CSMS 630 Project 3 – Feature Extraction and Classification

By: Kirtan Shah

This project is combination of different filters and conversions. It is written in Python and process 499 images in a span of 2.5 hours. The GitHub link for this project is here. There is a configuration .ini file *(userConfiguration.ini)* attached with the project which must be place in the same folder as the main python file. The user my edit the .ini file to receive different results. To get the most optimum results please follow the format of the already established values within the ini file. The output and original images are store in Google drive.

The image path must complete path to the images:

- C:/Users/user/Cancerous cell smears/*

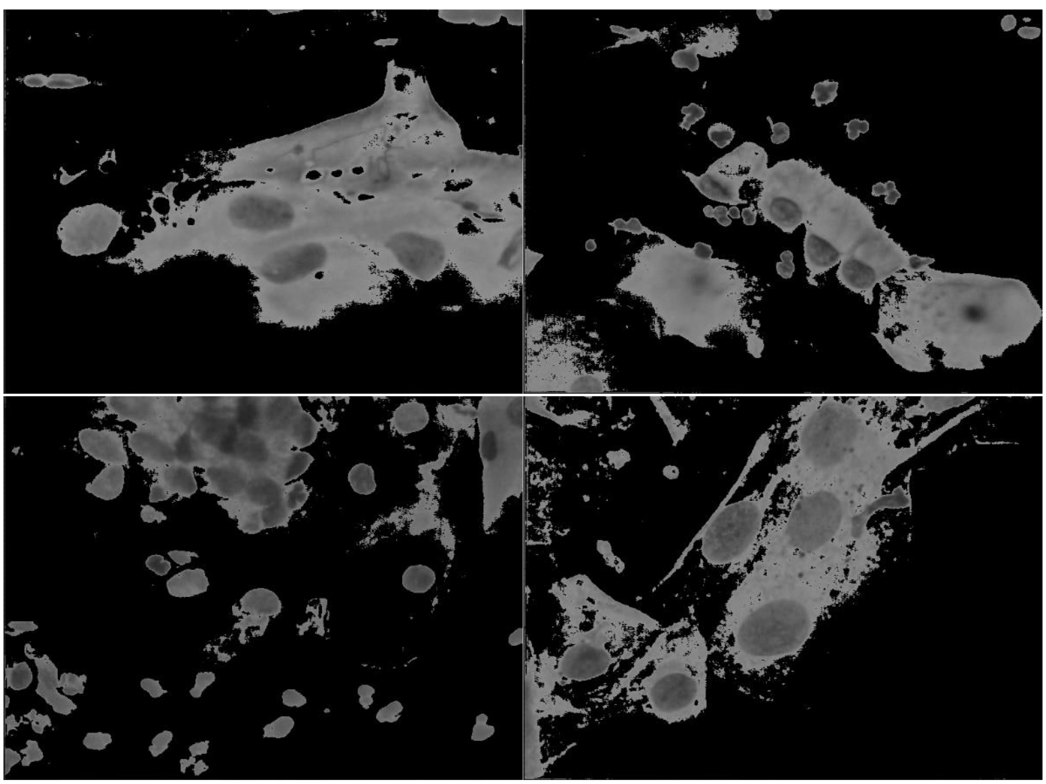
The output path must complete path to the directory to store the images:

- C:/Users/user/Documents/output

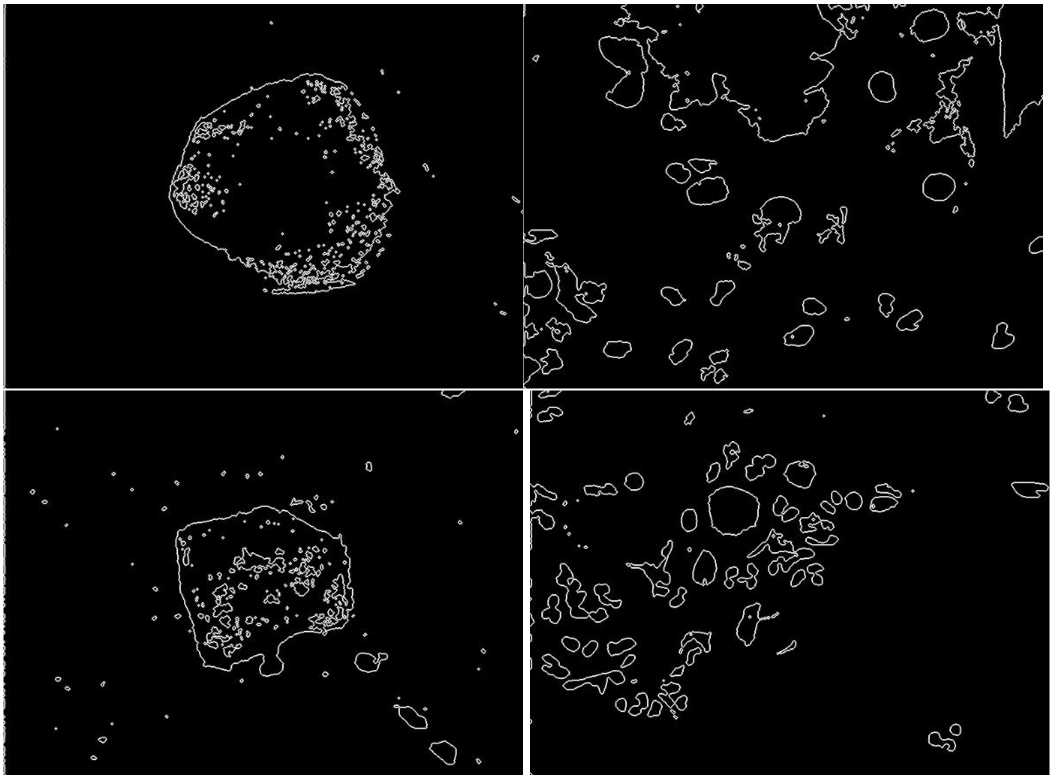
Image Segmentation

To highlight the important parts of the image I used Historgram based segmentation and Contours

Histogram Segmentation – This approach allowed me cluster import imptant pixels based on a threshold. The threshold is a combination of sigma, variance and cumulative mean for the image that when all combined helped me see the cells from the image.



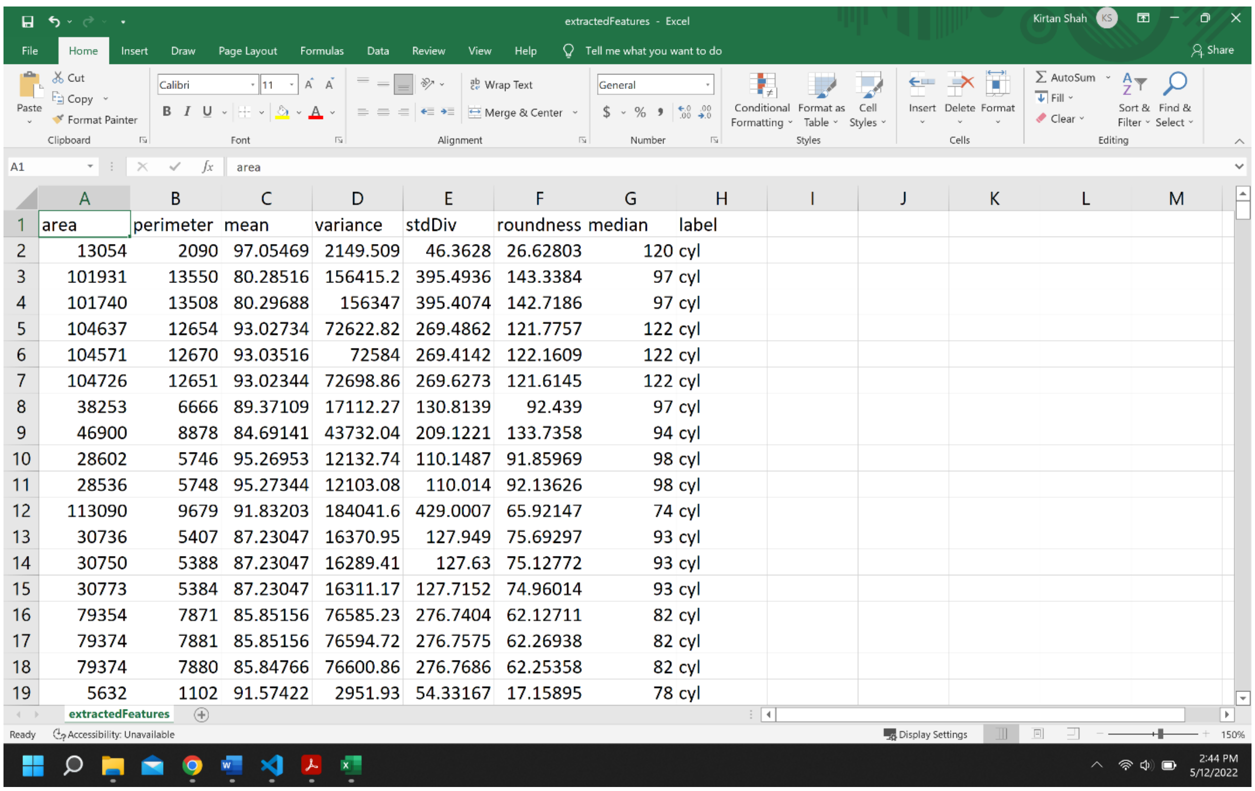
Contours – To find the contours of the cells within the image. I first eroded the image then dilated the eroded image and then subtracted the dilated by the eroded to just give the outline of the cells within.



Feature Extraction

Many features can be extracted form images. Features that can help identify important aspects within an image. You can view the all extracted features in extractedFeatures.csv. I managed to extract **Eight** different features:

- **Area**: count of pixels inside object region
- **Perimeter:** the count of the pixels in side the contour
- **Mean:** the object mean used to during the histogram segmentation
- **Median:** gather list of all pixels within the object, sorted and located the middle pixel
- **RGB_Mean:** the average mean of all three color channels combined
- **Roundness:** (perimeter*perimeter)/ 4*math.pi*Area
- **Standard Deviation:** the square root of the variance
- **Variance:** the object variance used to during the histogram segmentation



area	perimeter	mean	variance	stdDiv	roundness	median	label
13054	2090	97.05469	2149.509	46.3628	26.62803	120	cyl
101931	13550	80.28516	156415.2	395.4936	143.3384	97	cyl
101740	13508	80.29688	156347	395.4074	142.7186	97	cyl
104637	12654	93.02734	72622.82	269.4862	121.7757	122	cyl
104571	12670	93.03516	72584	269.4142	122.1609	122	cyl
104726	12651	93.02344	72698.86	269.6273	121.6145	122	cyl
38253	6666	89.37109	17112.27	130.8139	92.439	97	cyl
46900	8878	84.69141	43732.04	209.1221	133.7358	94	cyl
28602	5746	95.26953	12132.74	110.1487	91.85969	98	cyl
28536	5748	95.27344	12103.08	110.014	92.13626	98	cyl
113090	9679	91.83203	184041.6	429.0007	65.92147	74	cyl
30736	5407	87.23047	16370.95	127.949	75.69297	93	cyl
30750	5388	87.23047	16289.41	127.63	75.12772	93	cyl
30773	5384	87.23047	16311.17	127.7152	74.96014	93	cyl
79354	7871	85.85156	76585.23	276.7404	62.12711	82	cyl
79374	7881	85.85156	76594.72	276.7575	62.26938	82	cyl
79374	7880	85.84766	76600.86	276.7686	62.25358	82	cyl
5632	1102	91.57422	2951.93	54.33167	17.15895	78	cyl

Classification

10 Fold Cross Validation: Image classification is a hotly contested area within image processing and requires extensive research. All the features extracted above can used in conjunction to classify the image. I started the 10 old cross validation. By splitting the data let in to 2 sperate training and testing datasets. 9/10th of the data was used for training and 1/10th of the dataset was reserved for testing. Once data was split into 9 even fold and 1 fold with extra images (due to 1 corrupted image super45). I extracted the labels for the testing set and used as a validation point once I had the proper results.

K-Nearest Neighbor Classifier: I gathered the training and testing data, used the eight features inside, and one by one found an accuracy score of fold. I leverage the Euclidian distance between each training and testing feature, sorted the distances and compared the final label towards the training label. For some strange reason my accuracies were always around 25%. I believe this could be improved if there was way to incorporate the number of images per class probability and use it to gather more precise votes on the nearest neighbor. Due to time constraints, I was not able to test this theory yet I believe it could significantly improve the accuracy score. The tables below show the 10 folds with different K combinations:

K = 1

```
*****Starting*****
fold = 1: k = 1, Accuracy = 26.0
fold = 2: k = 1, Accuracy = 26.444
fold = 3: k = 1, Accuracy = 26.222
fold = 4: k = 1, Accuracy = 23.333
fold = 5: k = 1, Accuracy = 22.444000000000003
fold = 6: k = 1, Accuracy = 21.111
fold = 7: k = 1, Accuracy = 22.889
fold = 8: k = 1, Accuracy = 26.444
fold = 9: k = 1, Accuracy = 26.444
fold = 10: k = 1, Accuracy = 24.058999999999997
k = 1, Average Accuracy = 24.538999999999998
*****Finished*****
```

K = 3

```
fold = 1: k = 1, Accuracy = 24.414
fold = 2: k = 1, Accuracy = 19.308999999999997
fold = 3: k = 1, Accuracy = 27.169
fold = 1: k = 2, Accuracy = 28.018
fold = 2: k = 2, Accuracy = 25.015
fold = 3: k = 2, Accuracy = 27.47
fold = 1: k = 3, Accuracy = 28.018
fold = 2: k = 3, Accuracy = 26.216
fold = 3: k = 3, Accuracy = 28.675
k = 1, Average Accuracy = 23.630666666666663
k = 2, Average Accuracy = 26.834333333333333
```

```
k = 3, Average Accuracy = 27.636333333333337
```

K = 5

```
k = 1, Average Accuracy = 26.6328
k = 2, Average Accuracy = 27.0874
k = 3, Average Accuracy = 26.3798
k = 4, Average Accuracy = 26.0838
k = 5, Average Accuracy = 26.1292
```

K = 9

```
k = 1, Average Accuracy = 27.337222222222223
k = 2, Average Accuracy = 27.231333333333335
k = 3, Average Accuracy = 27.50888888888889
k = 4, Average Accuracy = 26.132222222222218
k = 5, Average Accuracy = 27.184666666666665
k = 6, Average Accuracy = 26.510666666666665
k = 7, Average Accuracy = 27.36
k = 8, Average Accuracy = 27.436555555555557
k = 9, Average Accuracy = 25.809555555555555
```

K = 11

```
k = 1, Average Accuracy = 25.992272727272727
k = 2, Average Accuracy = 26.733363636363638
k = 3, Average Accuracy = 26.052363636363637
k = 4, Average Accuracy = 25.07072727272727
k = 5, Average Accuracy = 25.371454545454547
k = 6, Average Accuracy = 26.393545454545457
k = 7, Average Accuracy = 24.647090909090913
k = 8, Average Accuracy = 26.272909090909096
k = 9, Average Accuracy = 25.77190909090909
k = 10, Average Accuracy = 25.570909090909094
k = 11, Average Accuracy = 26.172454545454546
```

K = 13

```
k = 1, Average Accuracy = 25.864384615384616
k = 2, Average Accuracy = 26.948923076923077
k = 3, Average Accuracy = 25.330076923076923
k = 4, Average Accuracy = 26.665692307692304
k = 5, Average Accuracy = 26.150076923076924
k = 6, Average Accuracy = 26.582384615384612
```

```
k = 7, Average Accuracy = 26.398384615384614
k = 8, Average Accuracy = 26.132384615384613
k = 9, Average Accuracy = 25.931307692307694
k = 10, Average Accuracy = 27.250230769230768
k = 11, Average Accuracy = 26.867230769230765
k = 12, Average Accuracy = 25.681153846153848
k = 13, Average Accuracy = 26.98507692307692
```

K = 15

```
k = 1, Average Accuracy = 26.388466666666666
k = 2, Average Accuracy = 26.288733333333333
k = 3, Average Accuracy = 25.67393333333333
k = 4, Average Accuracy = 26.77733333333333
k = 5, Average Accuracy = 26.304333333333336
k = 6, Average Accuracy = 25.715666666666664
k = 7, Average Accuracy = 26.345266666666664
k = 8, Average Accuracy = 25.831066666666665
k = 9, Average Accuracy = 27.292066666666674
k = 10, Average Accuracy = 27.2358
k = 11, Average Accuracy = 25.860400000000006
k = 12, Average Accuracy = 26.846799999999998
k = 13, Average Accuracy = 27.993933333333338
k = 14, Average Accuracy = 26.962333333333337
k = 15, Average Accuracy = 27.148266666666668
```

K = 17

```
k = 1, Average Accuracy = 26.29464705882353
k = 2, Average Accuracy = 25.90611764705882
k = 3, Average Accuracy = 26.00594117647059
k = 4, Average Accuracy = 26.34458823529412
k = 5, Average Accuracy = 25.805999999999997
k = 6, Average Accuracy = 26.56988235294118
k = 7, Average Accuracy = 26.419588235294118
k = 8, Average Accuracy = 25.80570588235294
k = 9, Average Accuracy = 25.266529411764708
k = 10, Average Accuracy = 26.155117647058823
k = 11, Average Accuracy = 26.018529411764707
k = 12, Average Accuracy = 25.993411764705883
k = 13, Average Accuracy = 26.345764705882356
k = 14, Average Accuracy = 26.85705882352941
k = 15, Average Accuracy = 26.155941176470588
k = 16, Average Accuracy = 25.61829411764706
k = 17, Average Accuracy = 24.765235294117645
```

K = 19

```
k = 1, Average Accuracy = 25.709
k = 2, Average Accuracy = 26.110842105263156
k = 3, Average Accuracy = 25.141473684210524
k = 4, Average Accuracy = 27.26584210526316
k = 5, Average Accuracy = 25.76452631578947
k = 6, Average Accuracy = 25.552894736842106
k = 7, Average Accuracy = 25.519684210526314
k = 8, Average Accuracy = 25.708000000000006
k = 9, Average Accuracy = 25.54310526315789
k = 10, Average Accuracy = 25.062315789473683
k = 11, Average Accuracy = 26.68826315789474
k = 12, Average Accuracy = 26.788789473684208
k = 13, Average Accuracy = 26.52263157894737
k = 14, Average Accuracy = 24.585
k = 15, Average Accuracy = 26.132105263157893
k = 16, Average Accuracy = 24.774842105263154
k = 17, Average Accuracy = 25.843894736842103
k = 18, Average Accuracy = 26.331421052631576
k = 19, Average Accuracy = 26.066315789473688
```

K = 21

```
k = 1, Average Accuracy = 26.65704761904762
k = 2, Average Accuracy = 27.29442857142857
k = 3, Average Accuracy = 25.20795238095238
k = 4, Average Accuracy = 25.547857142857143
k = 5, Average Accuracy = 26.502999999999997
k = 6, Average Accuracy = 25.918285714285716
k = 7, Average Accuracy = 27.8252380952381
k = 8, Average Accuracy = 25.36509523809524
k = 9, Average Accuracy = 26.539095238095236
k = 10, Average Accuracy = 25.67847619047619
k = 11, Average Accuracy = 25.033190476190473
k = 12, Average Accuracy = 25.88995238095238
k = 13, Average Accuracy = 25.621285714285712
k = 14, Average Accuracy = 25.467333333333336
k = 15, Average Accuracy = 26.42990476190476
k = 16, Average Accuracy = 26.61995238095238
k = 17, Average Accuracy = 25.619333333333333
k = 18, Average Accuracy = 27.03904761904762
k = 19, Average Accuracy = 26.078999999999997
k = 20, Average Accuracy = 26.65085714285714
k = 21, Average Accuracy = 25.76004761904762
```

Metrics

Method	Average Time per image in seconds	Total Time in Seconds
Histogram Segmentation	4.143223947895792	2068.328125
Contours	7.9309556613226455	5838.796875
Classification & K-nn	0.9397858216432866	5906.921875
Total	13.01396543	13814.04688

The average time per image in seconds: 13.01396543

The total Process time in seconds: 13814.04688