

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 may 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/*

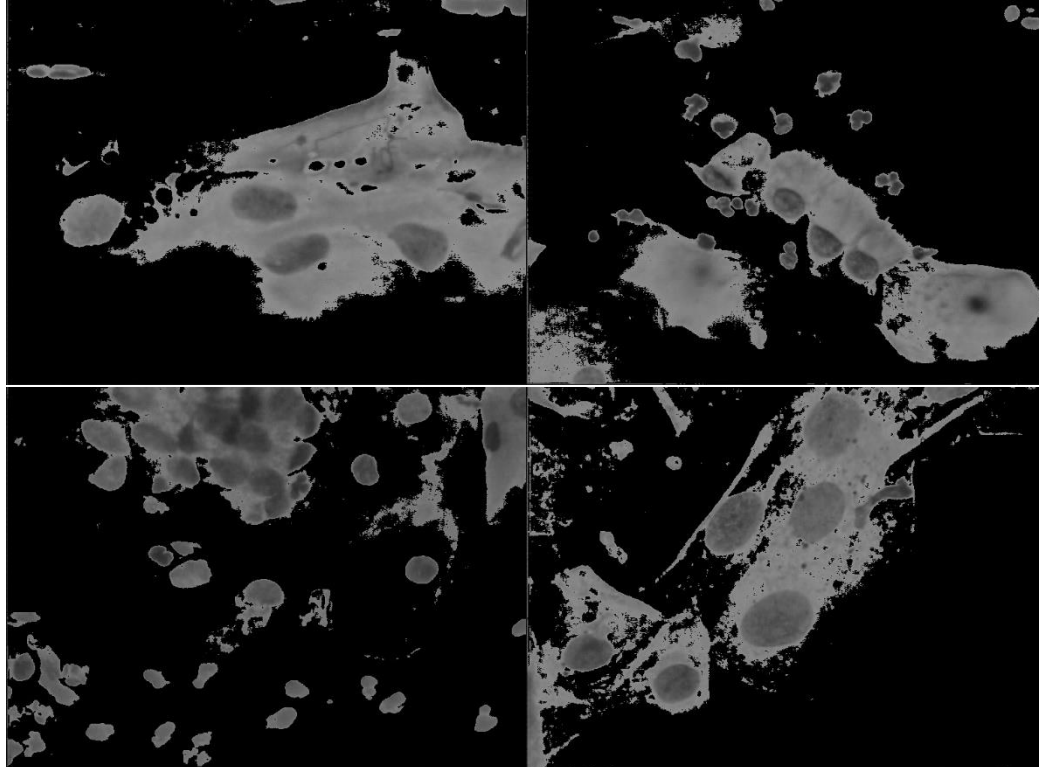
The output path must complete path to the directory to store the images:

- C:/Users/user/Documents/output

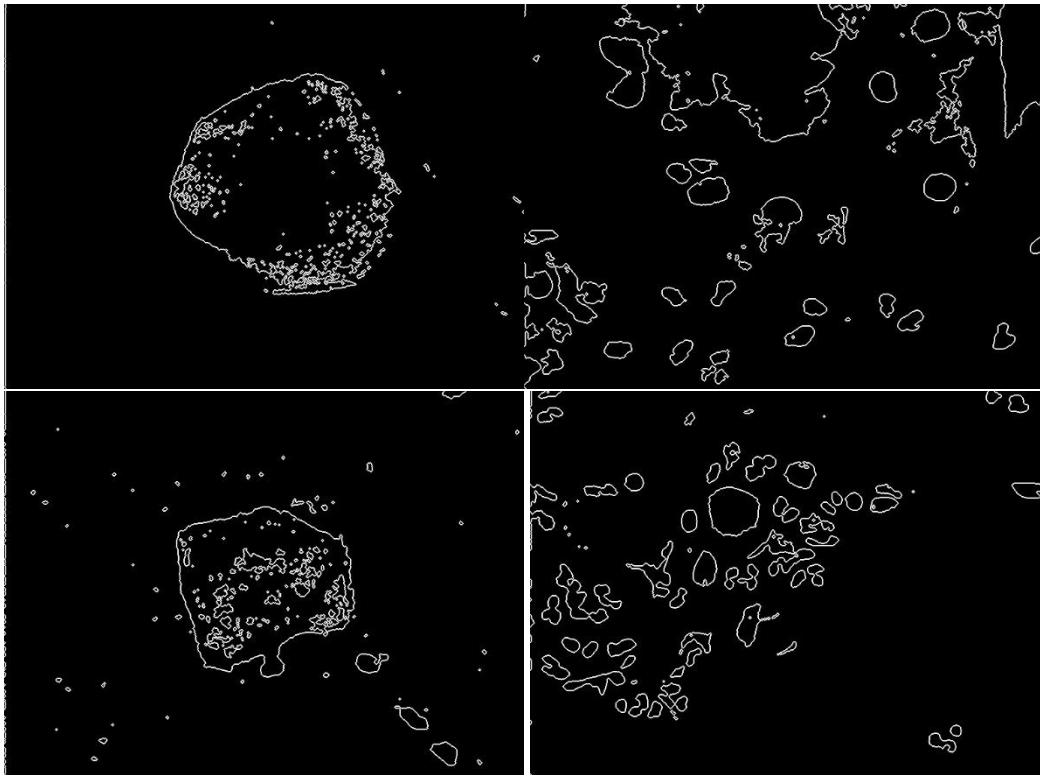
Image Segmentation

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

Histogram Segmentation – This approach allowed me cluster important 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.



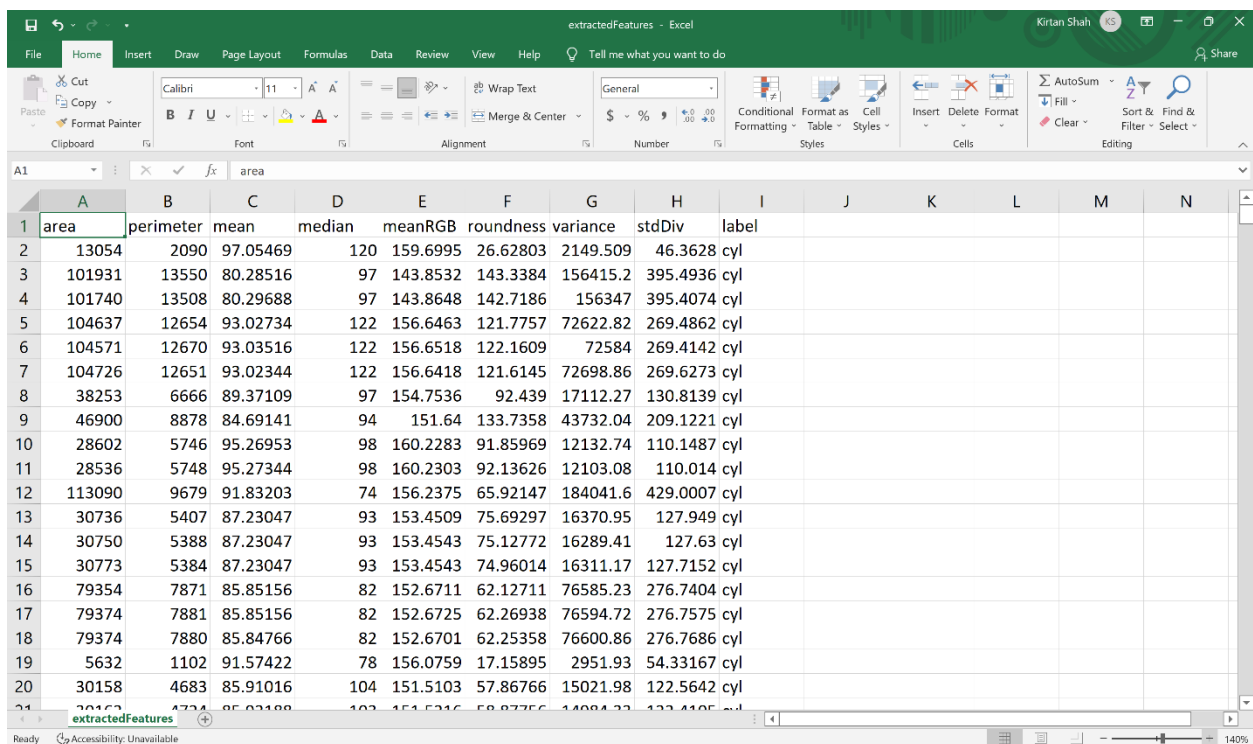
Contours – To find the contours of the cells within the image. I first eroded the image then dilated the eroded image and then subtracted the dilated by the eroded to just give the outline of the cells within.



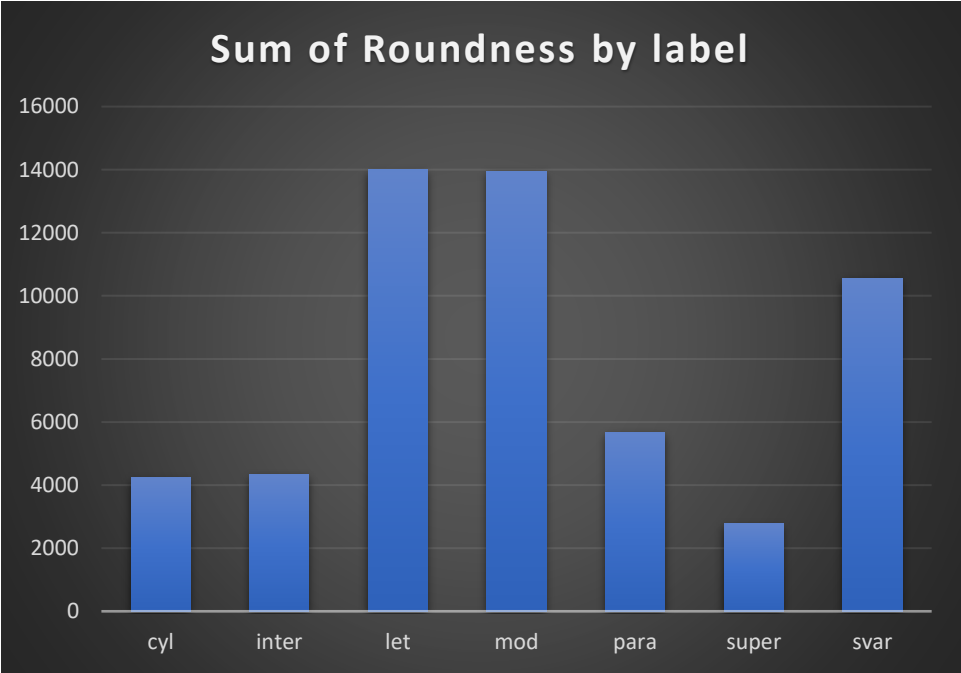
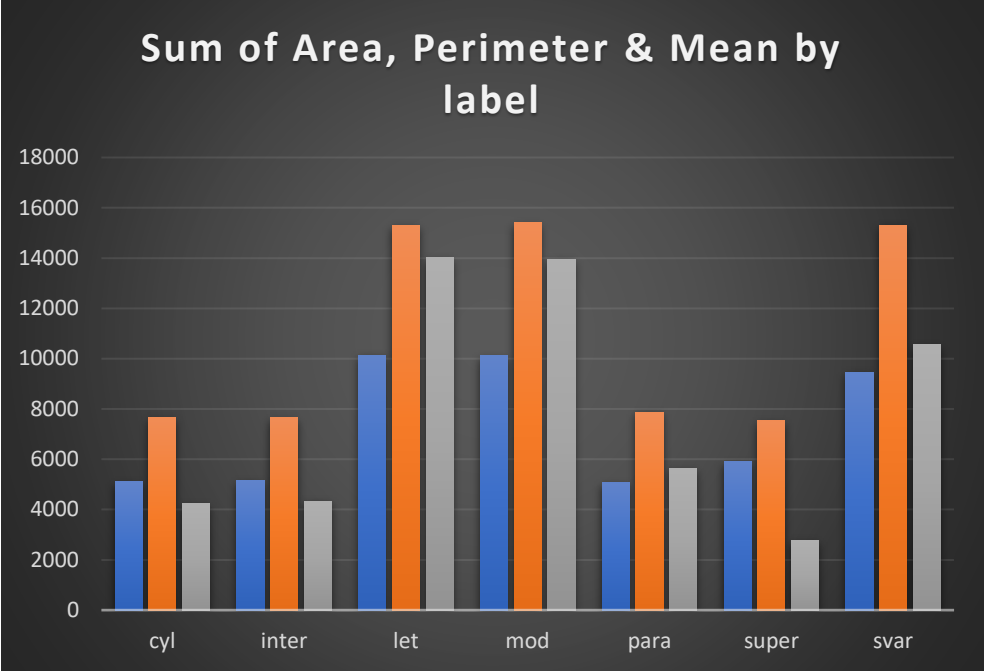
Feature Extraction

Many features can be extracted from images. Features that can help identify important aspects within an image. You can view the all extracted features in extractedFeatures.csv and find them [here](#). The Bar graphs display some stats from the data collected. 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:** $(\text{perimeter} * \text{perimeter}) / 4 * \text{math.pi} * \text{Area}$
- **Standard Deviation:** the square root of the variance
- **Variance:** the object variance used to during the histogram segmentation



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	area	perimeter	mean	median	meanRGB	roundness	variance	stdDiv	label					
2	13054	2090	97.05469	120	159.6995	26.62803	2149.509	46.3628	cyl					
3	101931	13550	80.28516	97	143.8532	143.3384	156415.2	395.4936	cyl					
4	101740	13508	80.29688	97	143.8648	142.7186	156347	395.4074	cyl					
5	104637	12654	93.02734	122	156.6463	121.7757	72622.82	269.4862	cyl					
6	104571	12670	93.03516	122	156.6518	122.1609	72584	269.4142	cyl					
7	104726	12651	93.02344	122	156.6418	121.6145	72698.86	269.6273	cyl					
8	38253	6666	89.37109	97	154.7536	92.439	17112.27	130.8139	cyl					
9	46900	8878	84.69141	94	151.64	133.7358	43732.04	209.1221	cyl					
10	28602	5746	95.26953	98	160.2283	91.85969	12132.74	110.1487	cyl					
11	28536	5748	95.27344	98	160.2303	92.13626	12103.08	110.014	cyl					
12	113090	9679	91.83203	74	156.2375	65.92147	184041.6	429.0007	cyl					
13	30736	5407	87.23047	93	153.4509	75.69297	16370.95	127.949	cyl					
14	30750	5388	87.23047	93	153.4543	75.12772	16289.41	127.63	cyl					
15	30773	5384	87.23047	93	153.4543	74.96014	16311.17	127.7152	cyl					
16	79354	7871	85.85156	82	152.6711	62.12711	76585.23	276.7404	cyl					
17	79374	7881	85.85156	82	152.6725	62.26938	76594.72	276.7575	cyl					
18	79374	7880	85.84766	82	152.6701	62.25358	76600.86	276.7686	cyl					
19	5632	1102	91.57422	78	156.0759	17.15895	2951.93	54.33167	cyl					
20	30158	4683	85.91016	104	151.5103	57.86766	15021.98	122.5642	cyl					
21	30163	4724	85.92198	103	151.5316	58.87766	14984.33	122.4195	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 be used in conjunction to classify the image. I started the 10 fold cross validation. By splitting the data into 2 separate 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. You can find the results [here](#).

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 = 26.83433333333333
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 = 27.508888888888889
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 = 25.673933333333333
k = 4, Average Accuracy = 26.777333333333333
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
Feature Extraction	0.9397858216432866	5906.921875
Classification & K-nn		1.24658355
Total	13.01396543	13814.04688

The average time per image in seconds: 13.01396543

The total Process time in seconds: 13814.04688