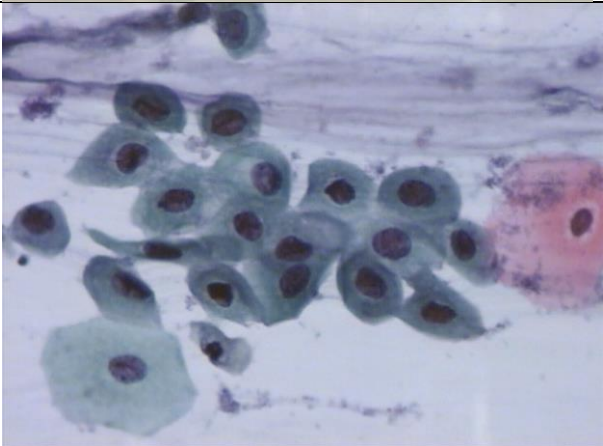
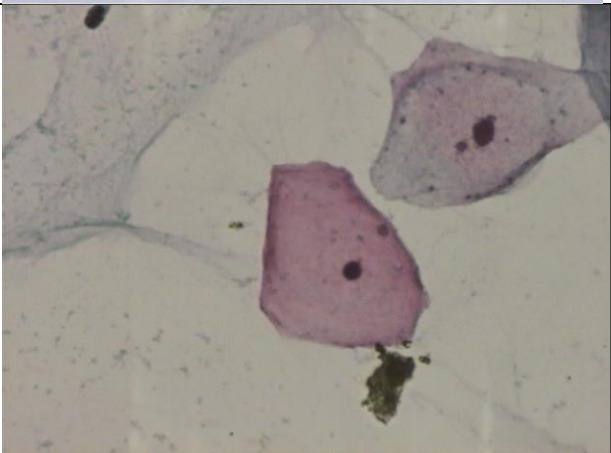
Relevant files: part2.py, inputFilePart2.txt, and BrandonMohan\_CMSC630\_part2.pdf

Functions:

- Main(): This function parses the information found in the input file and runs each image through each function. It also tracks the time of each run.
- monoColor(img): This function was imported from part 1, as I still use the grayscale images in this project. It takes a color image and RGB option as input and returns a grayscale version of that image. For example, if I give it "image.bmp R", it will provide a grayscale version of that image based on the red values.
  - It takes 1.34 seconds on average to run monoColor.
- sobelFilter(img): This function follows the Sobel algorithm to identify large changes in intensity and thus find the edges in the image. I used the operator provided in Lecture 8 slide 27.
  - It takes about 283 seconds on average to run sobelFilter and is by far the slowest part of running the program. I would encourage you to comment it out when grading my code (after you verify it's correct).
  - My Sobel implementations iterates through each pixel and grabs the surrounding pixels to measure the amount of change in the group. A large amount of change indicates an edge. It then marks the intensity of the change so it can be output in a new image.
- dilationErosion(img, kernel, dilationOrErosion): This function follows the algorithm for dilation to increase the size of edges. Erosion works with a similar algorithm, so instead of rewriting another function I combined them. Set the parameter dilationOrErosion to True for Dilation, or False for Erosion. Unfortunately, the structuring element is hardcoded, as I struggled to get it to work until I was too close to the deadline to adapt it.
  - Similarly to Sobel, it parses each pixel and grabs the pixels around it. However, it then performs a union/multiplication on each group with the structuring element. We then save the highest value left in the group and put that as the new pixel value.
  - Erosion works the same way as Dilation, except we save the lowest non-zero value in the group.
  - It takes about 7 seconds on average to process image dilation.
  - It takes about 4.8 seconds to process image erosion.
- clustering(img, clusters, iters): This function attempts to use k-means clustering to divide the image. The cells in the foreground will be highlighted, whereas everything else will be removed in the background. Note that due to the random numbers generated, it tends to be hit or miss with the foreground. It will usually find the darker centers of the cells, but it doesn't always pull out the entire cell.
  - My clustering function takes about 28 seconds to run.

The dataset that we are using contains 7 different classes of cells. I have made a table below showing each class and the example image + directory used in the input file.

Classification	File	Stock Image
cyl	"Cancerous cell smears 2023\Cancerous cell smears\cyl02.BMP" R	
Inter	"Cancerous cell smears 2023\Cancerous cell smears\inter24.BMP" G	
Let	"Cancerous cell smears 2023\Cancerous cell smears\let73.BMP" B	

Mod	"Cancerous cell smears 2023\Cancerous cell smears\mod19.BMP" R		
Para	"Cancerous cell smears 2023\Cancerous cell smears\para05.BMP" G		
Super	"Cancerous cell smears 2023\Cancerous cell smears\super11.BMP" B		

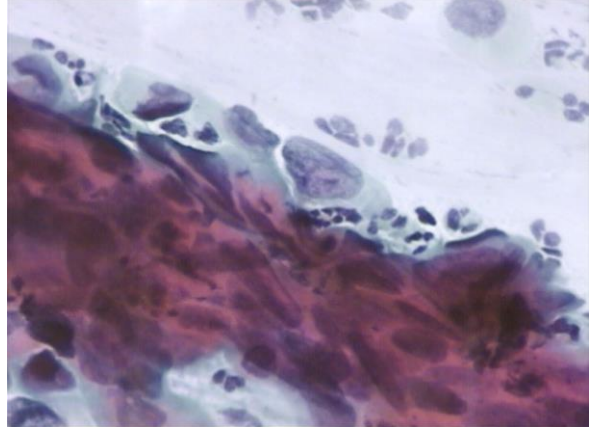
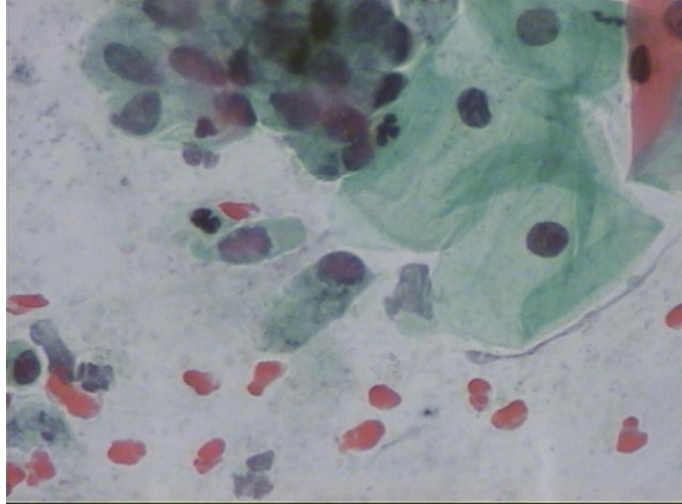
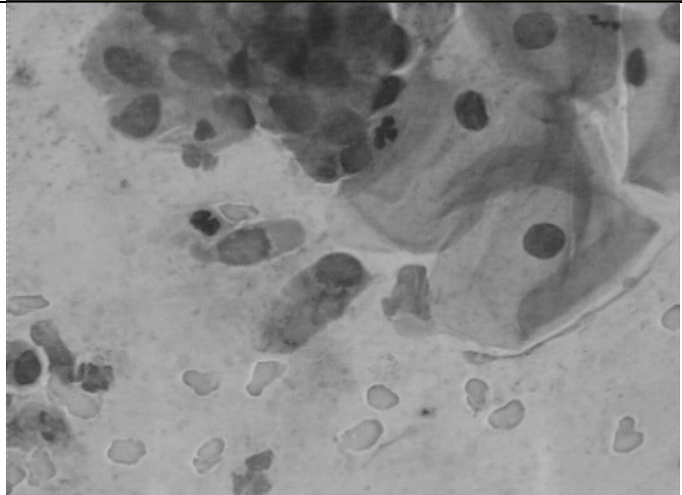
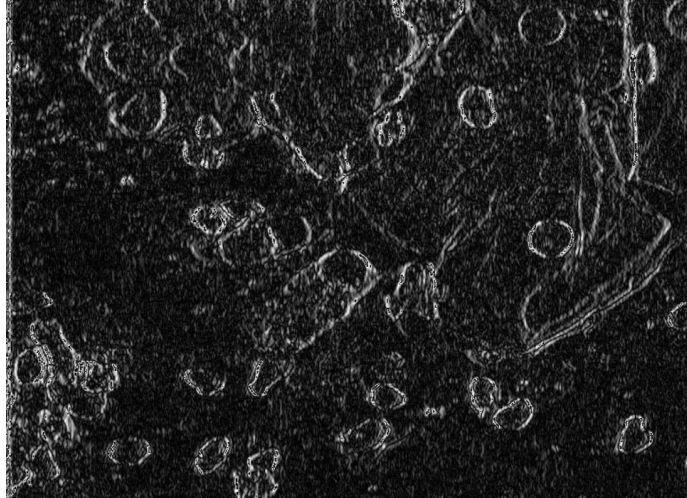
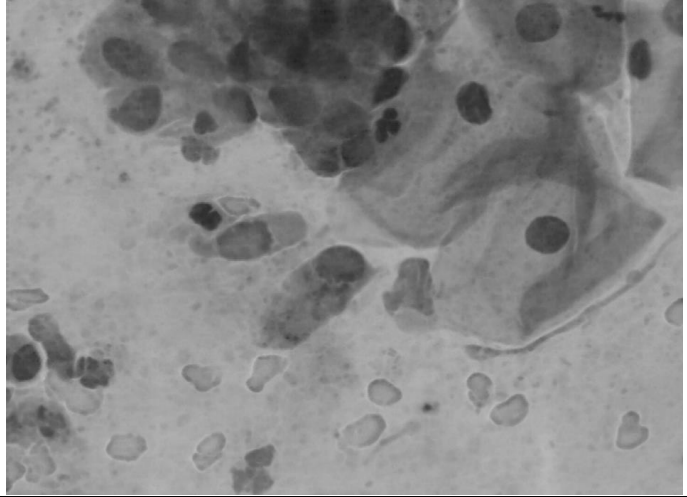
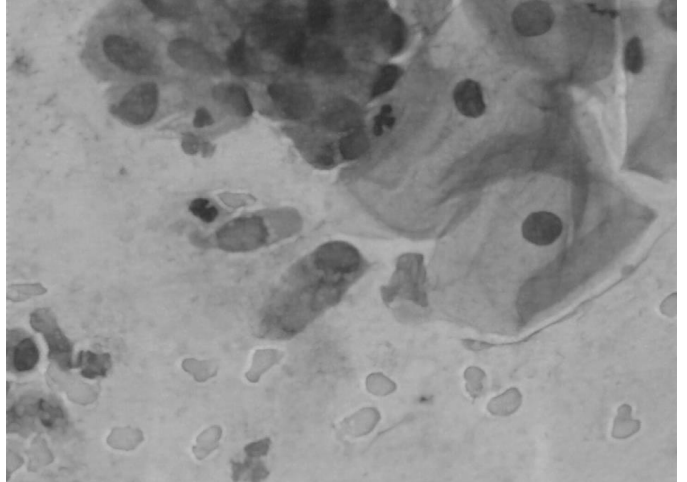
svar	"Cancerous cell smears 2023\Cancerous cell smears\svar73.BMP" R		
------	---	--	--

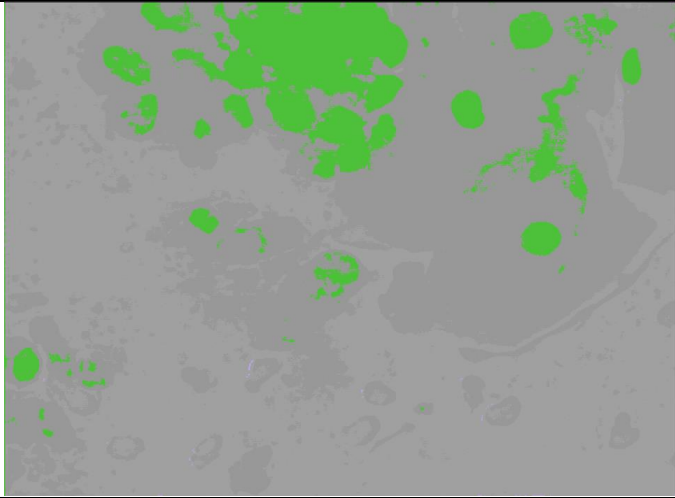
Image Class Results:

cyl02.BMP comparison:

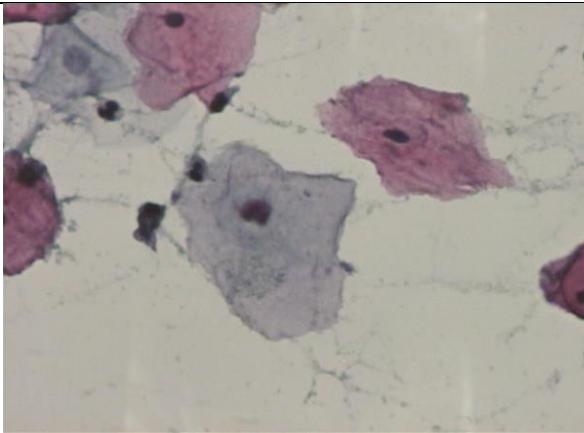
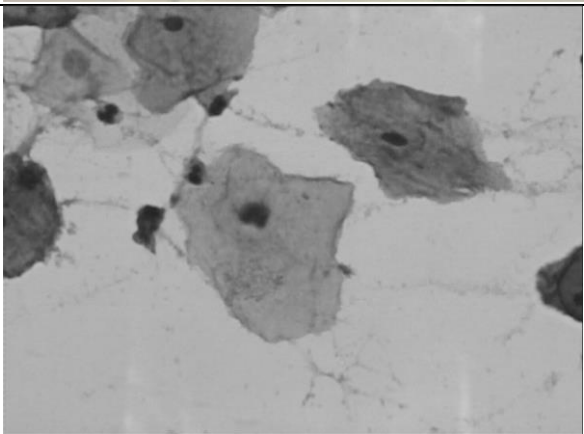
Stock		This is the untouched photo
GrayScale (R)		This photo was created by taking the Red values in the original image. This is the "model" photo we use when comparing the results of edge detection.

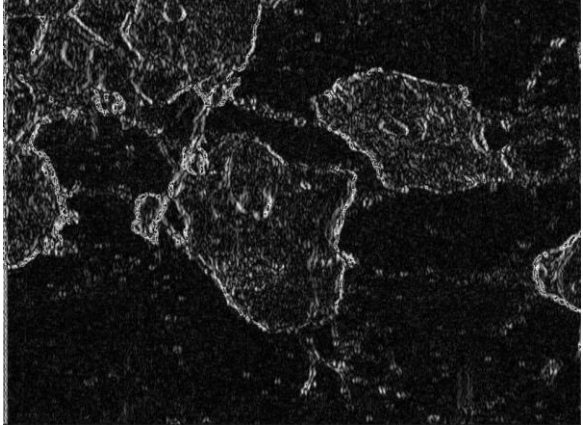
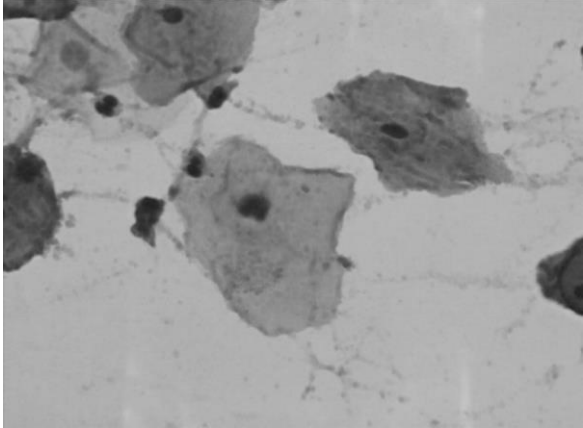
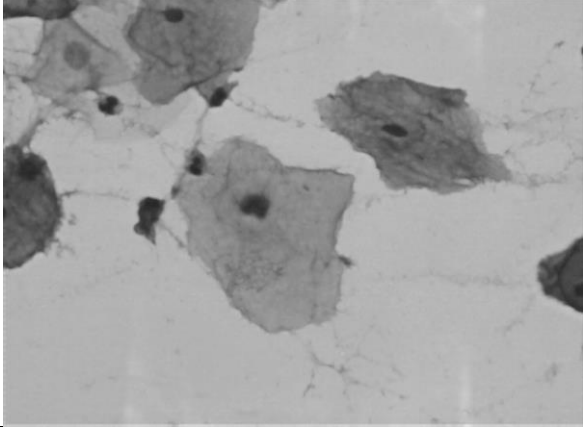

Sobel Edge Detection		This photo was created using the edges found in the grayscale image.
Dilation		This photo is almost the same as the grayscale image, except the edges are slightly bigger. It is most noticeable on overlaying the images on top of each other.
Erosion		This photo is almost the same as the grayscale/dilation image, except the edges are slightly smaller. Once again, it is easier to see when overlaying the images on top of each other.



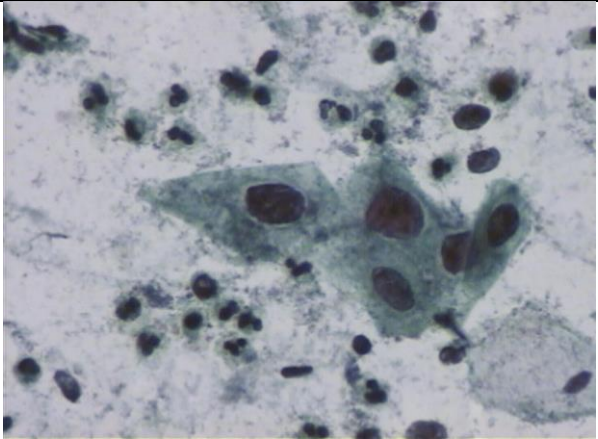
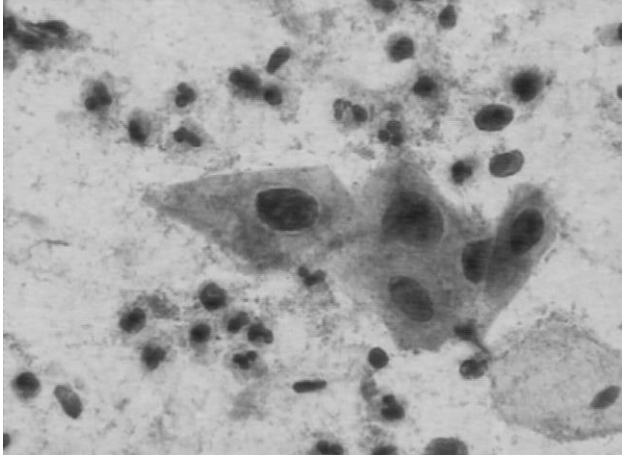

Segmentation		The parts of the cells found in the foreground are marked in this image.
--------------	--	--

inter24.BMP:

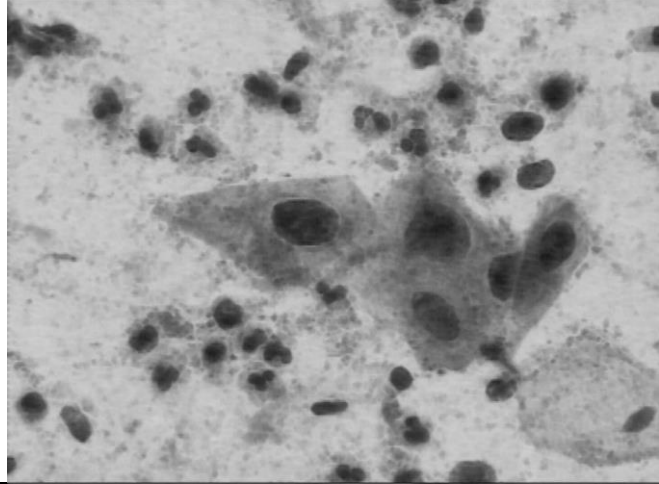
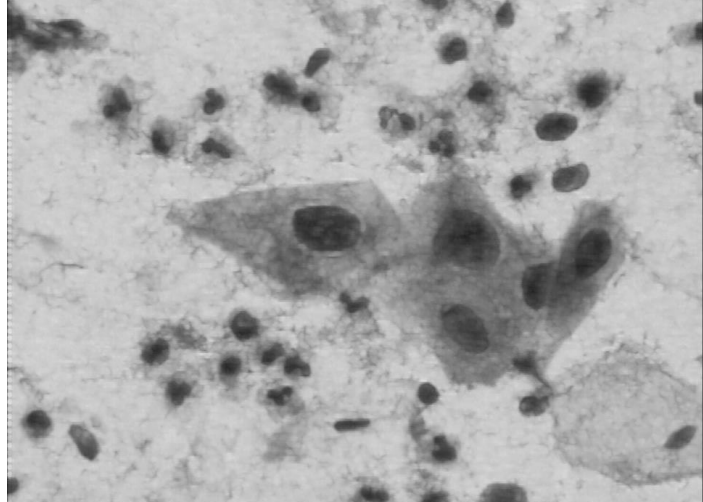
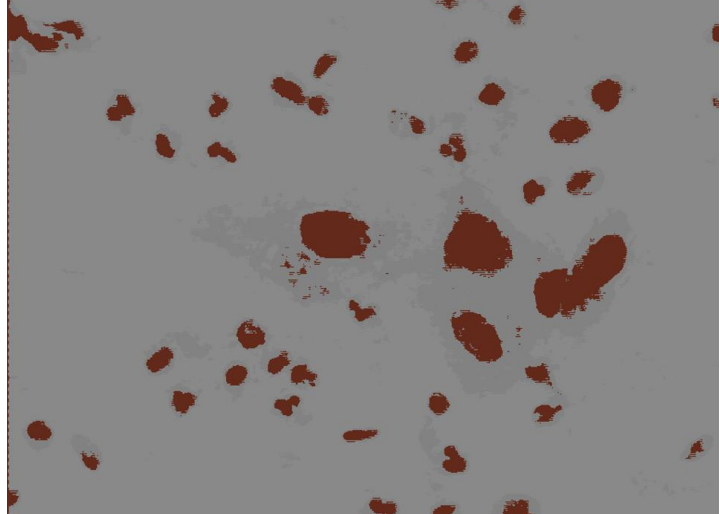
Stock	
Grayscale(G)	

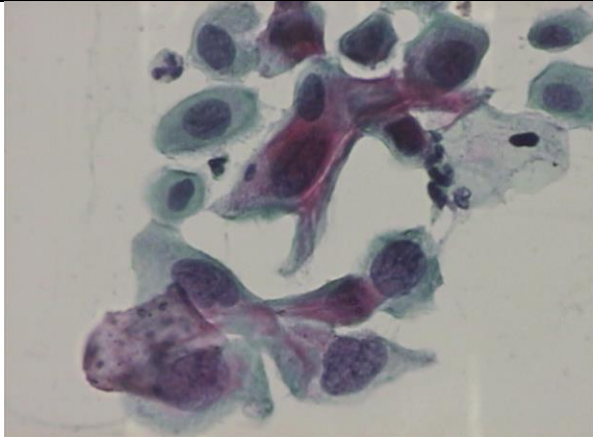
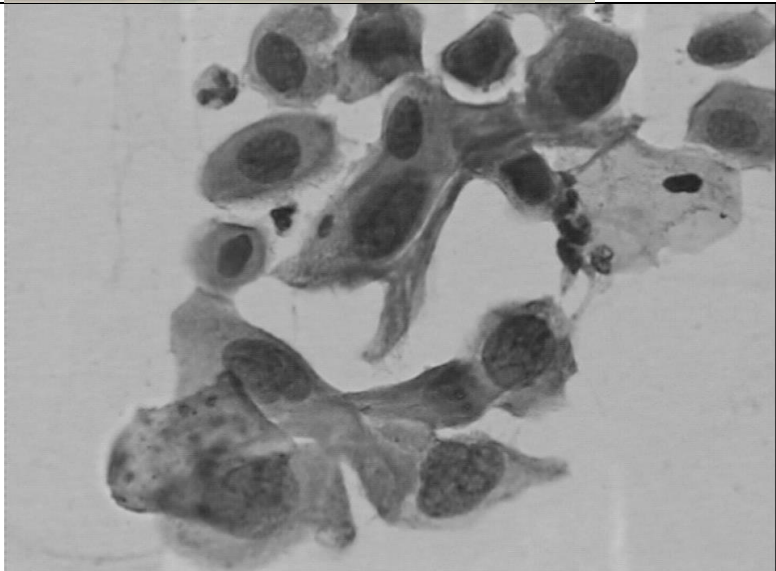
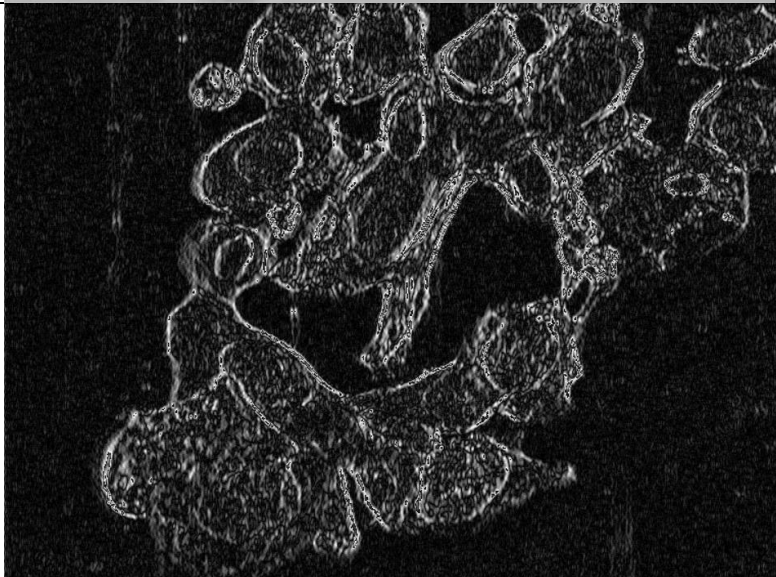
Sobel	
Dilation	
Erosion	
Segmentation	

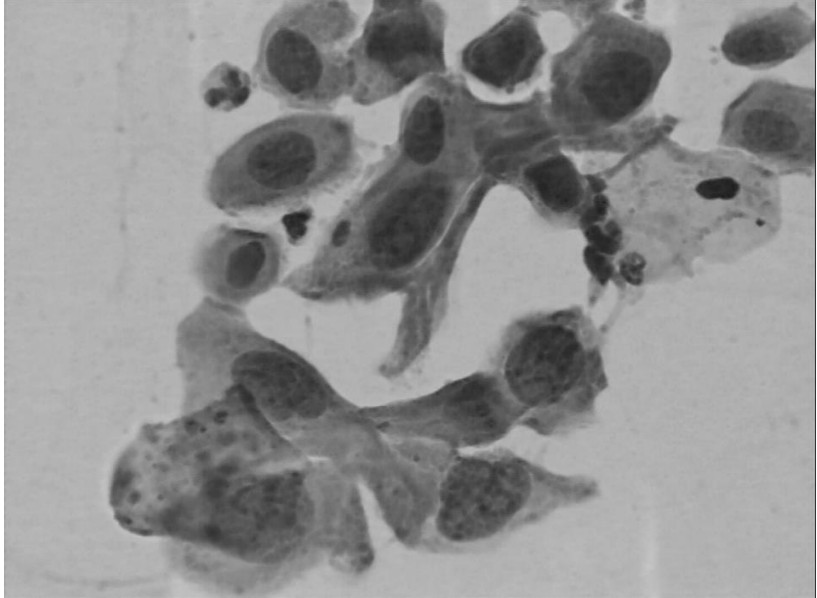
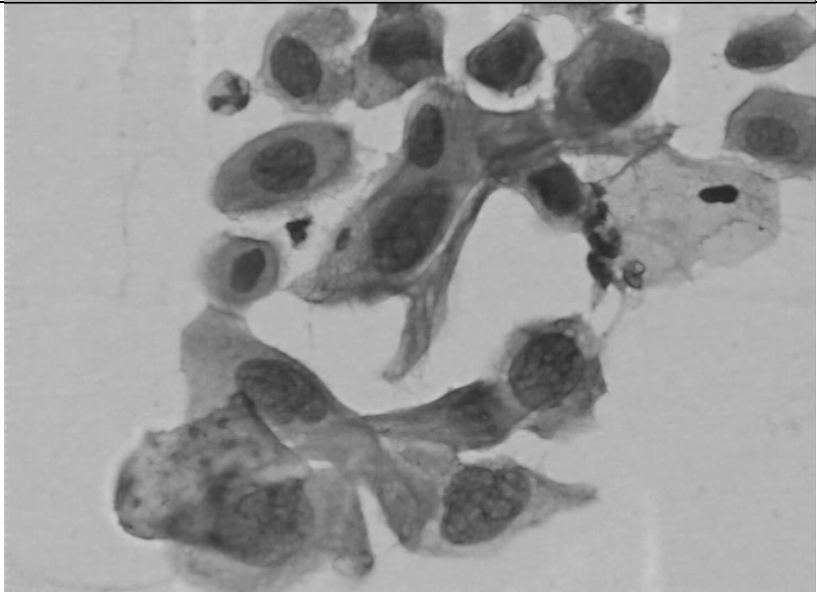
let73.BMP

Stock		
Grayscale(B)		
Sobel		

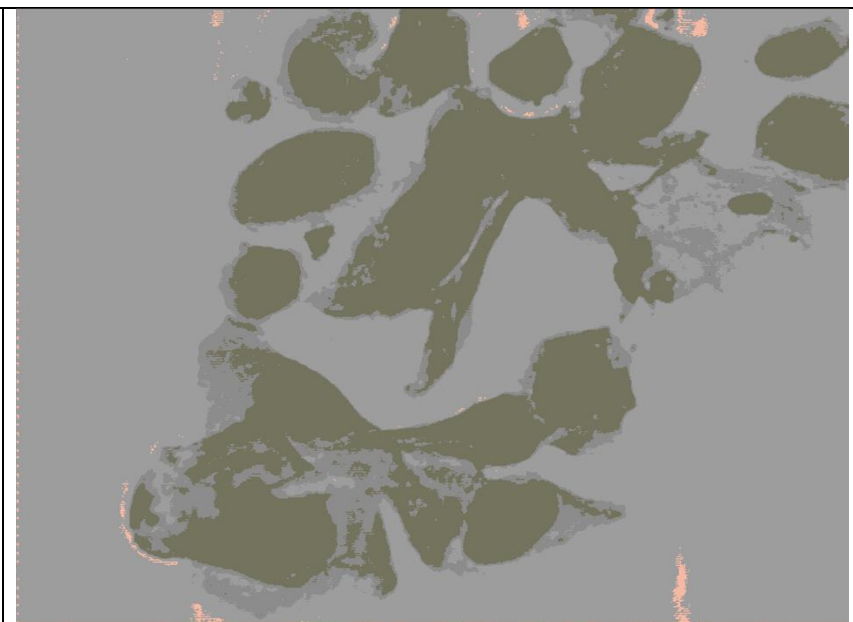


Dilation		
Erosion		
Segmentation		

Stock	
Grayscale(R)	
Sobel	

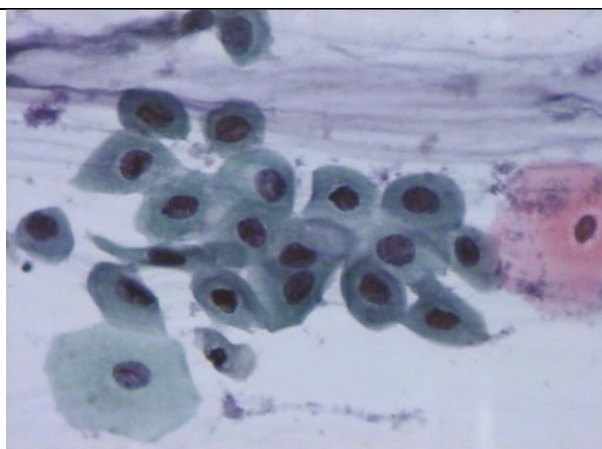
<p>Dilation</p>	
<p>Erosion</p>	

Segmentation

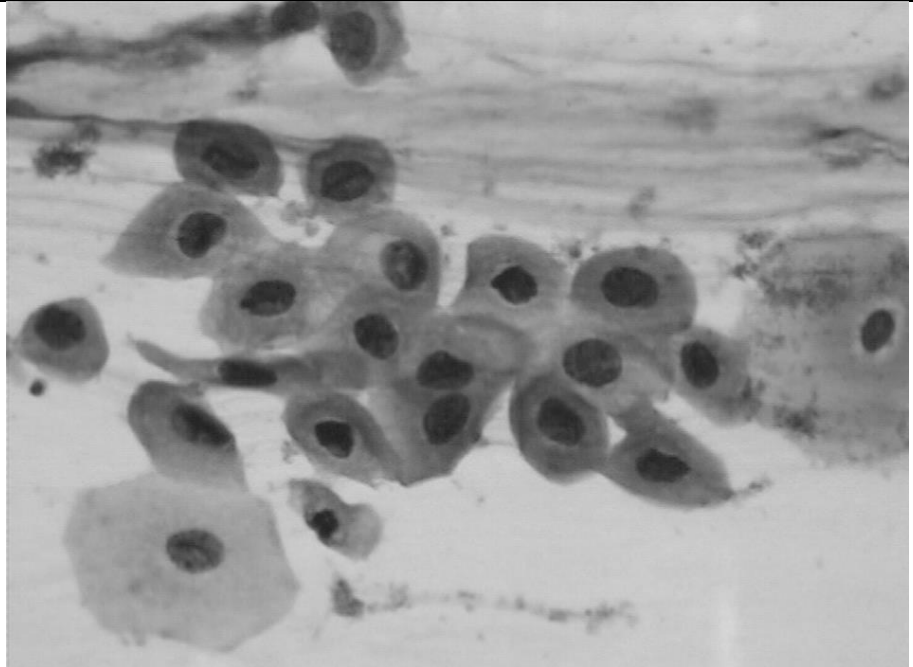


para05.BMP

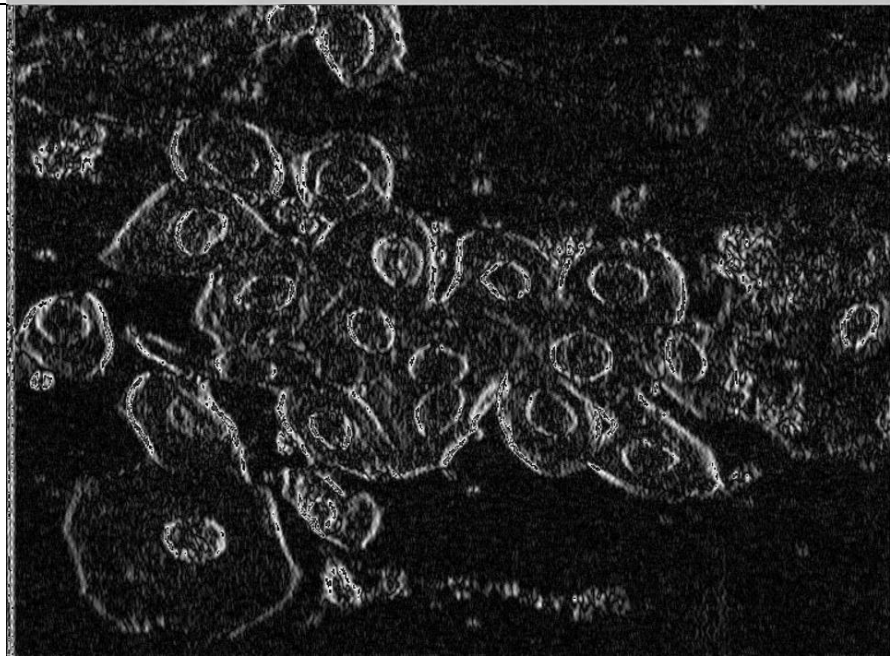
Stock



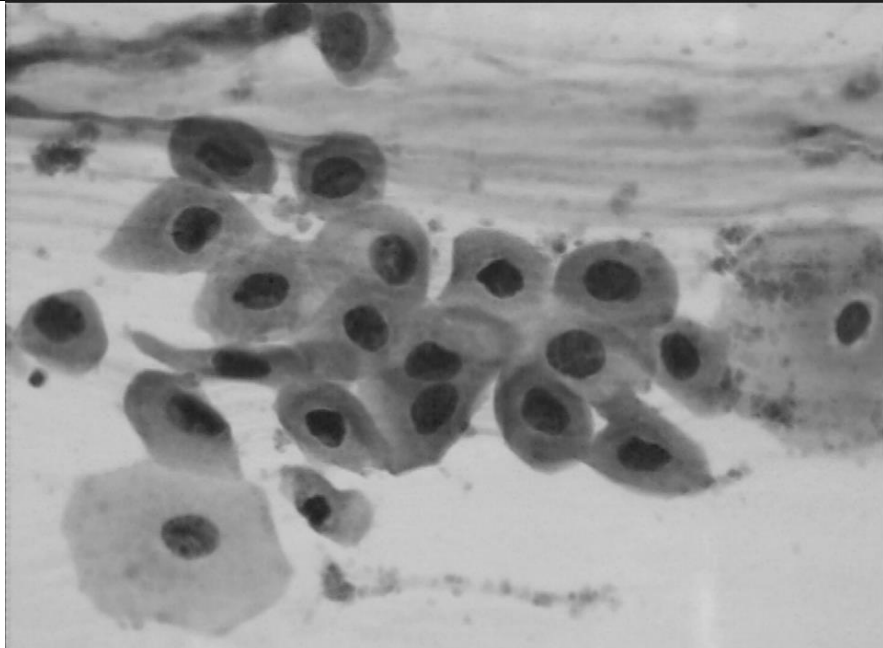
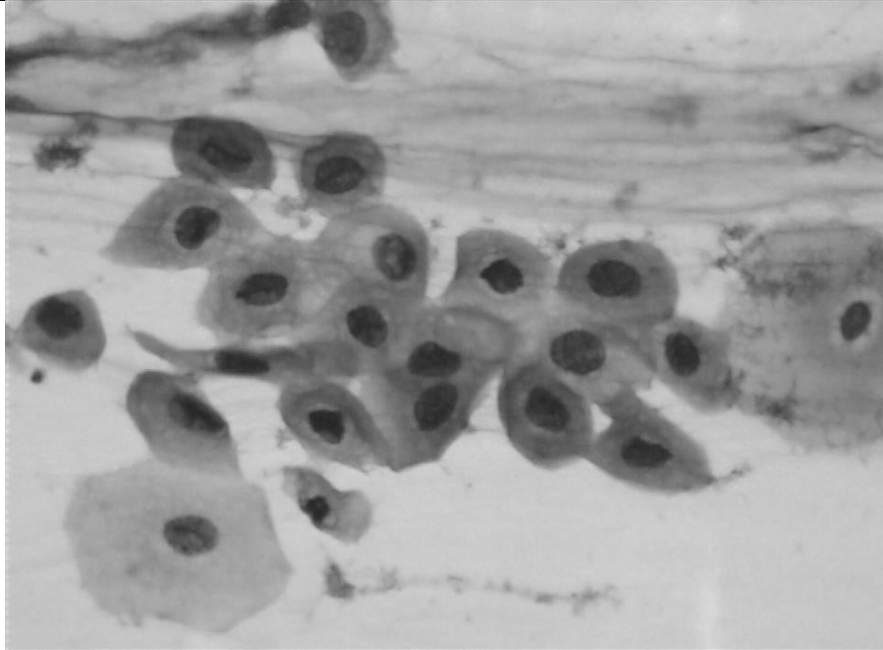
Grayscale(G)

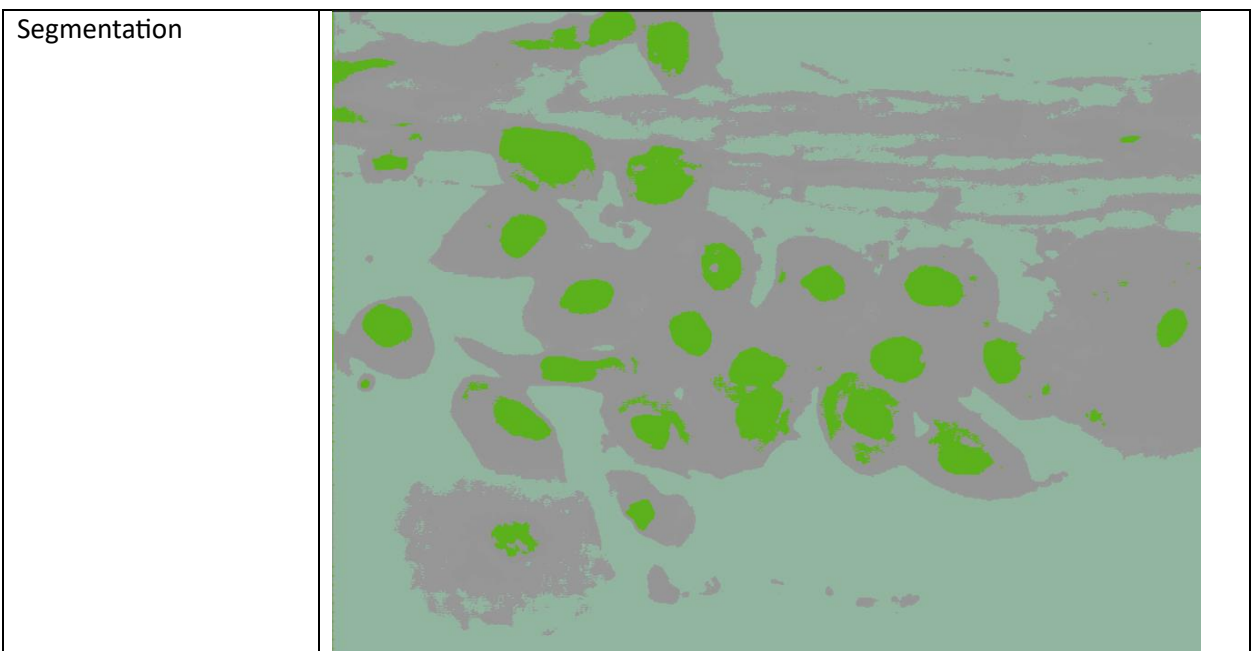


Sobel

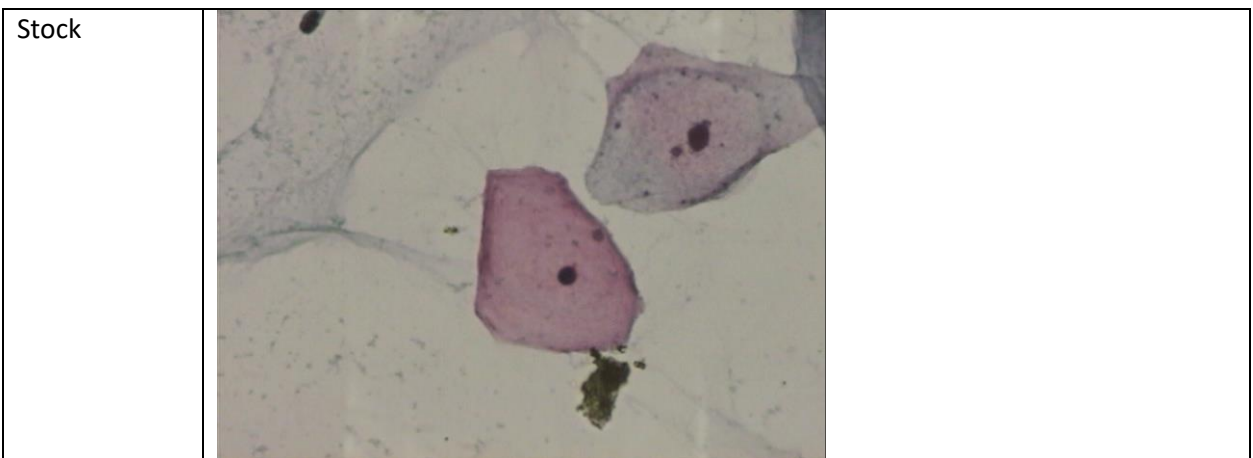


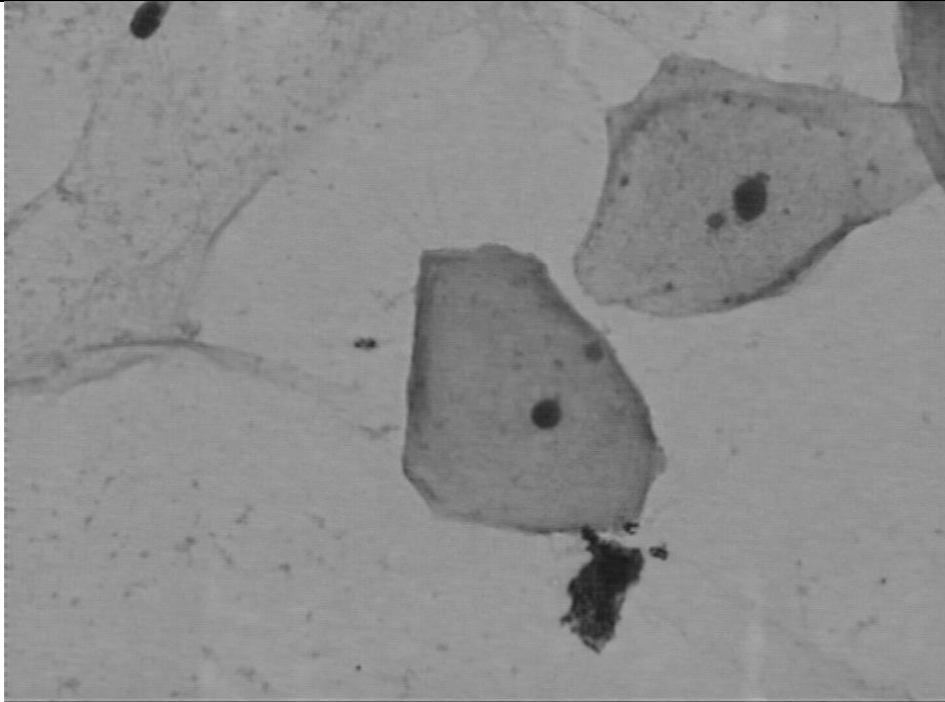
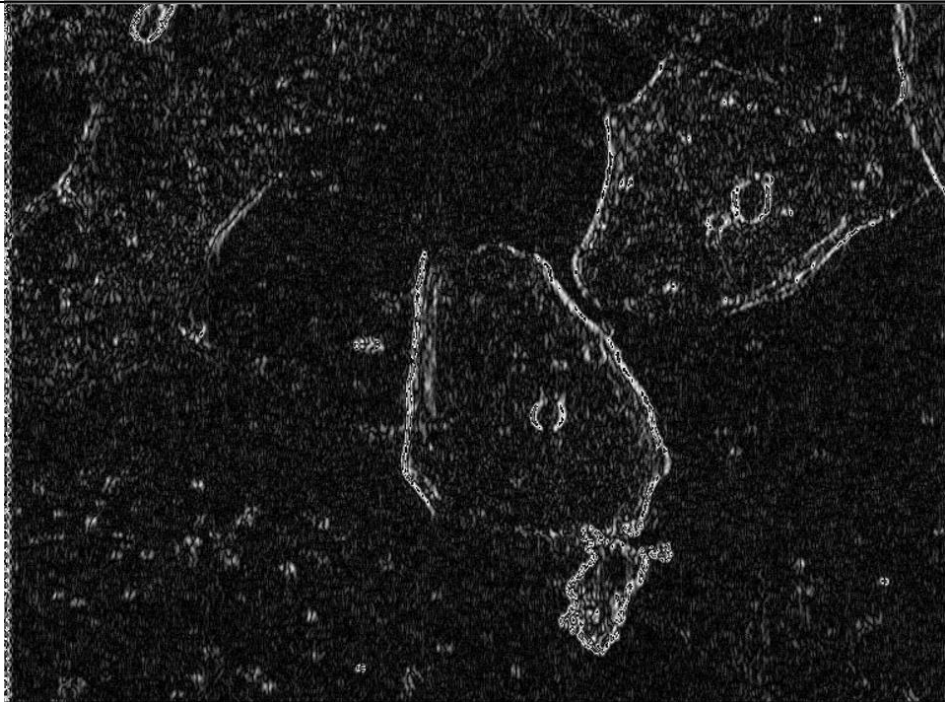


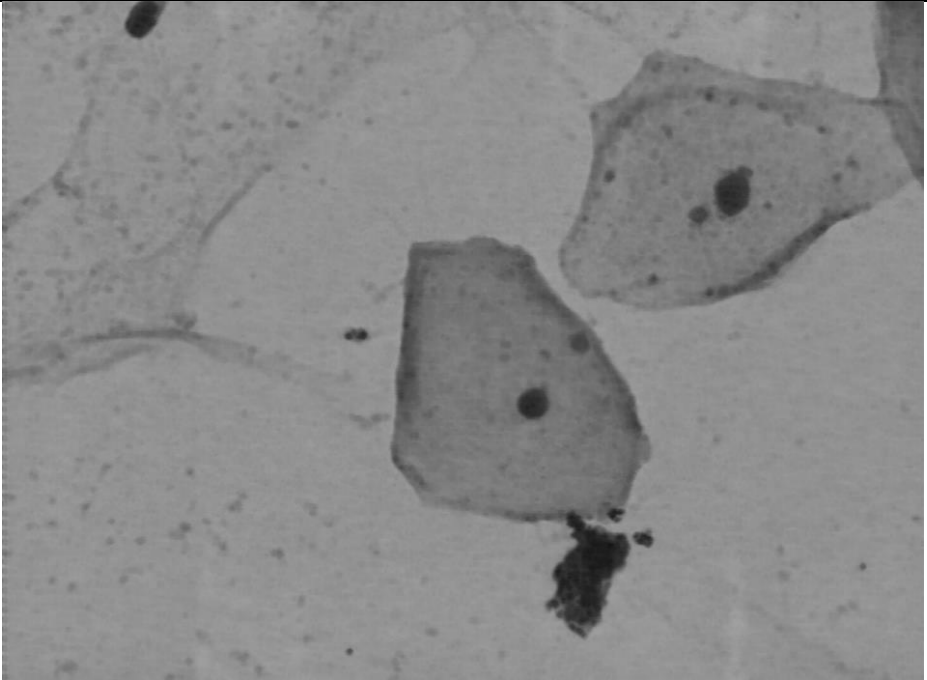
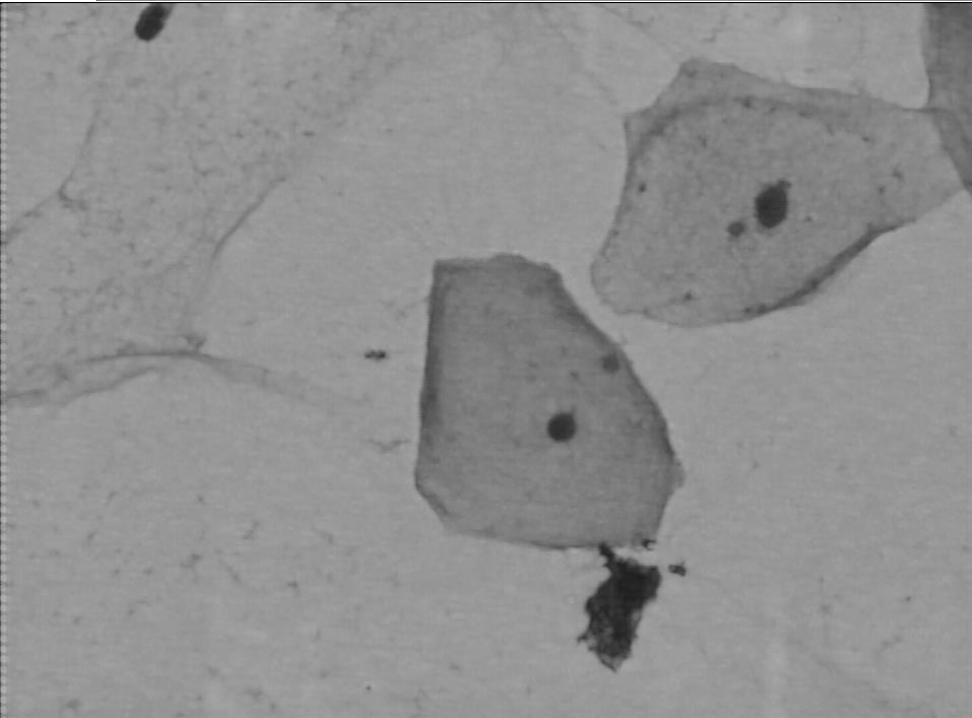
<p>Dilation</p>	
<p>Erosion</p>	



super11.BMP

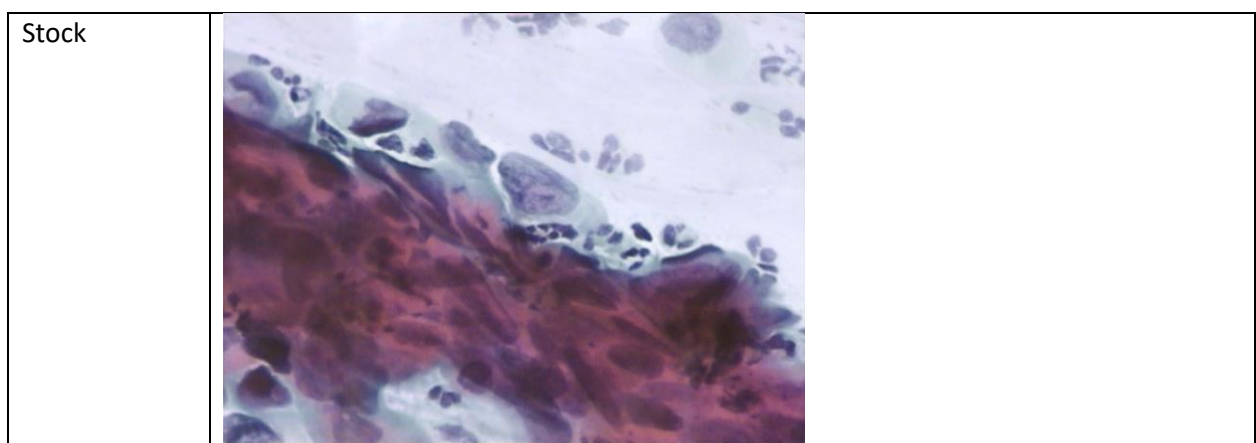


Grayscale(B)	 A grayscale micrograph showing a biological specimen, possibly a cell or tissue section. The image is mostly light gray with some darker, irregular shapes and a few small, dark, circular features. The overall texture is grainy and noisy.
Sobel	 The same grayscale image as above, but processed using Sobel edge detection. The background is now almost entirely black, and the edges of the biological structures are highlighted in bright white and light gray. The edges are somewhat jagged and noisy, typical of edge detection algorithms.

Dilation	
Erosion	

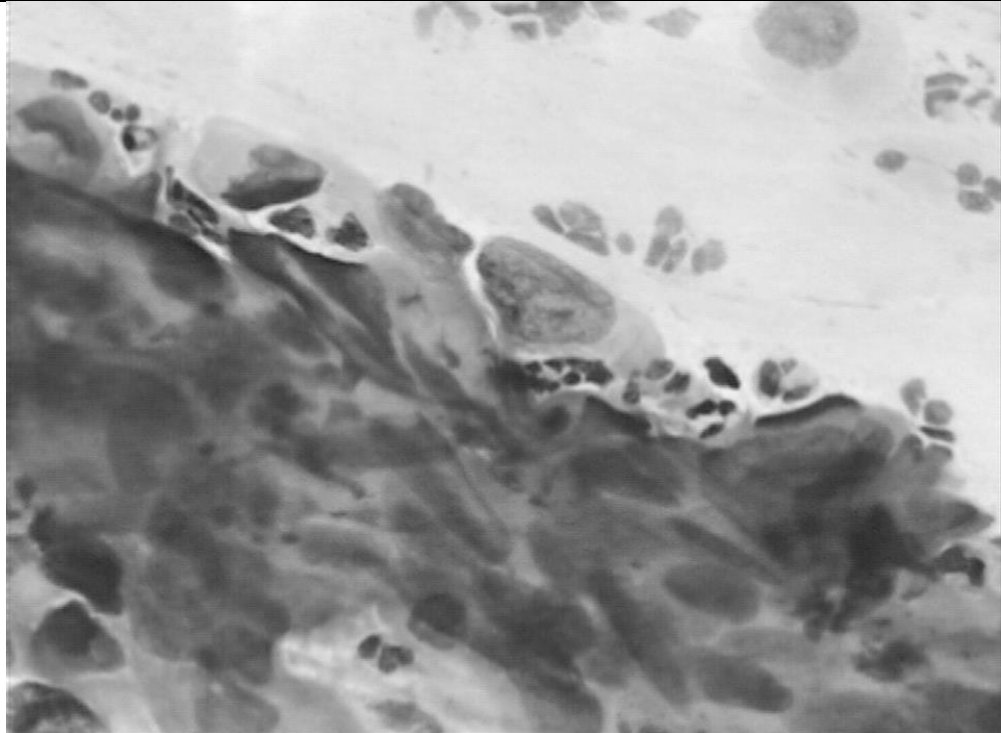


svar73.BMP

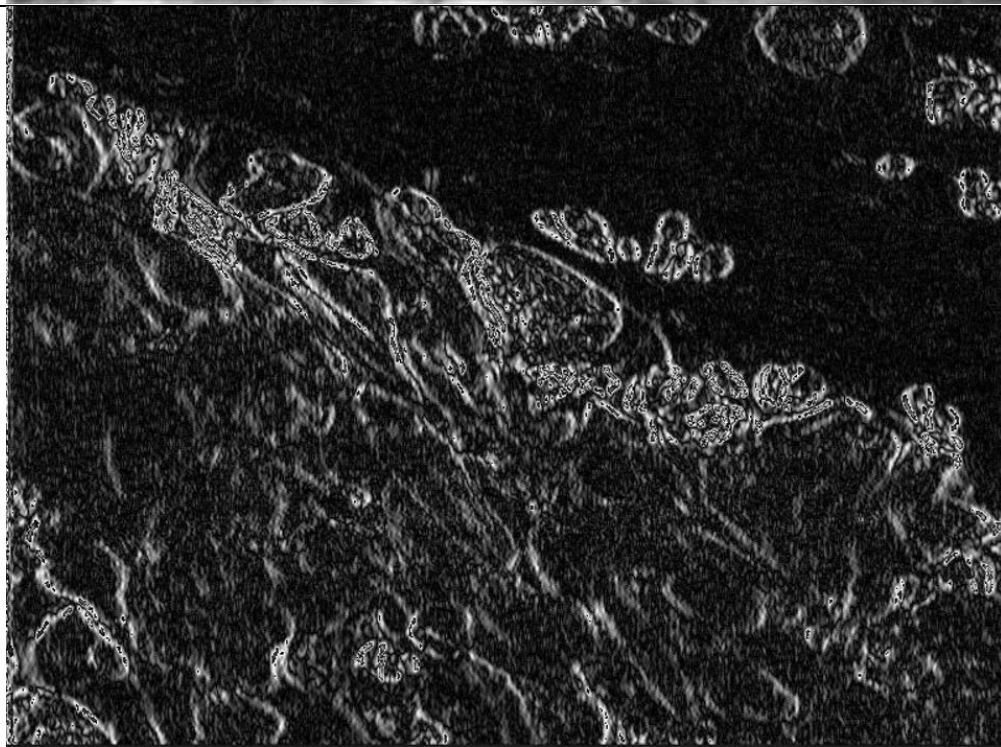




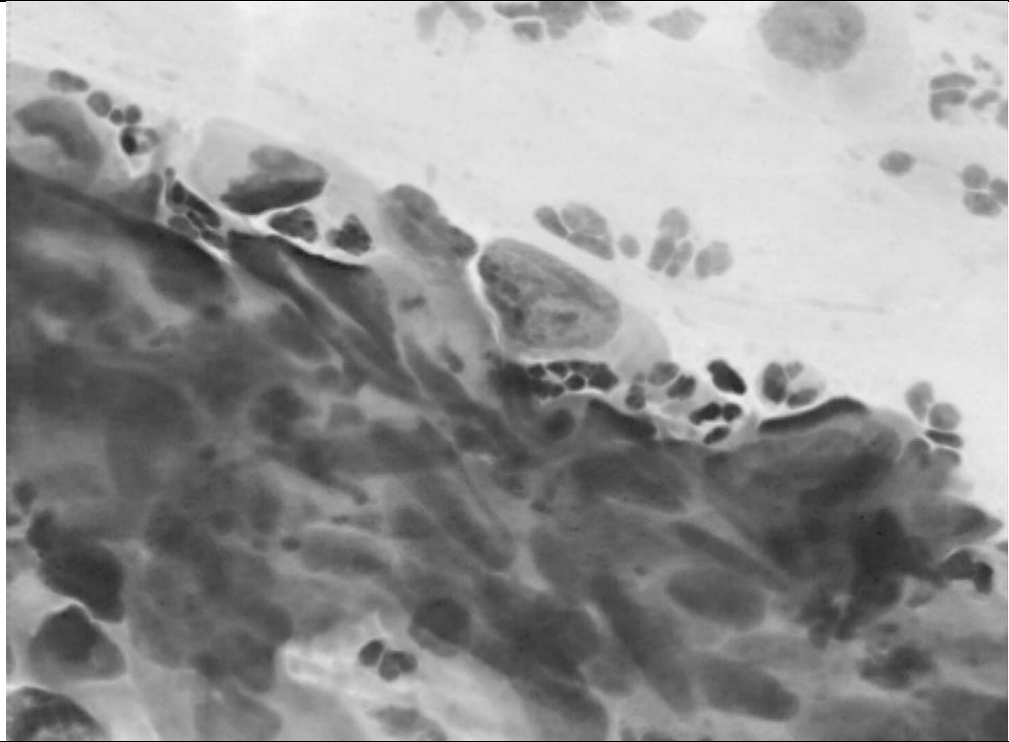
Grayscale(R)



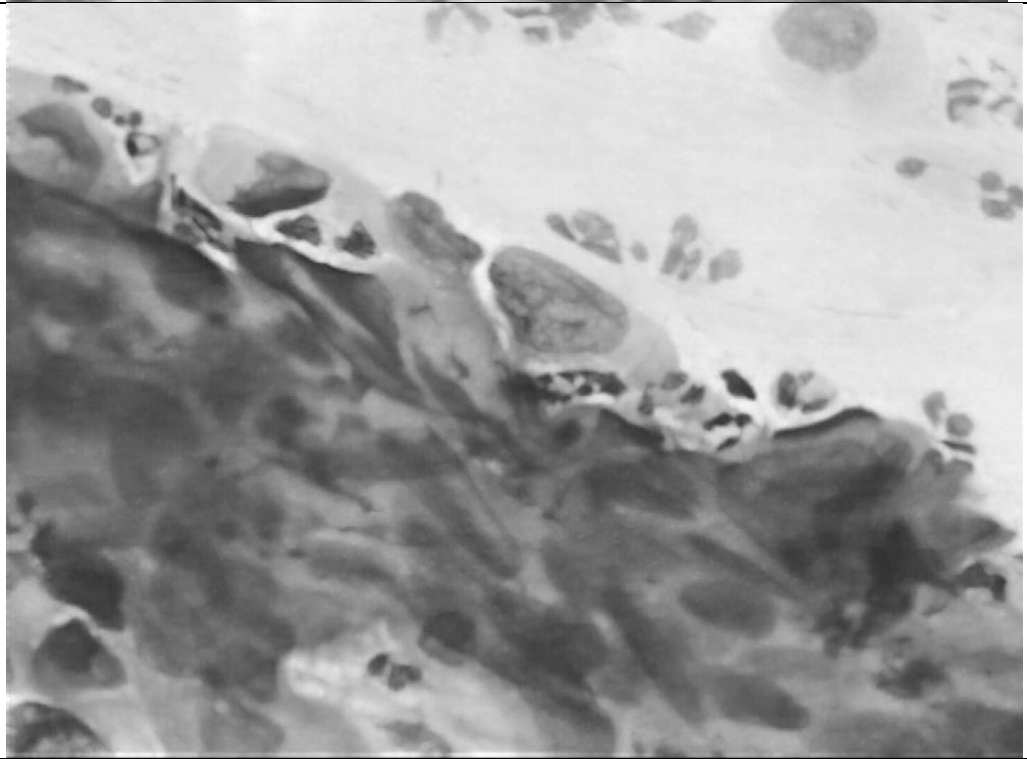
Sobel



Dilation



Erosion



Segmentation

