all ImageSegmentation.java methods applied to a few pairs of images that are panoramic sets or stereo image sets with the goal of finding best segmentation for finding blobs to make matchable contours.

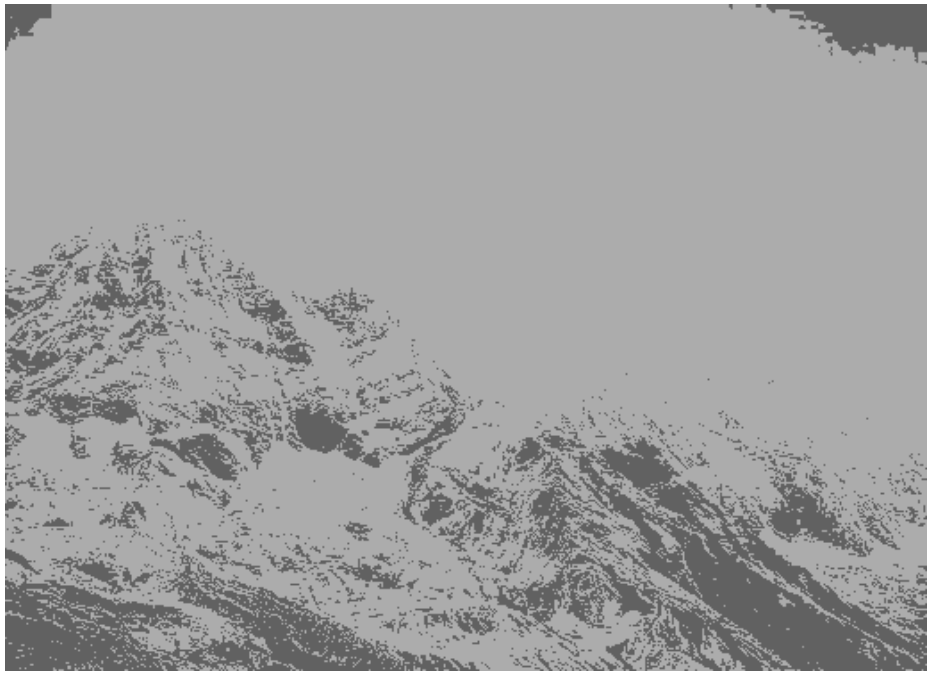
summary of next pages:

- for brown & lowe 2003, best was KMPP w/ k=2
- for Venturi, best was PolarCIEXYAndFrequency
- for books, best was KMPP w/ k=2, 3 or 8

Venturi has lots of texture, so attempting a low resolution segmentation first is faster (the polar ciexy and frequency segmentation is O(N)… check that).

Books best matchable features w/o illumination and projection differences are the text. The text needs further processing such as adaptive mean thresholding.

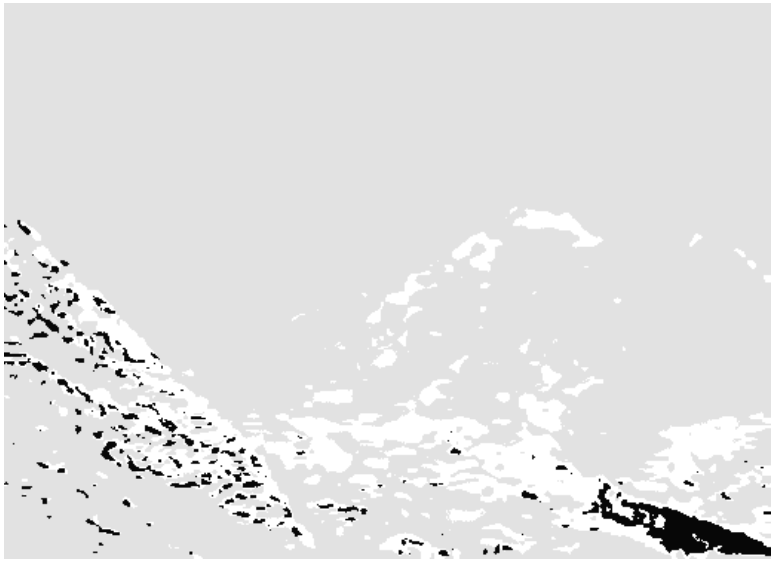
```
int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

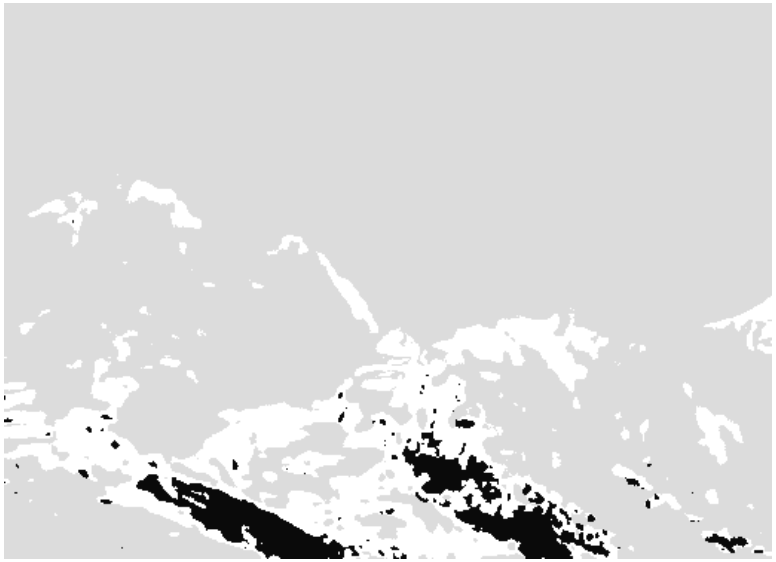


```
int kBands = 3; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

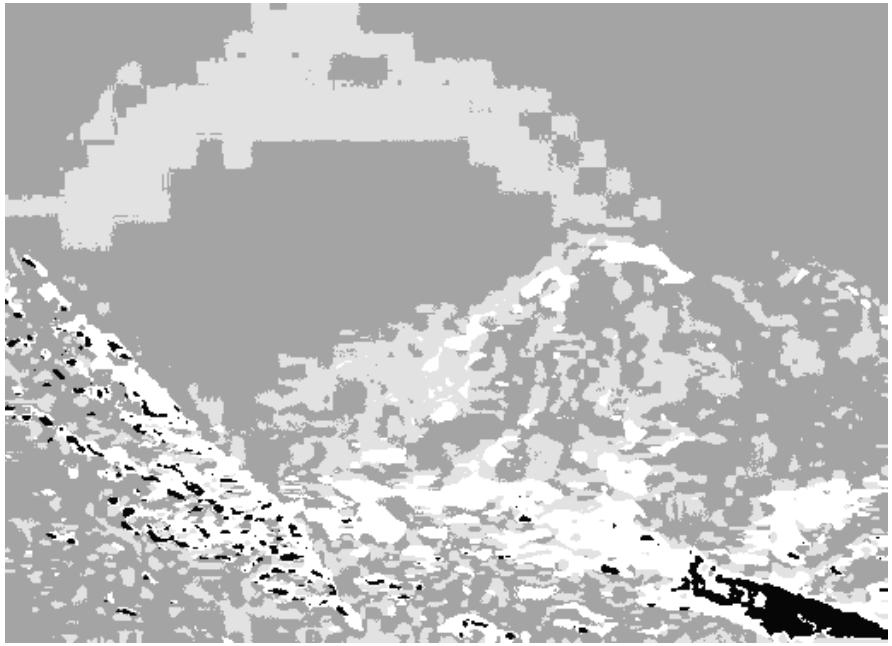
int kBands = 8; imageSegmentation.applyUsingKMPP(gsImg1, kBands);

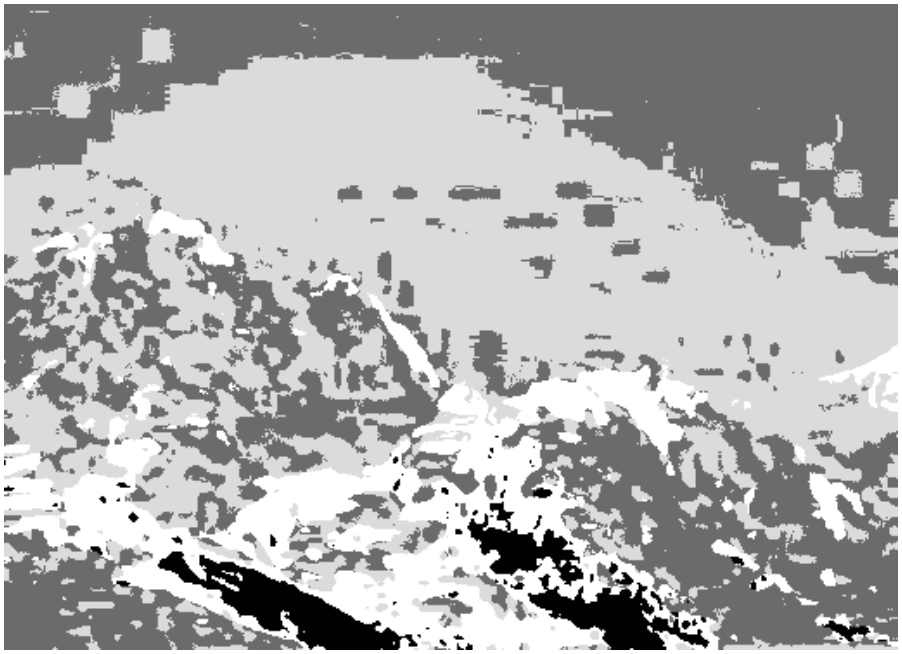
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);



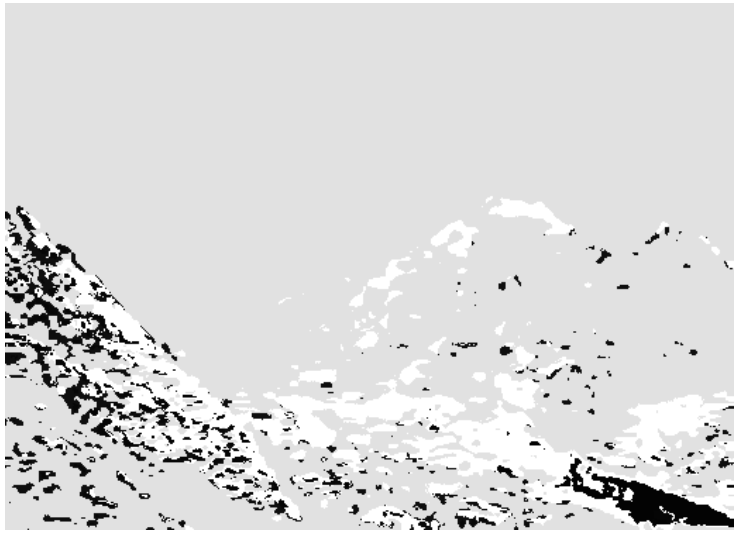


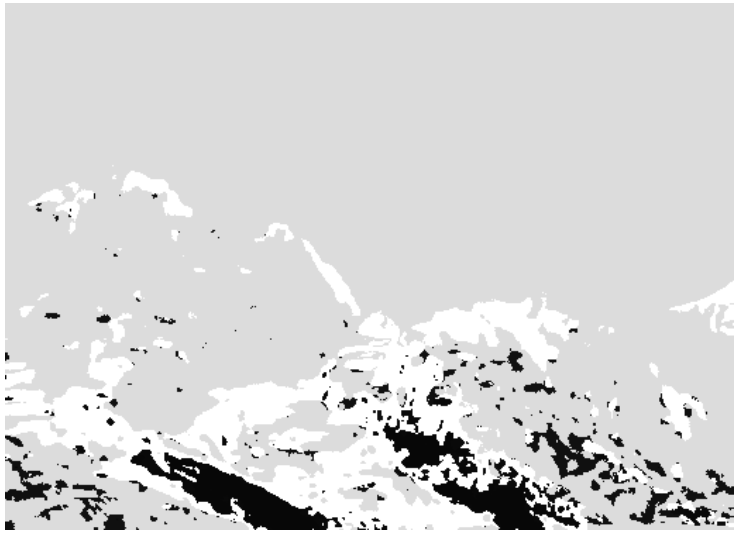
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);



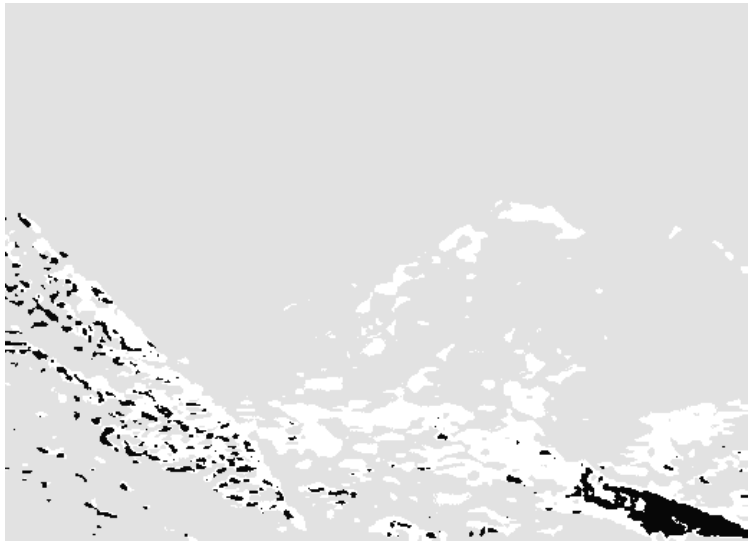


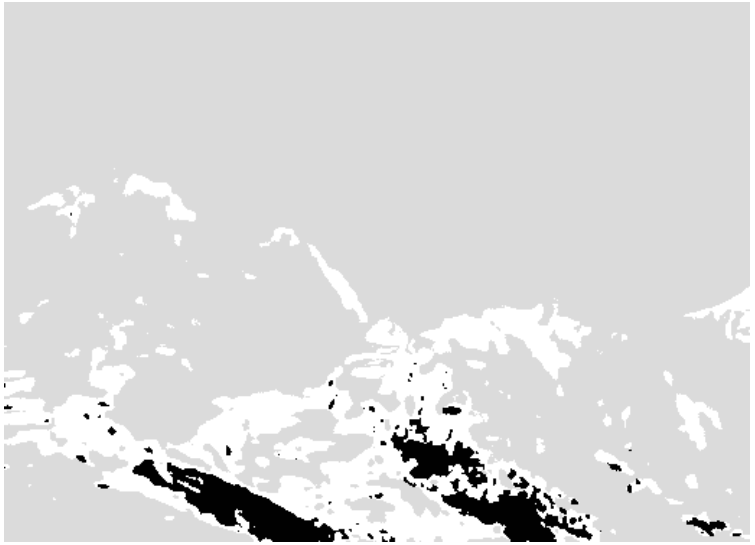
int kBands =8; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);



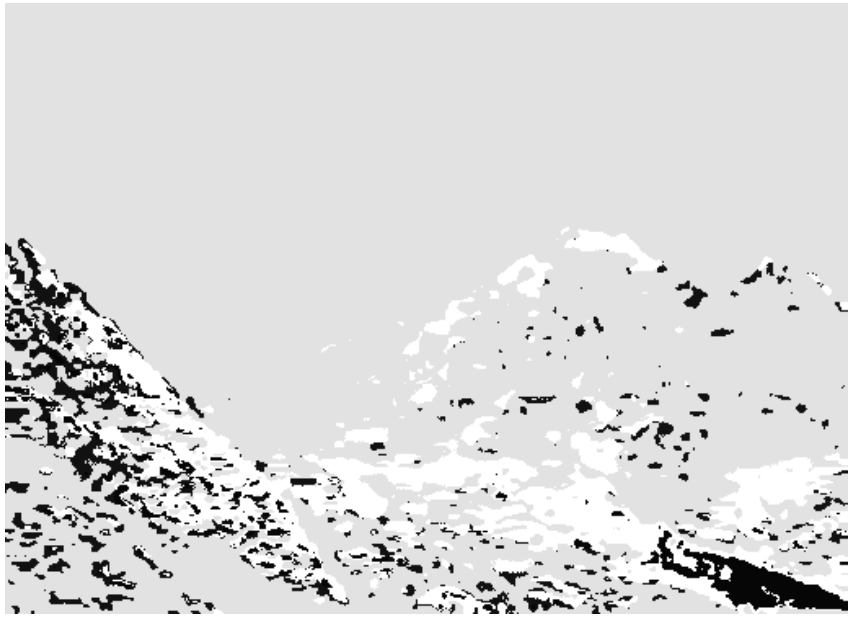


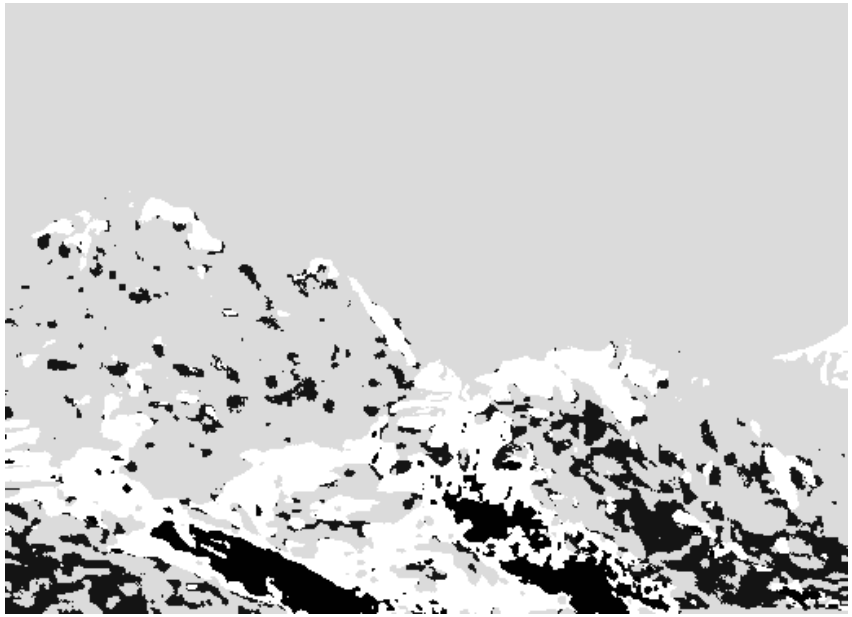
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);
```



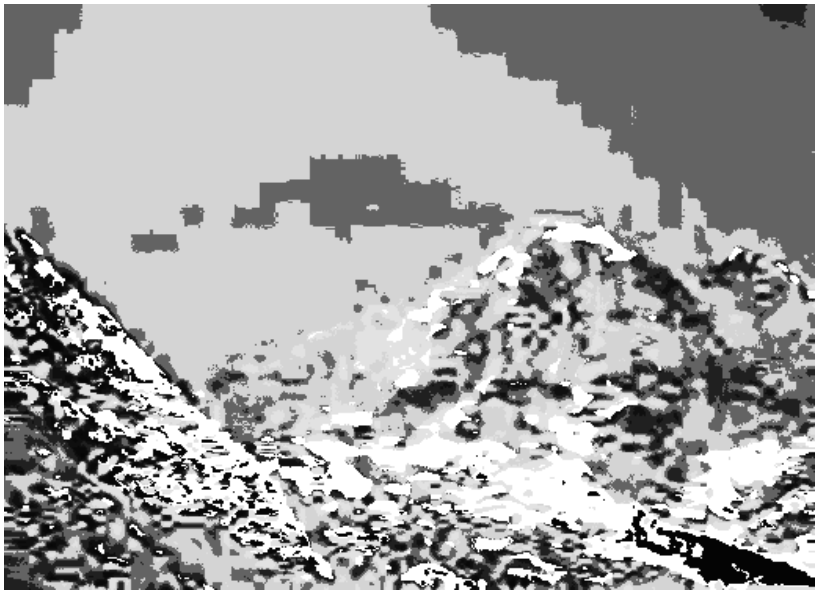


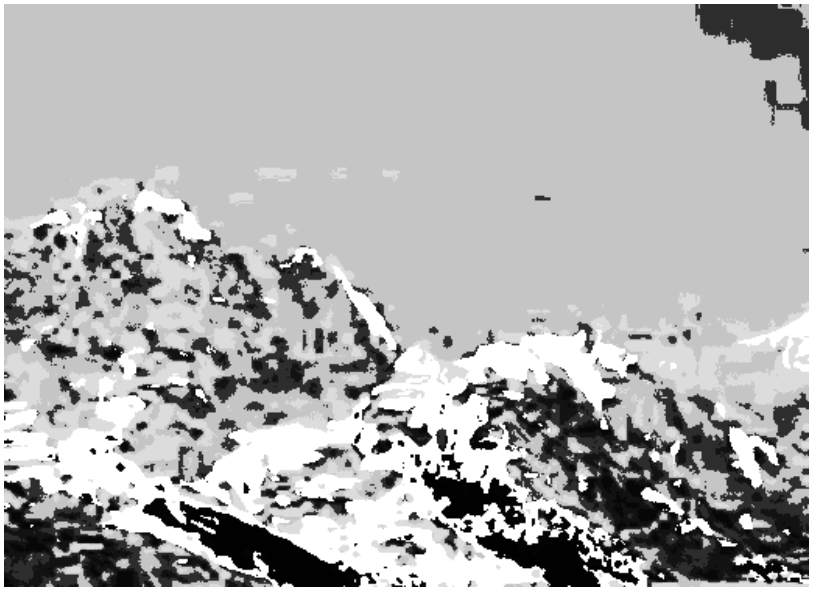
```
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);
```



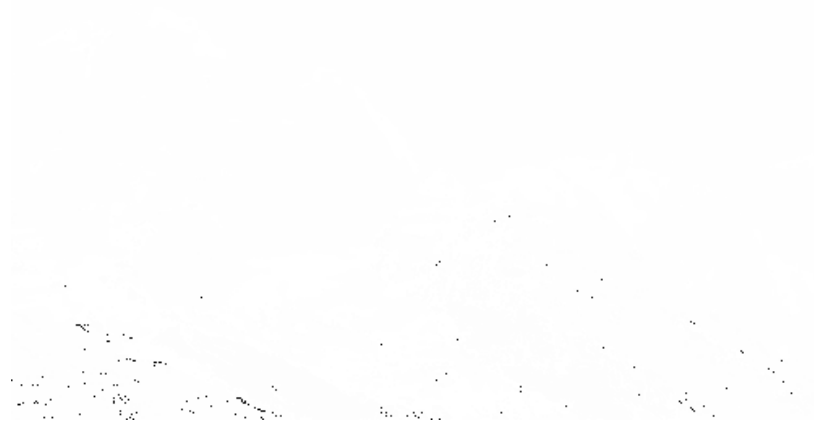


```
int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);
```

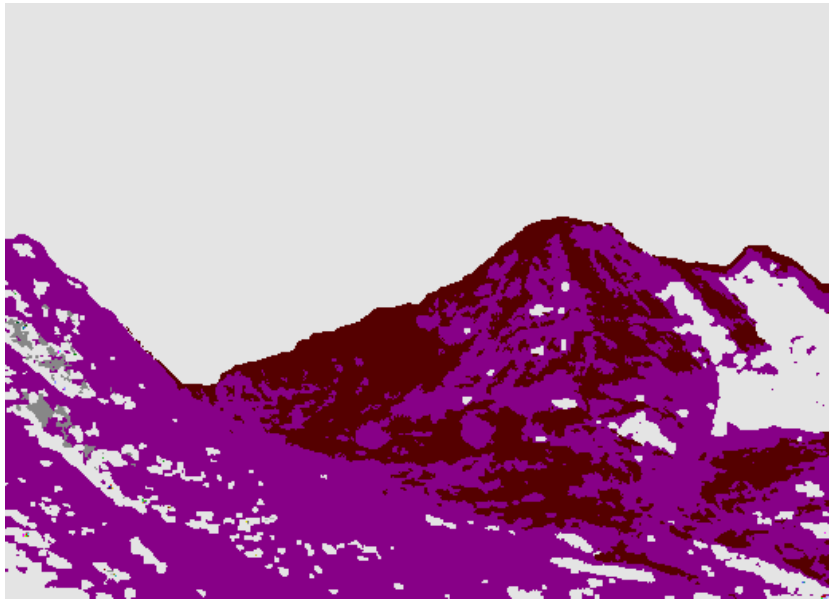


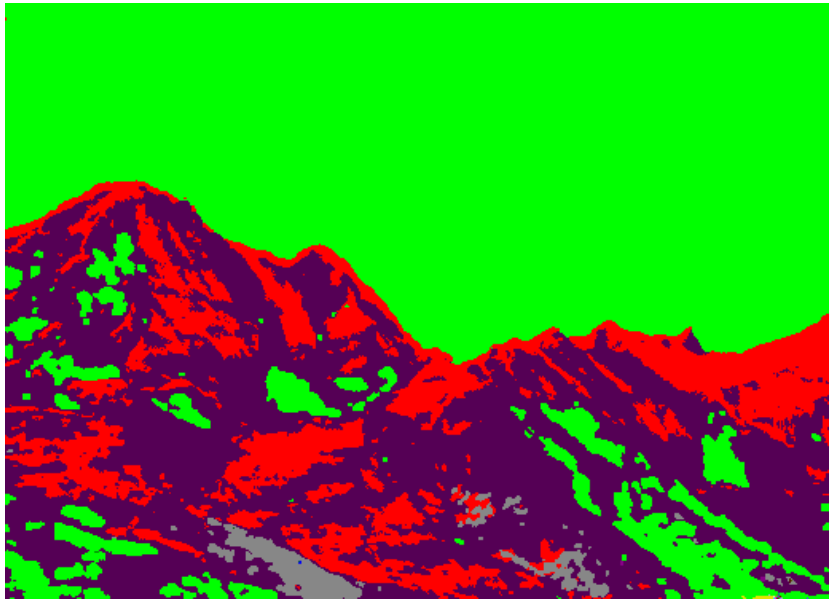


```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistogram(gsImg1, kBands);
```

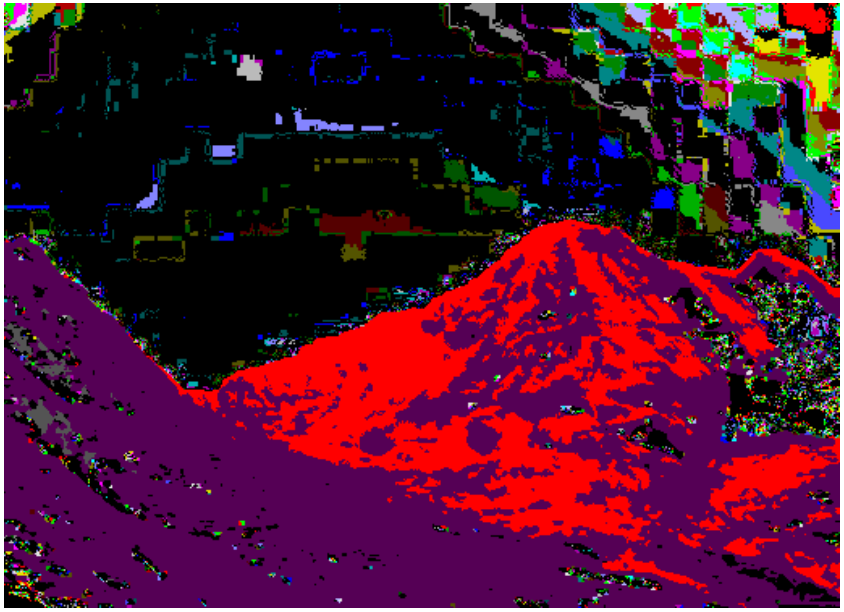


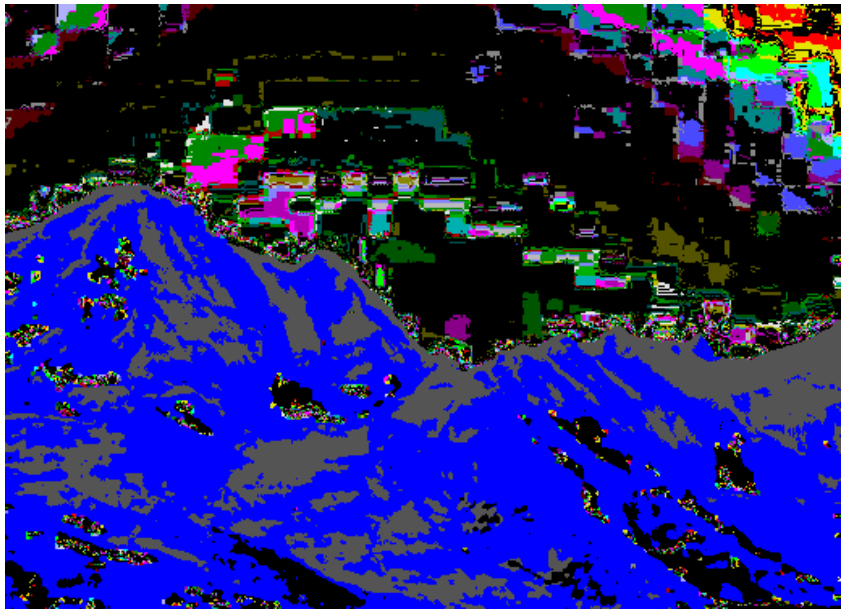
imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);



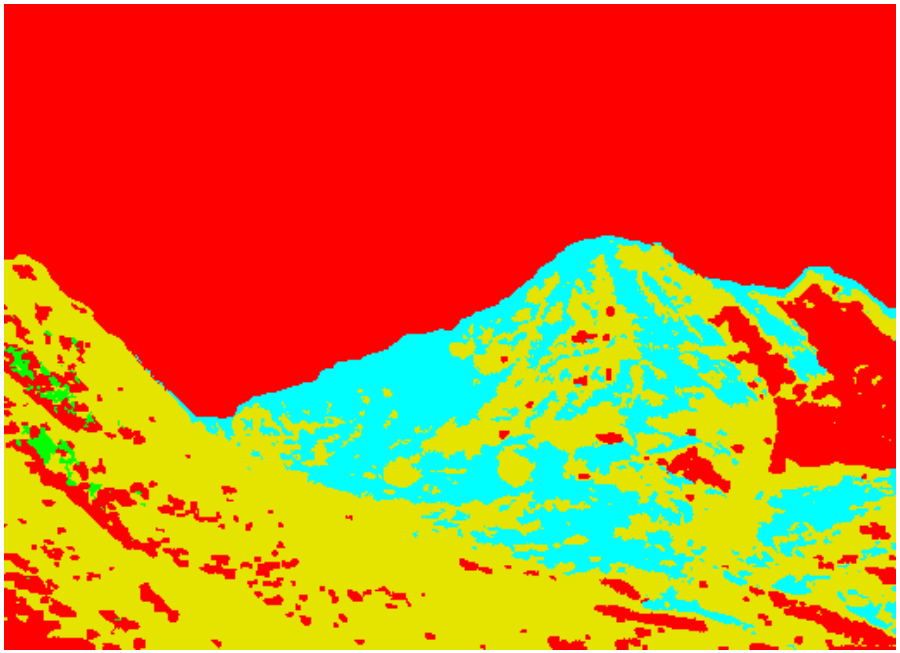


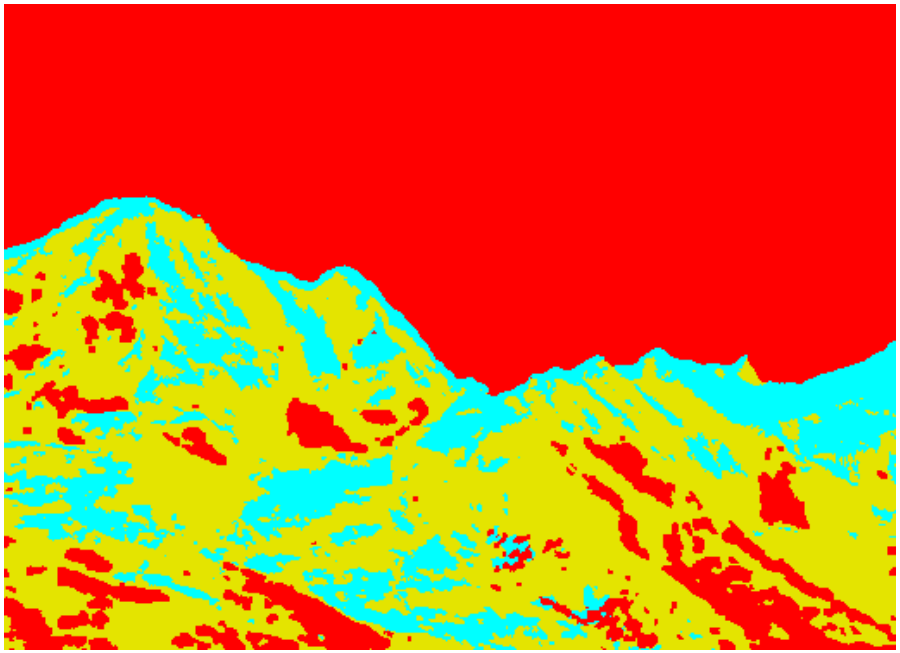
imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);



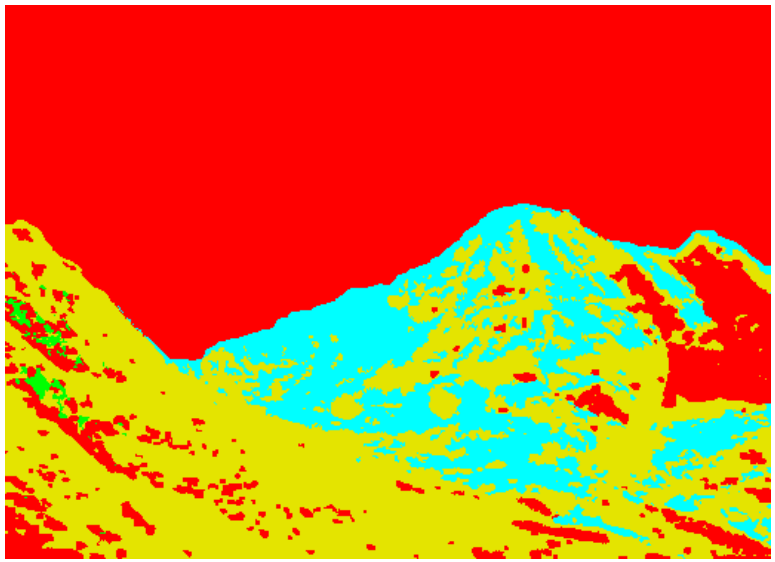


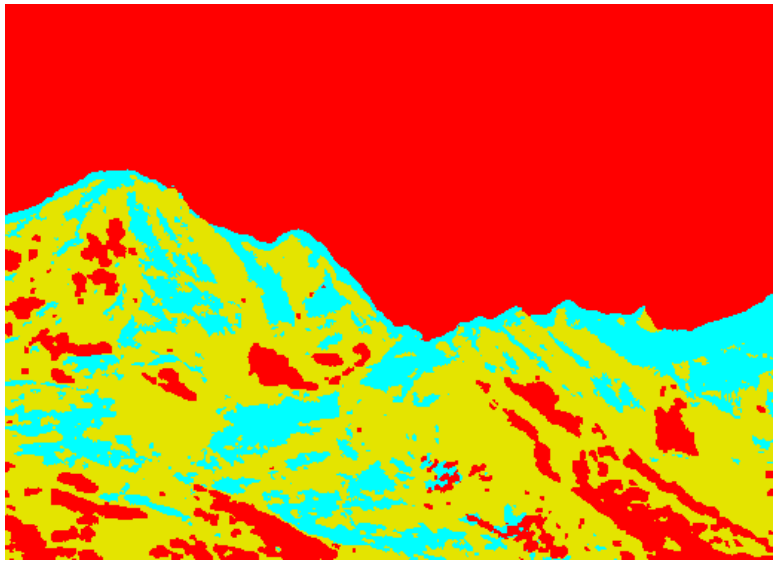
```
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, true);
```





```
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, 0.2f, true);
```





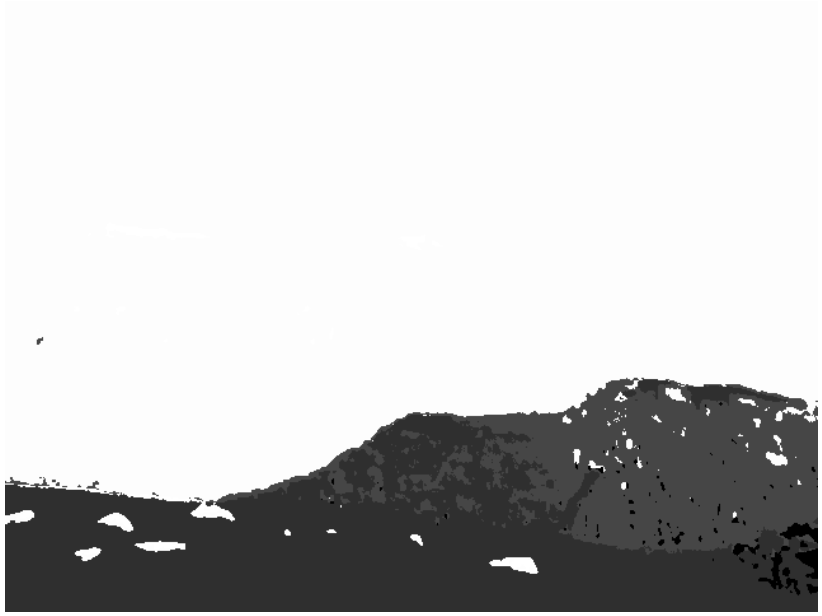
```
int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

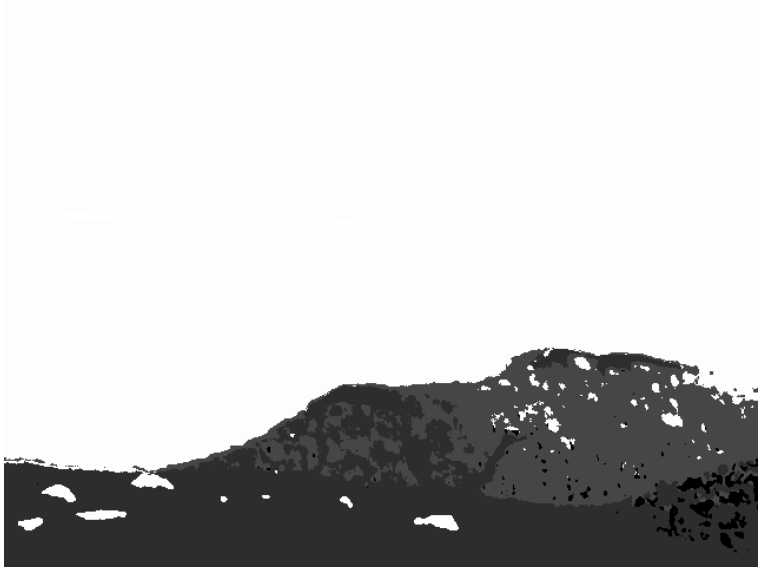
```
int kBands = 3; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

```
int kBands = 8; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

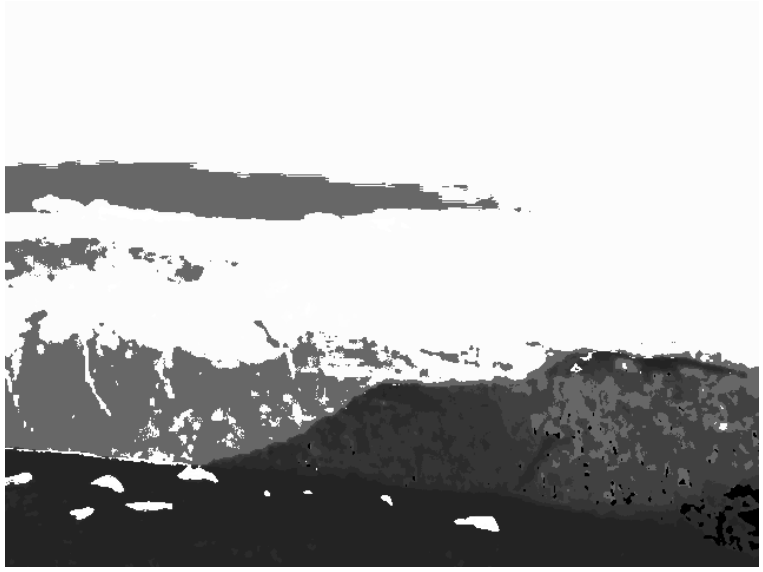
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);
```

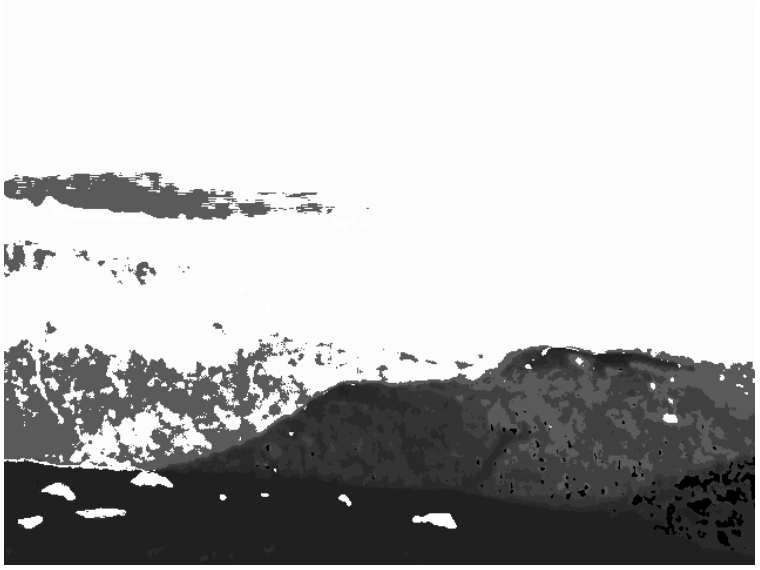
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);





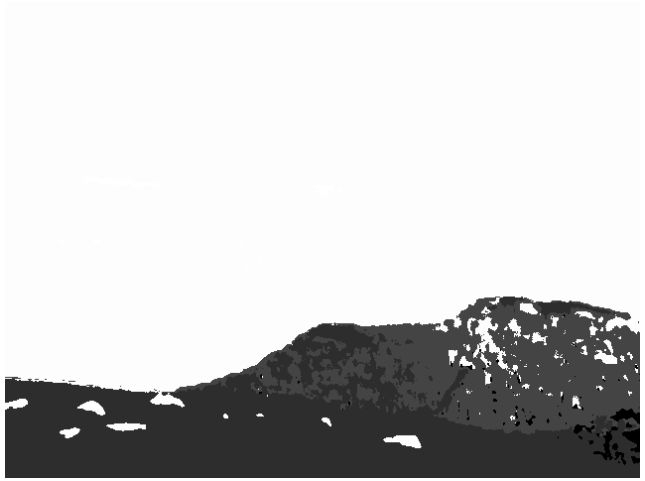
int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);

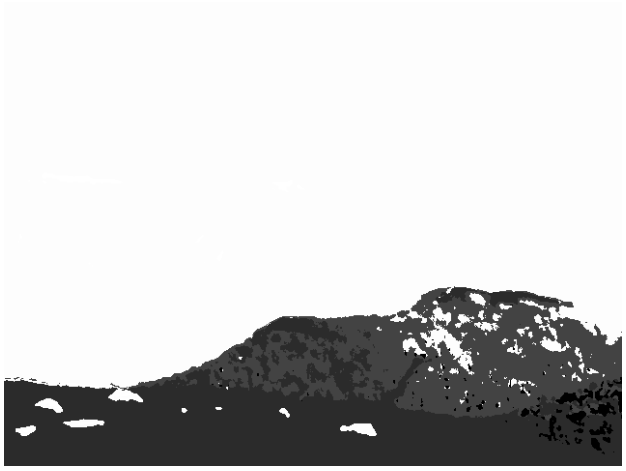




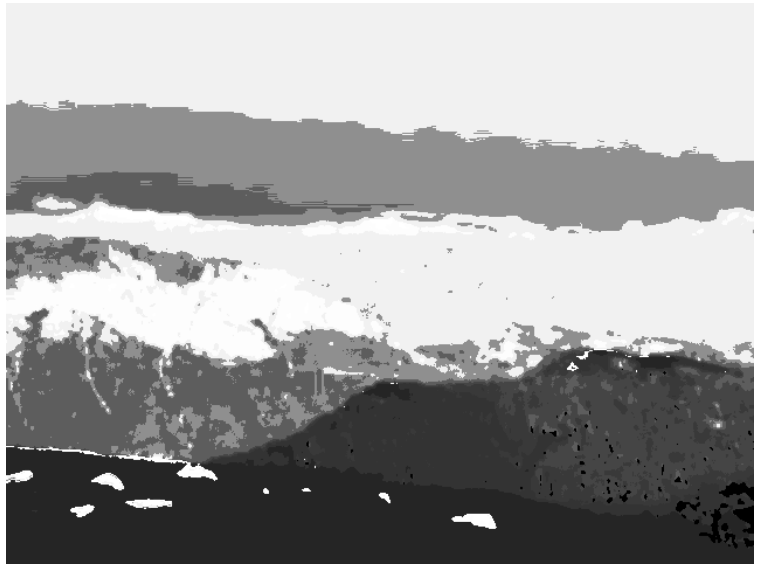
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

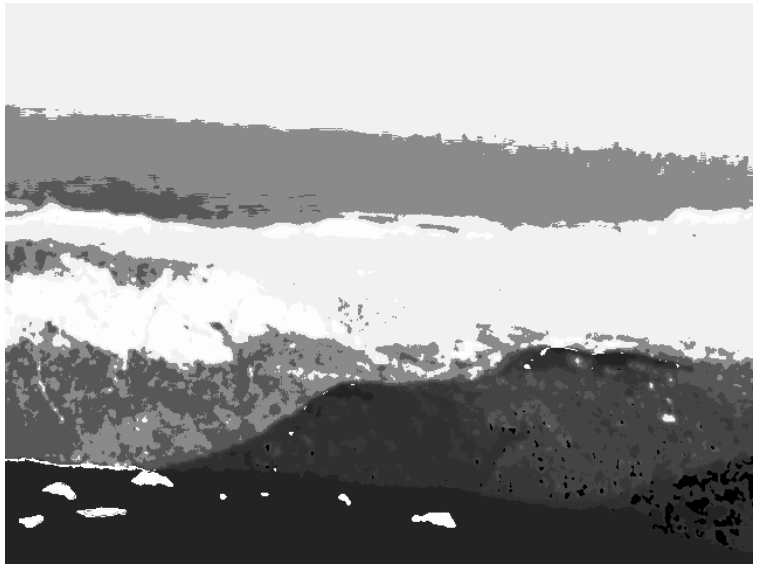
int kBands =3; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);





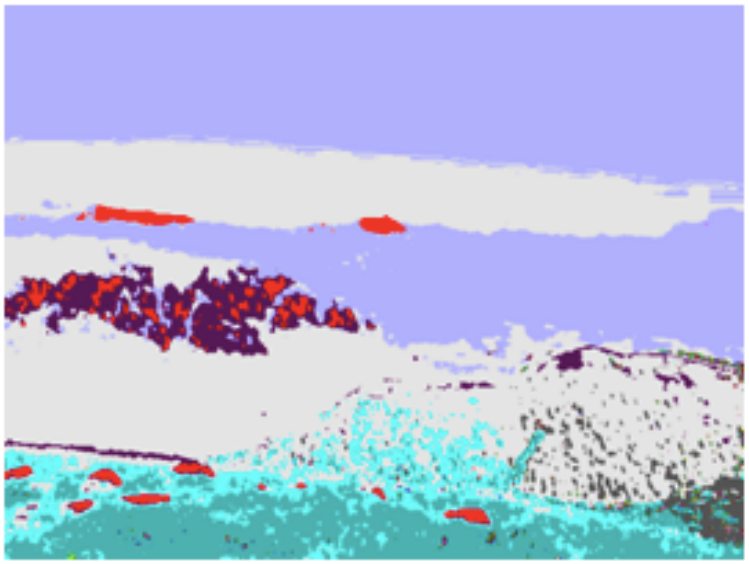
int kBands =8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq

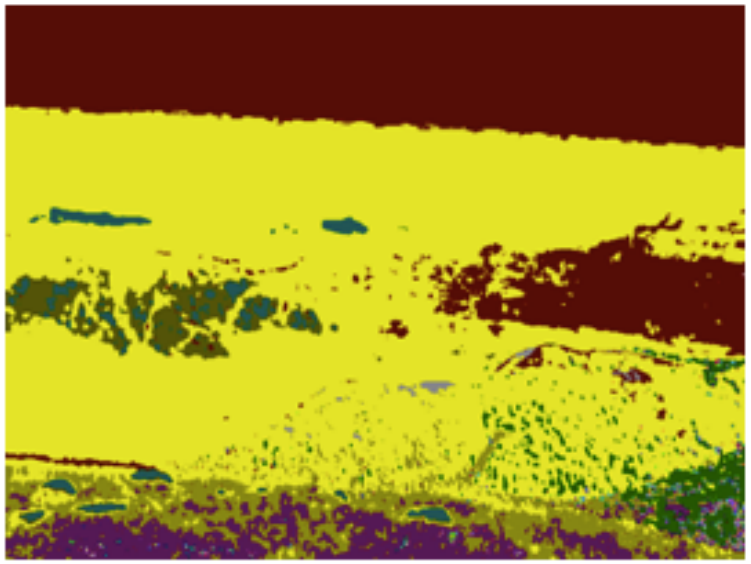




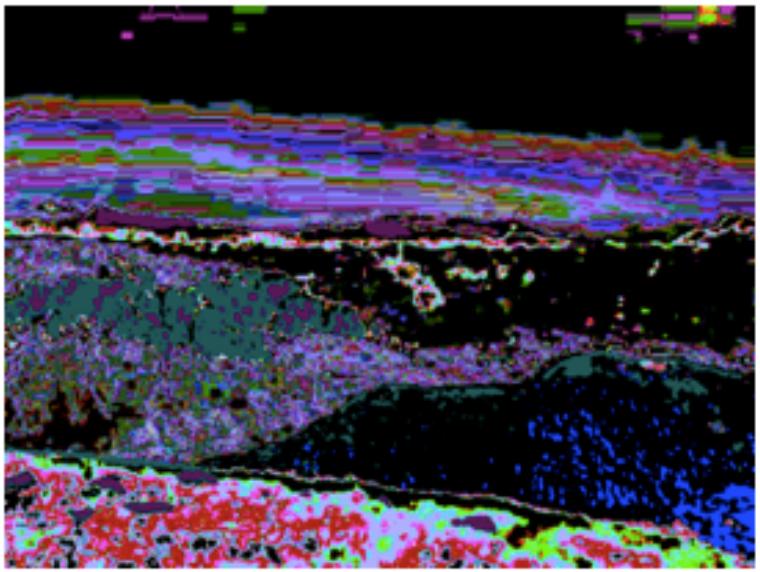
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistogram(gsImg1, kBands);

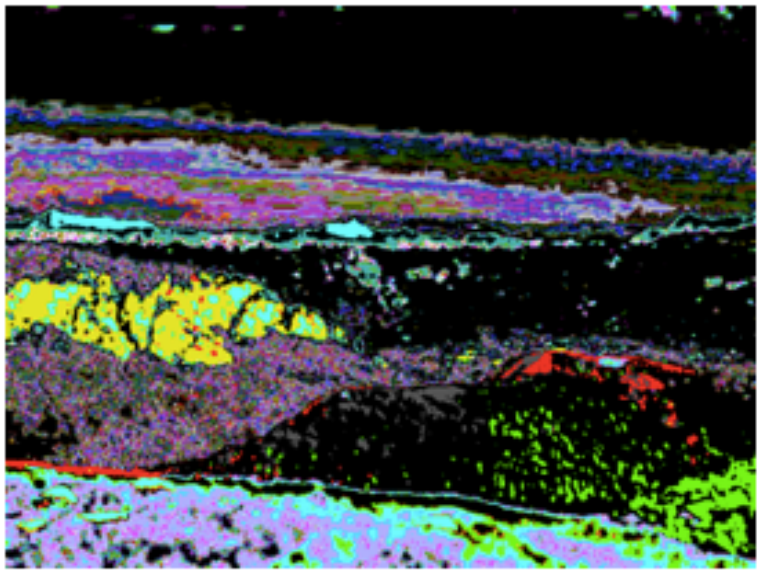
imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);



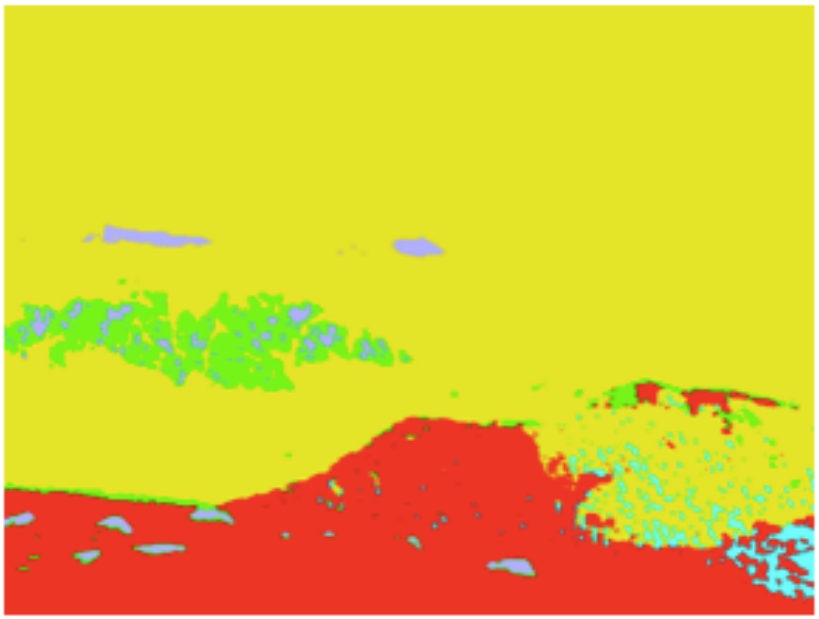


imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);

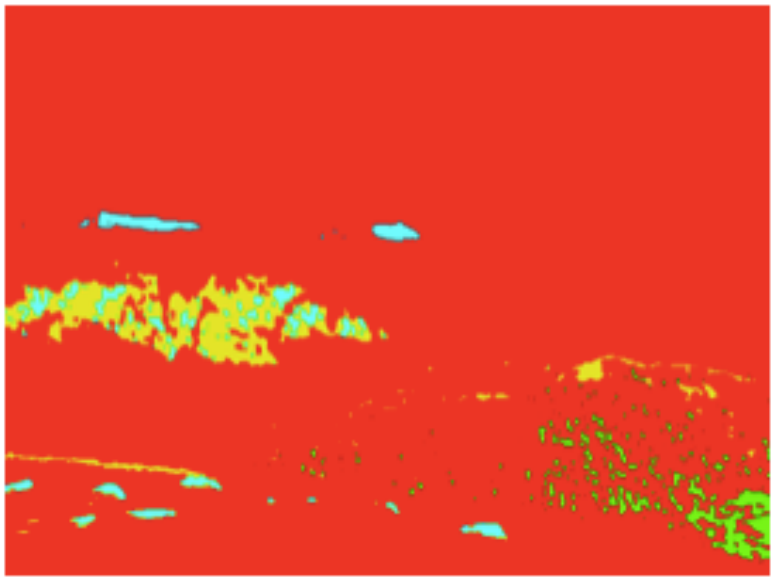


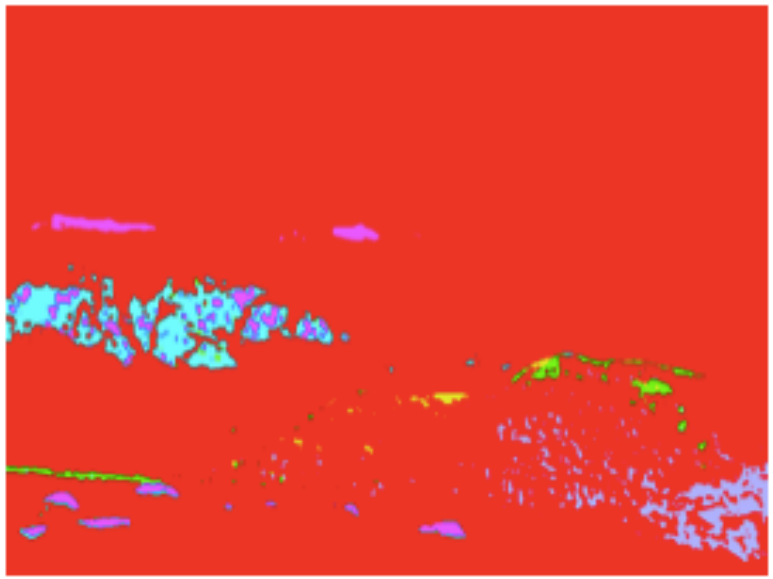


imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, true);



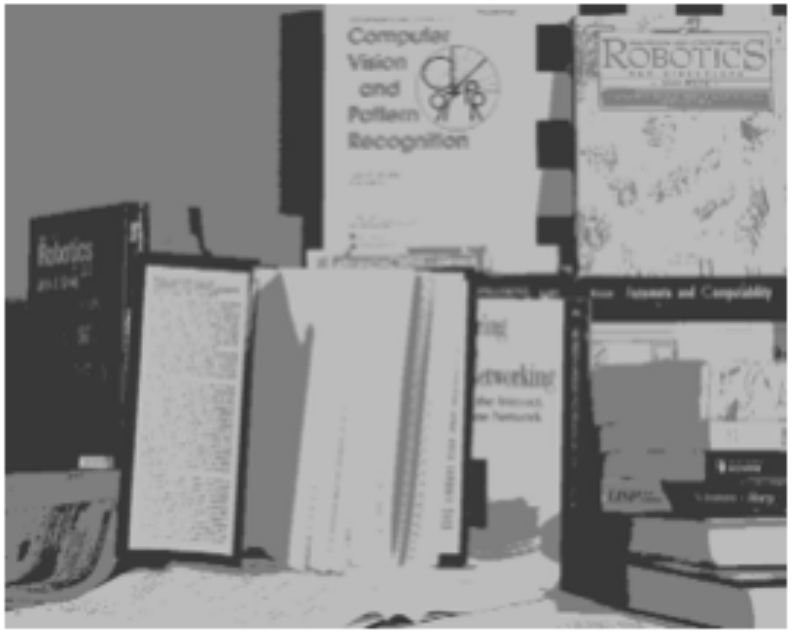
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, 0.2f, true);

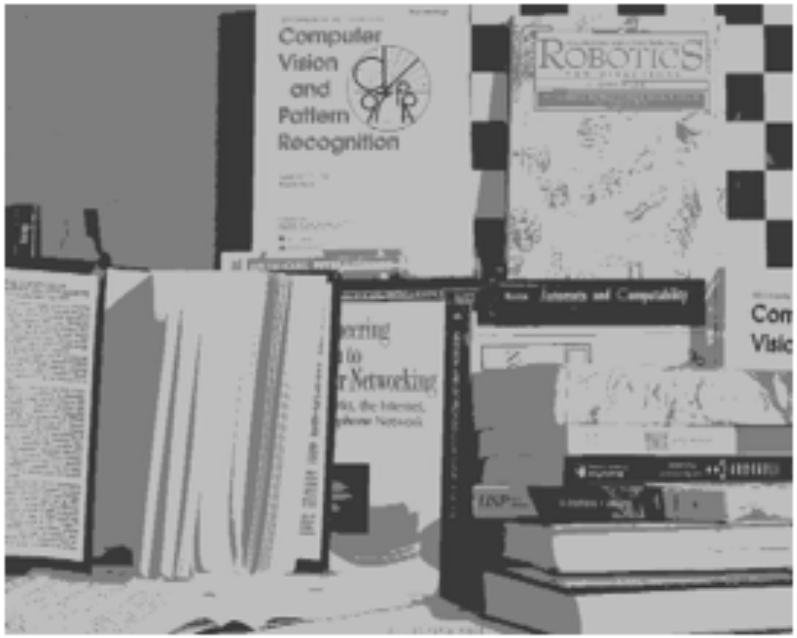




int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);

int kBands = 3; imageSegmentation.applyUsingKMPP(gsImg1, kBands);



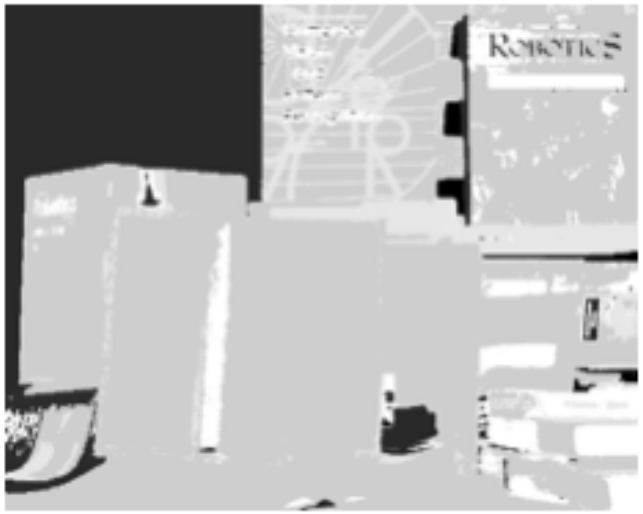


int kBands = 8; imageSegmentation.applyUsingKMPP(gsImg1, kBands);

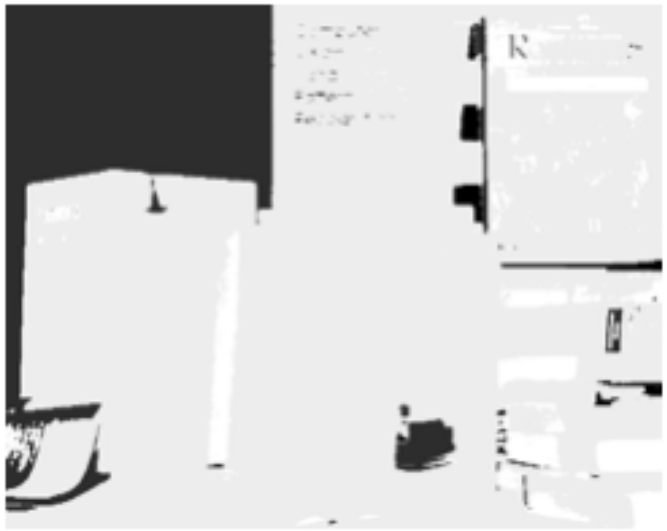
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);

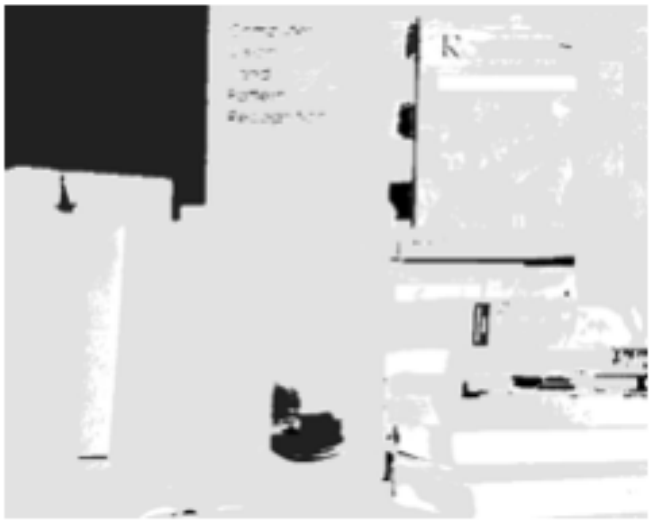
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);

int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);

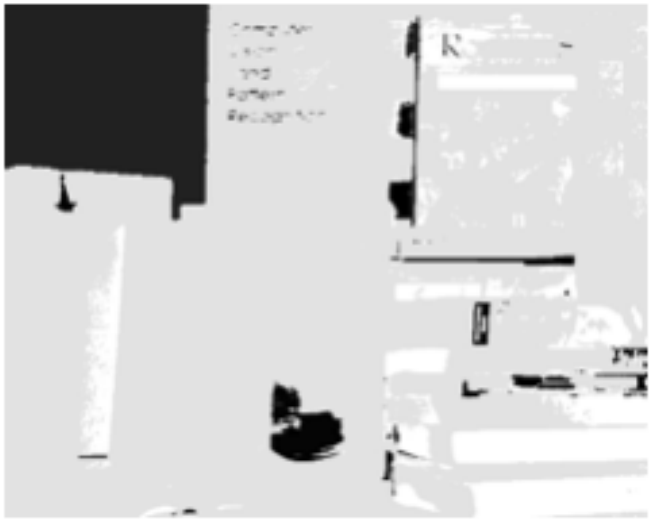


int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

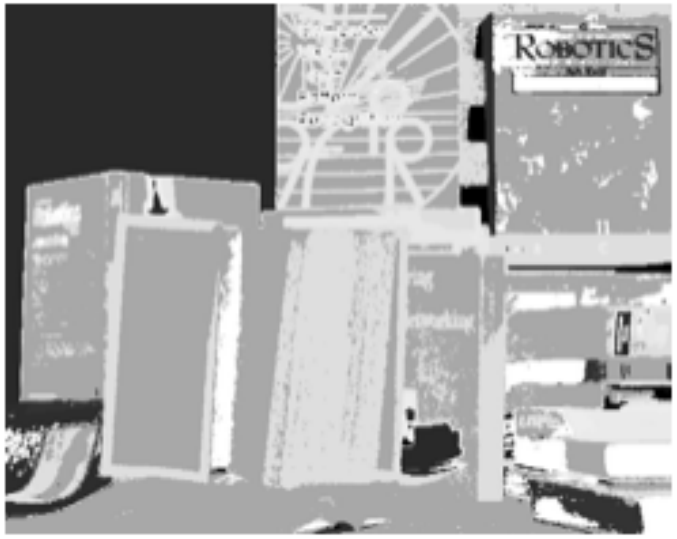


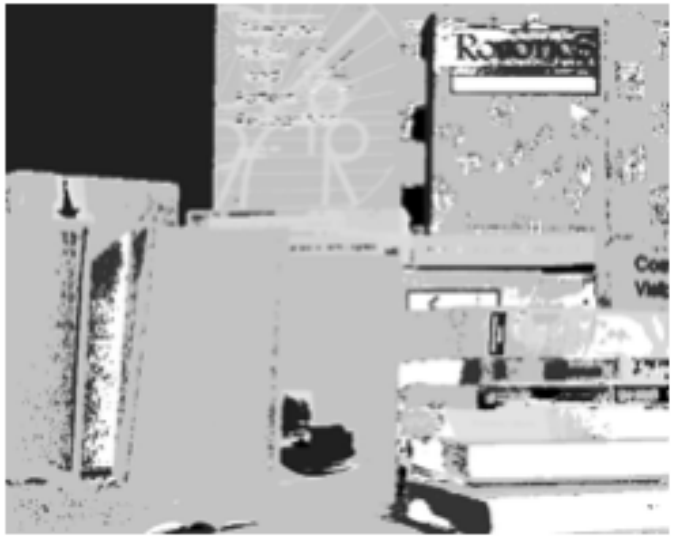


int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

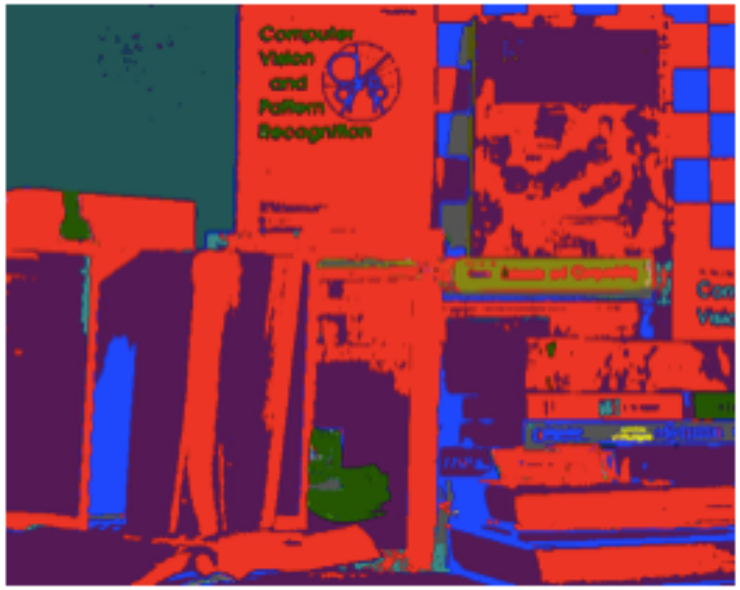


int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

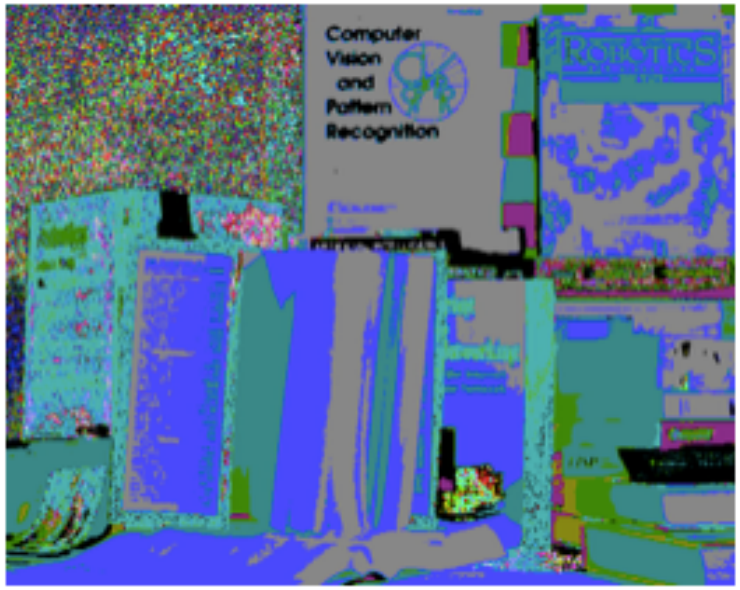


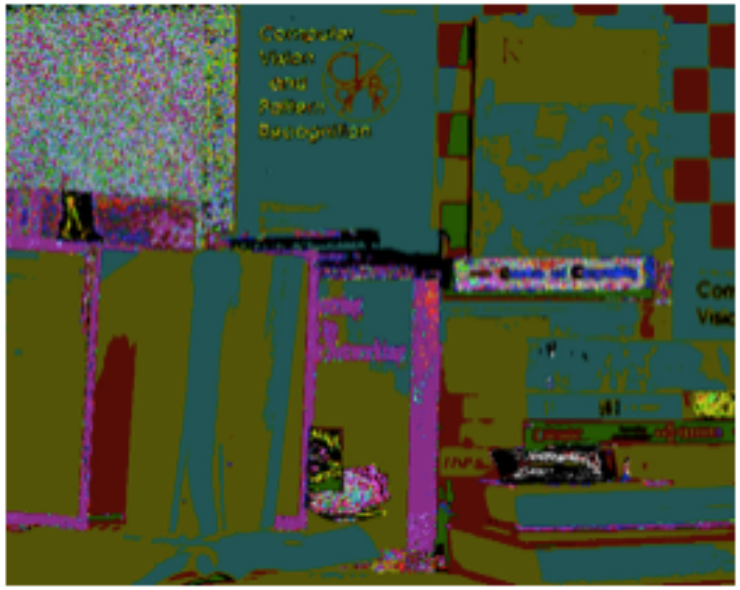


imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);

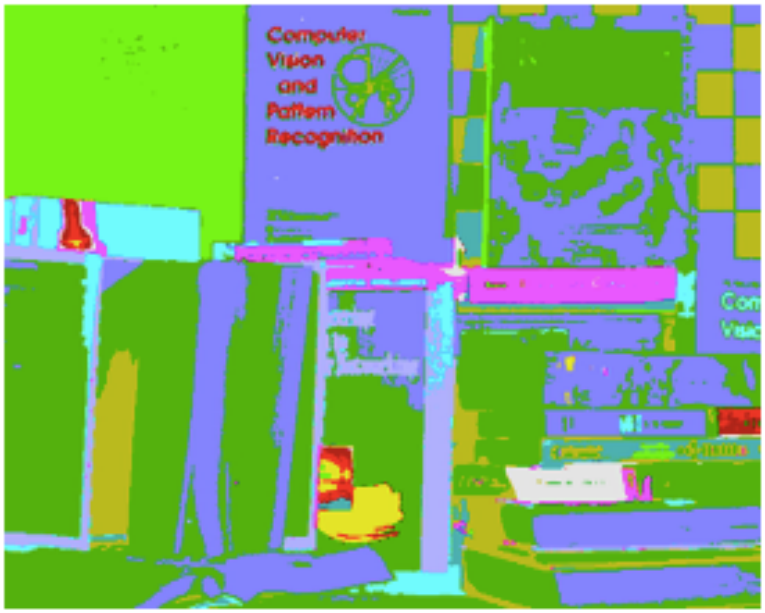


imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);

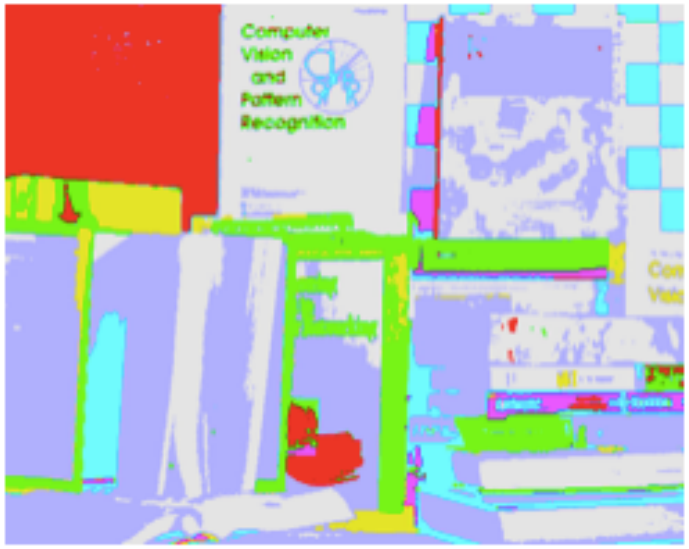




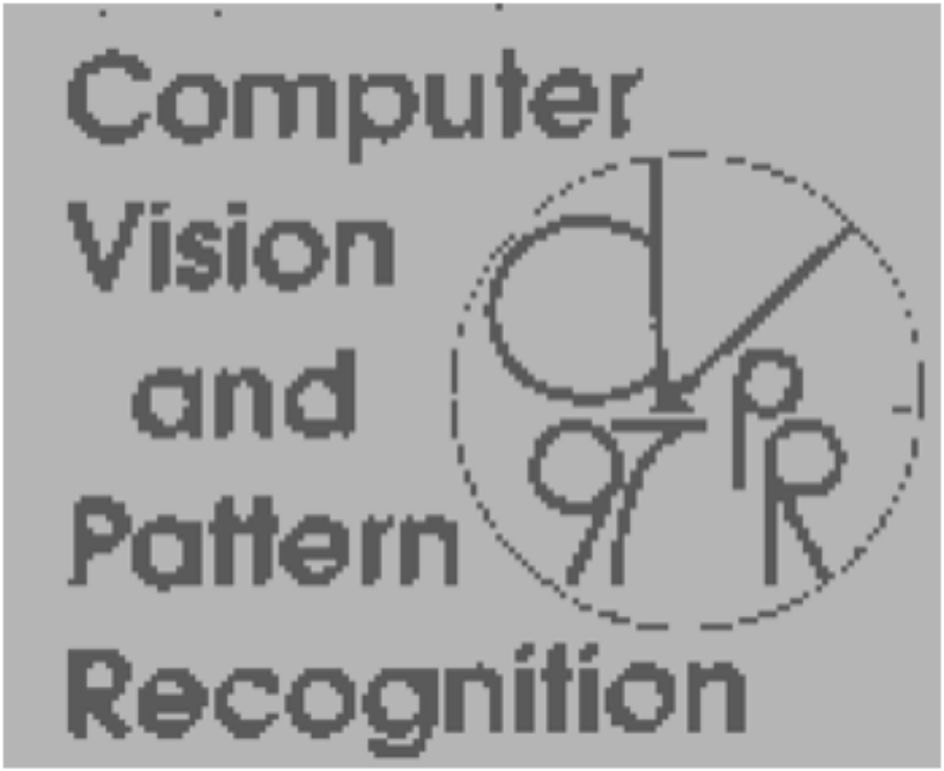
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, true);

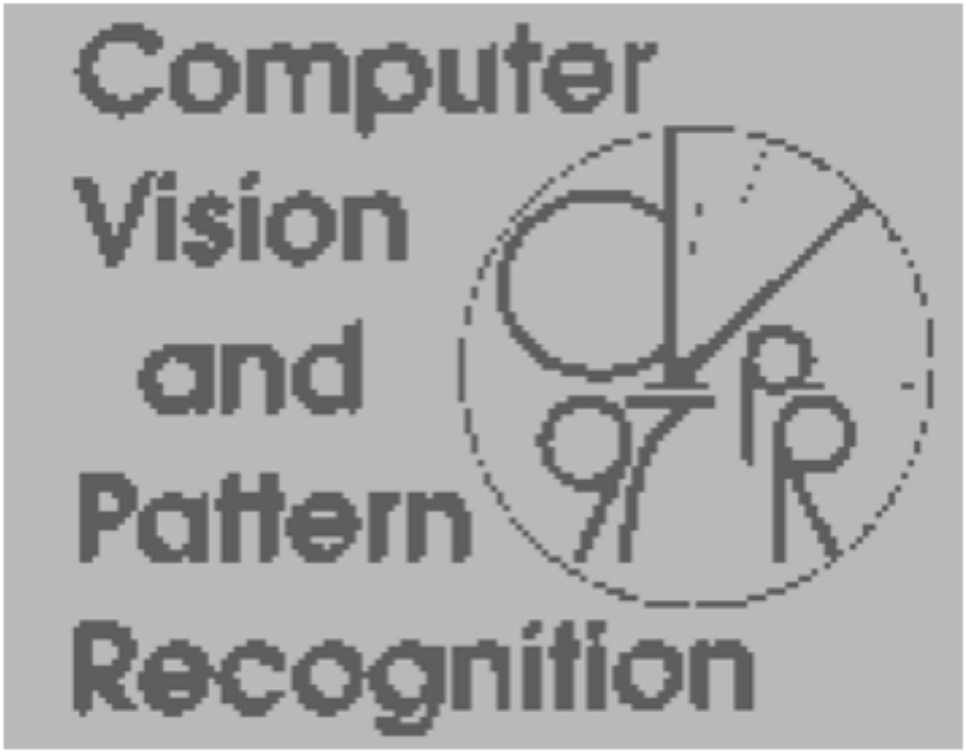


imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, 0.2f, true);

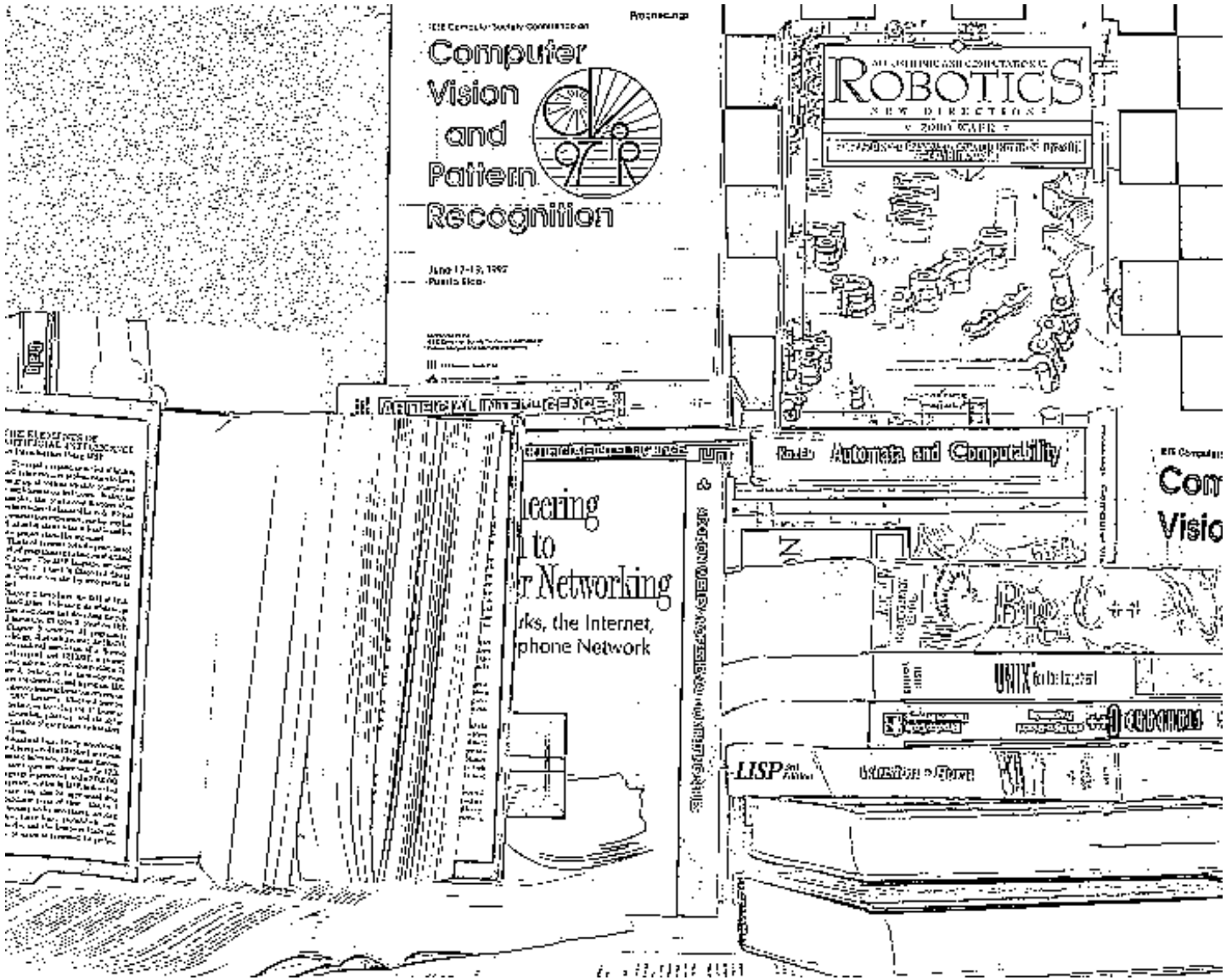


int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
zoom in

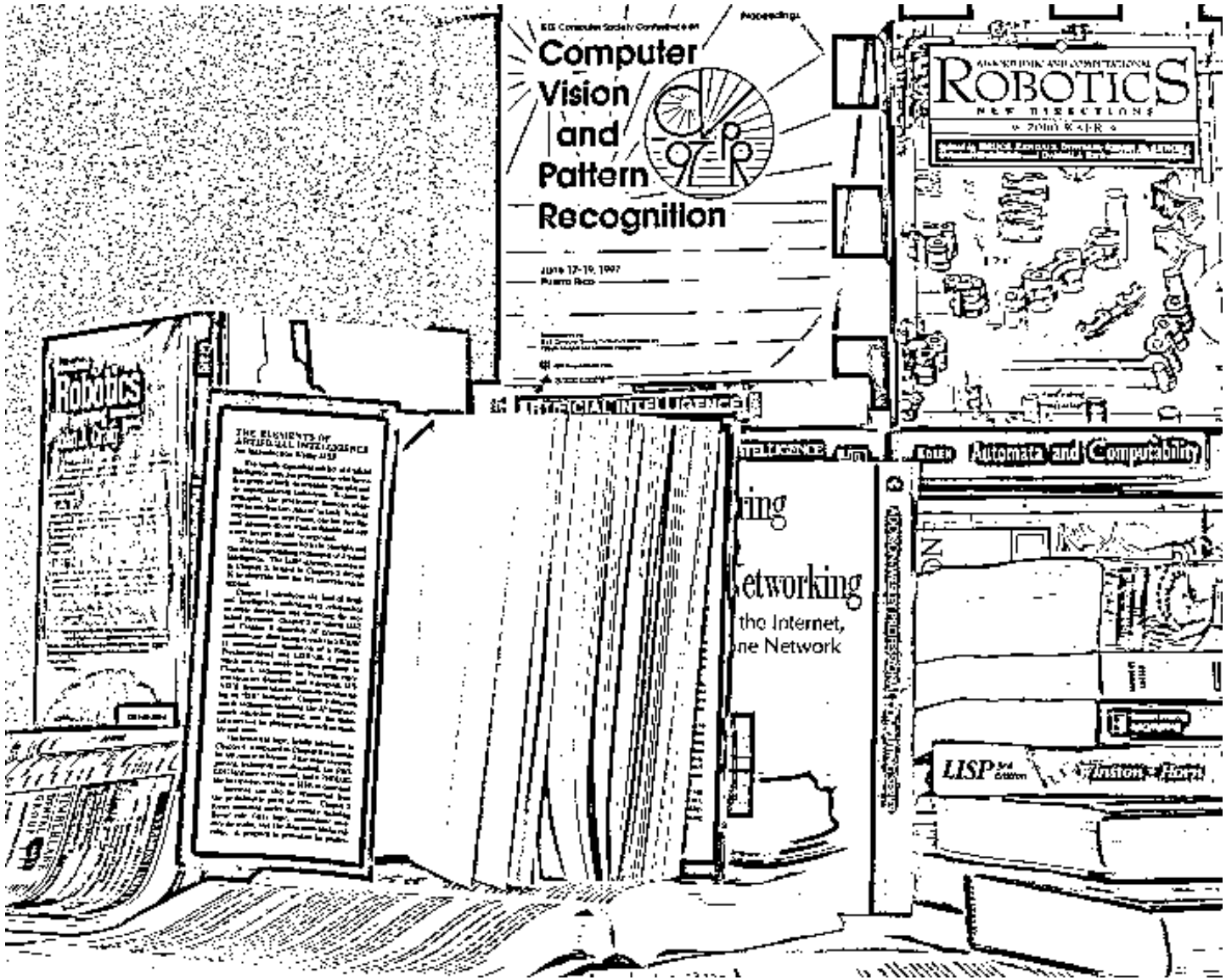




adaptive means, h=1

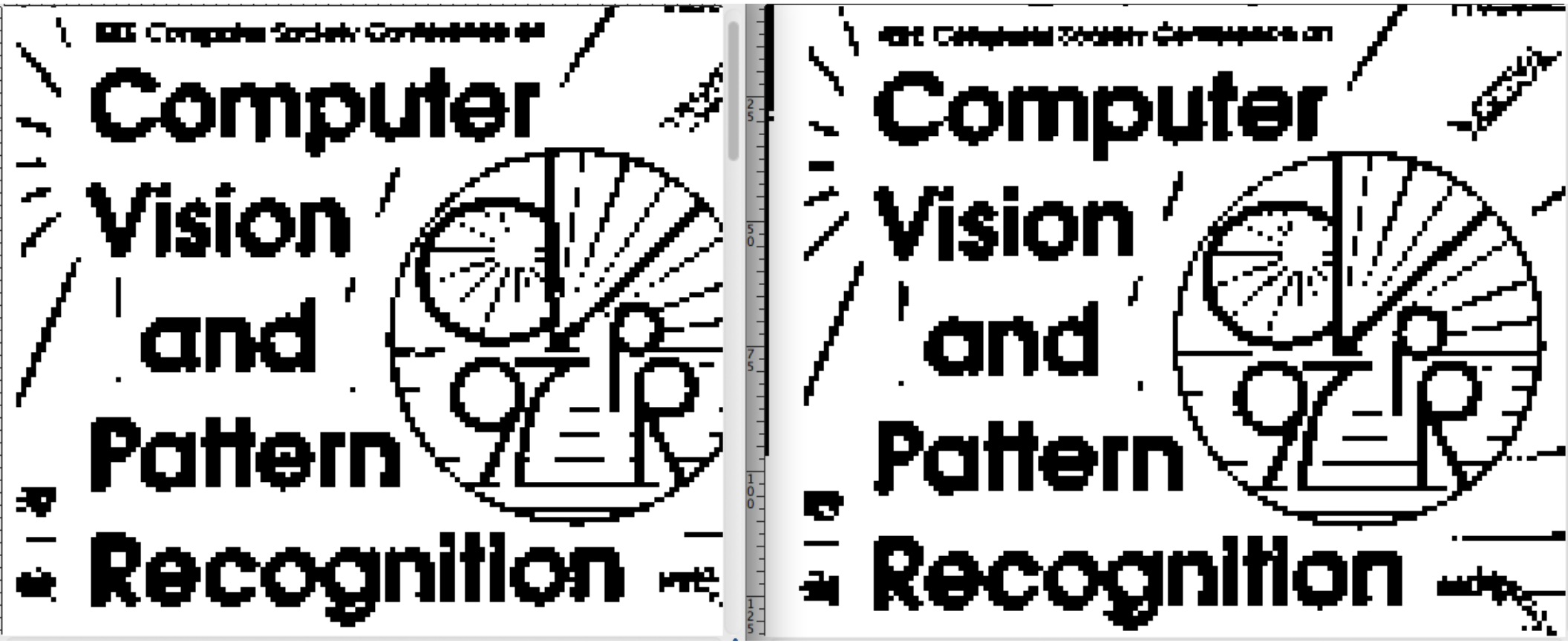


adaptive means, h=3

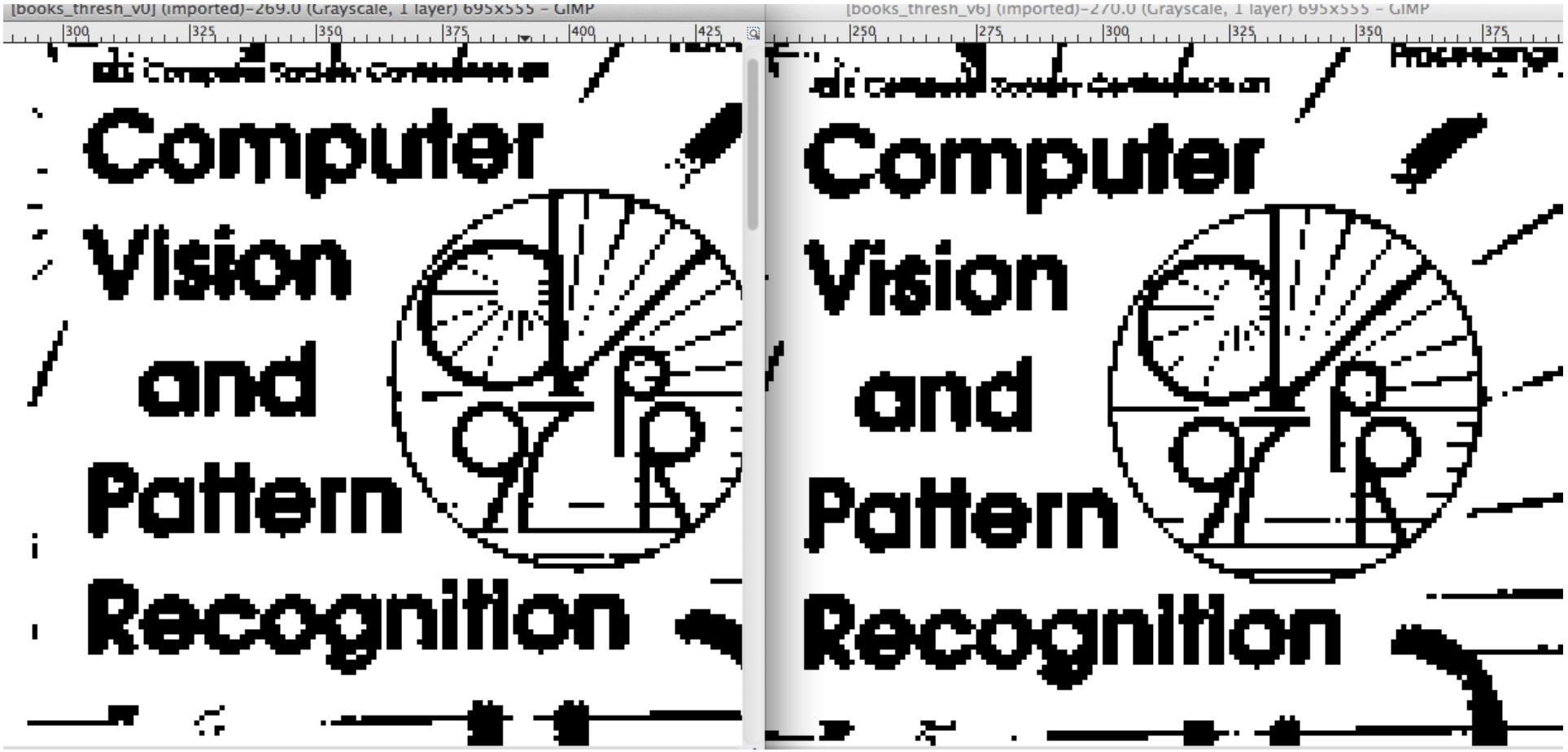


adaptive means, h=5

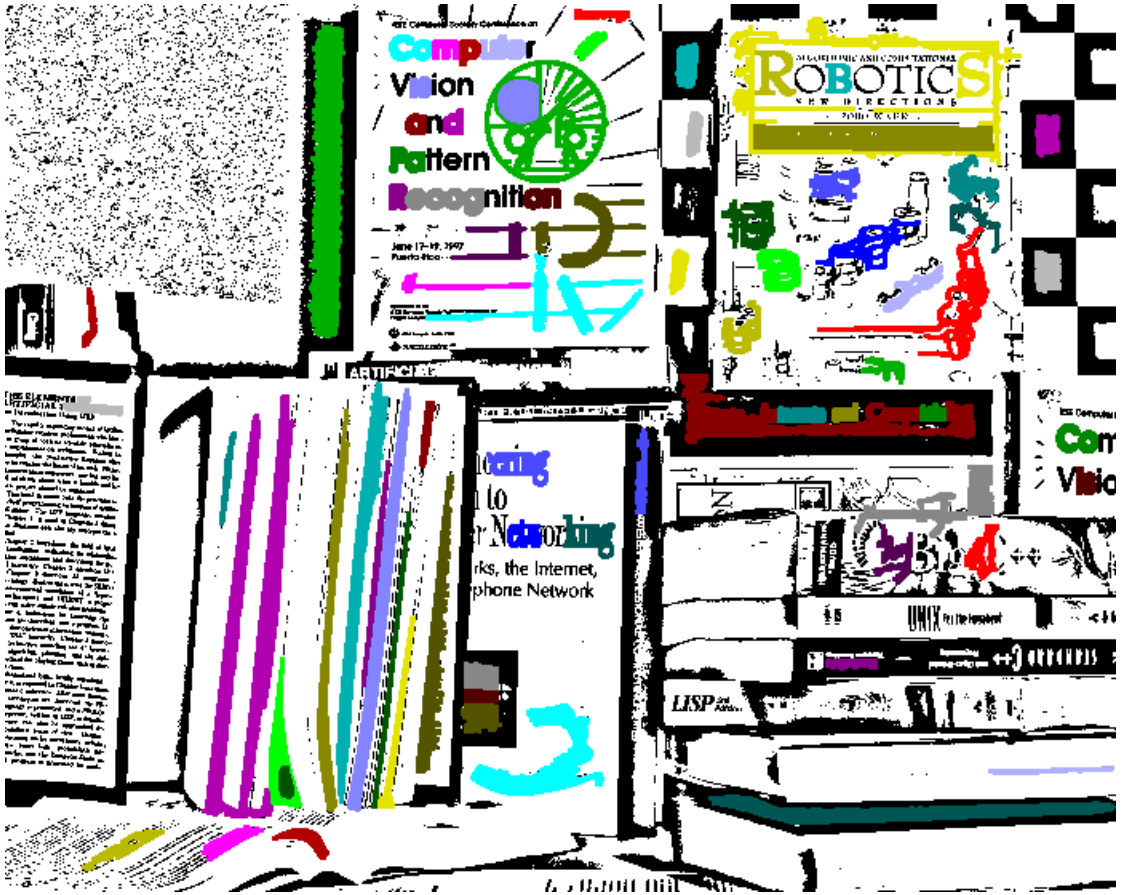
adaptive means, h=7



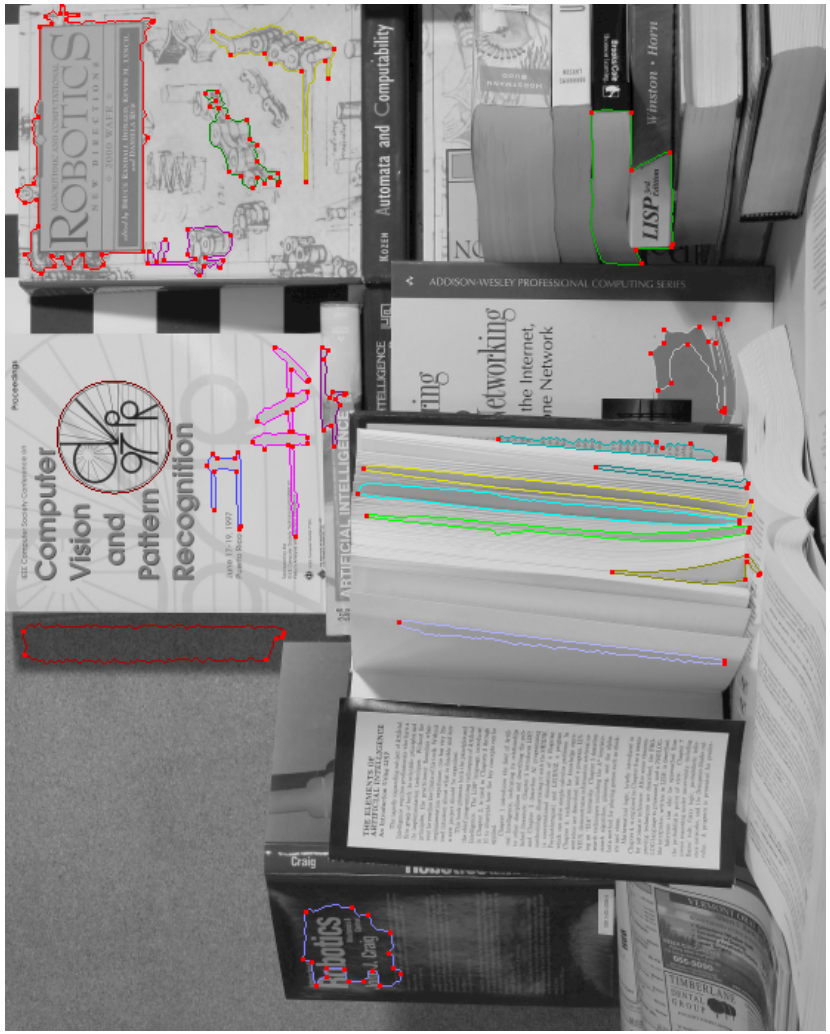
adaptive means, h=11 for adaptive to join close letters

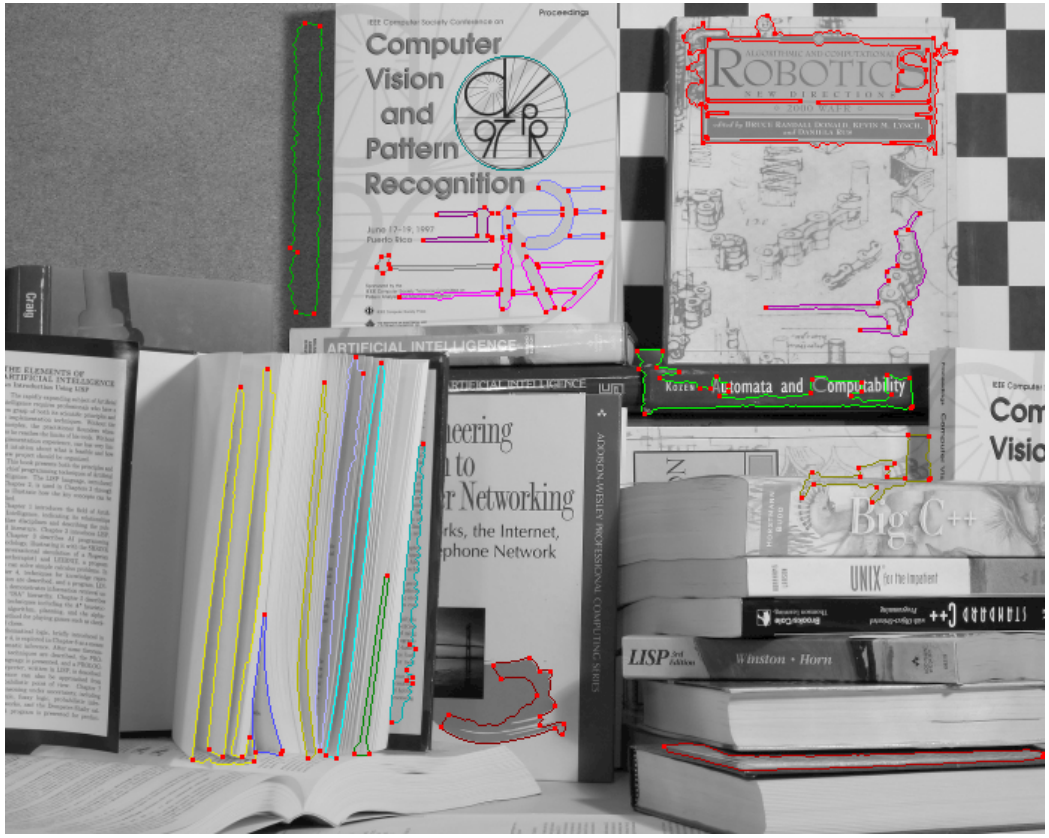


adaptive means, h=11 blobs (extr using scale calc code)



adaptive means, h=11 blob corners

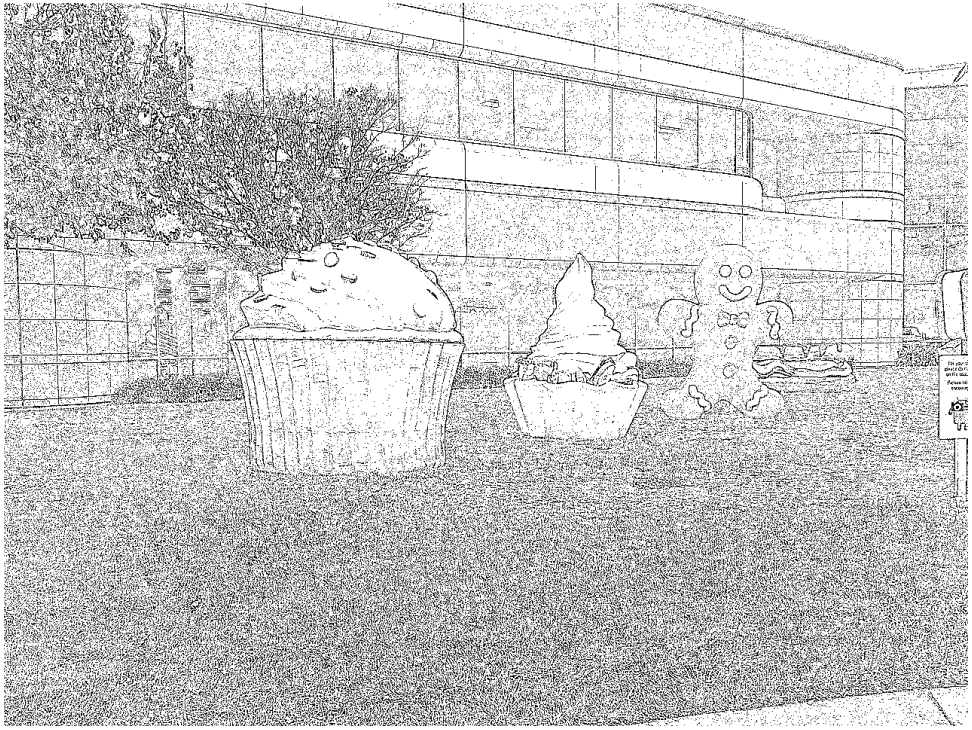




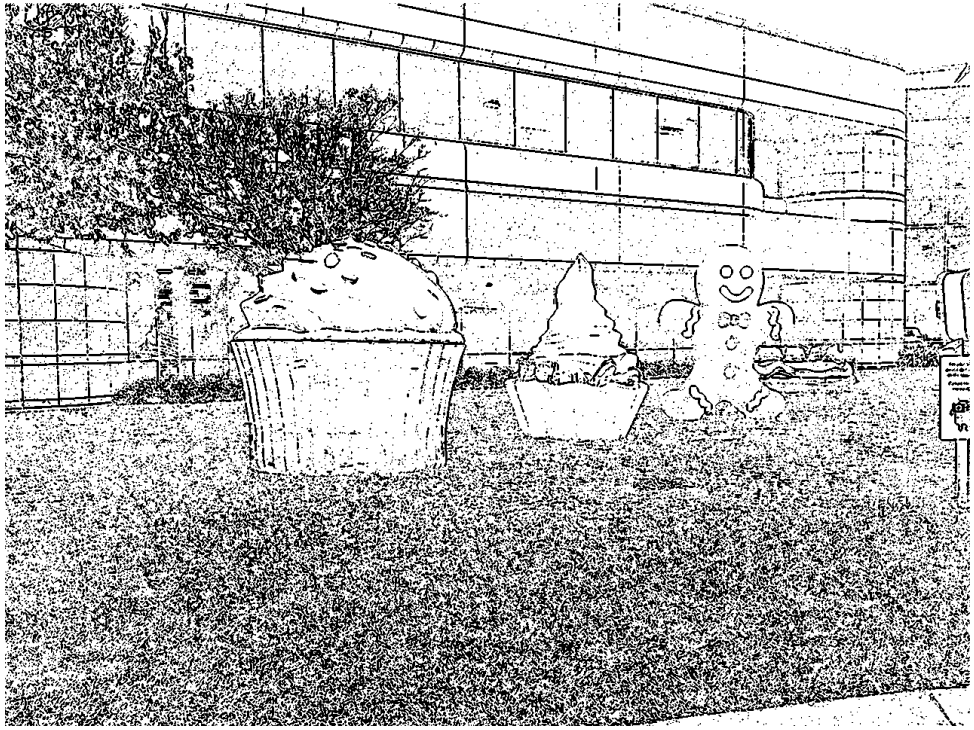
the letters are filtered out due to large nPerimeter compared to nInternal

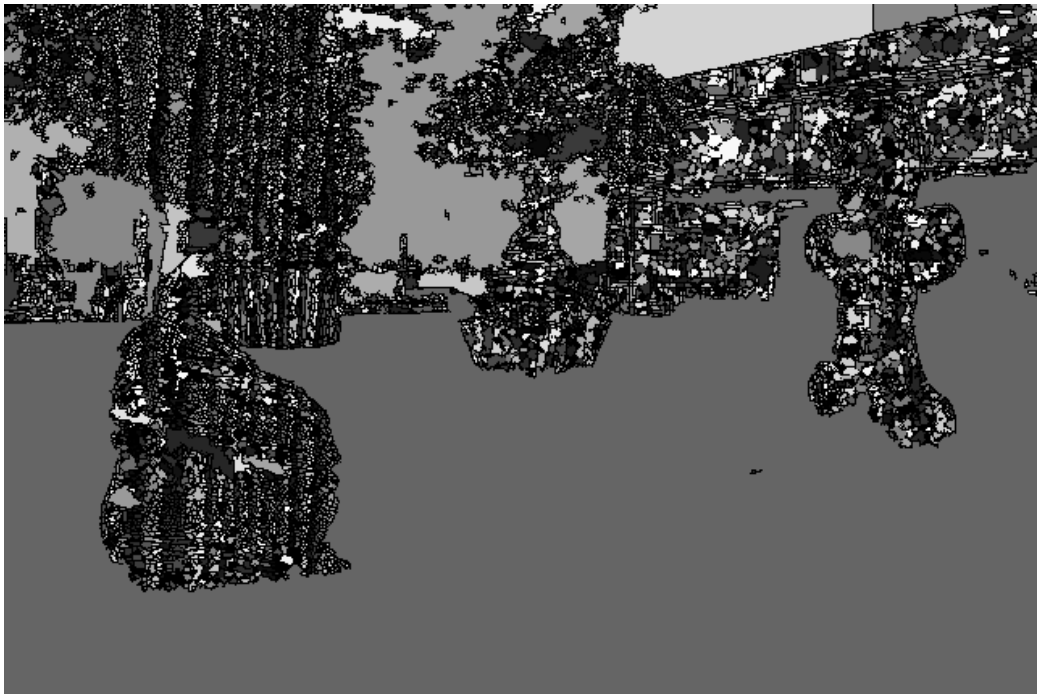
adaptive means, h=3 was better for letters as blobs

adaptive means, h=1



two median smooth with h=2



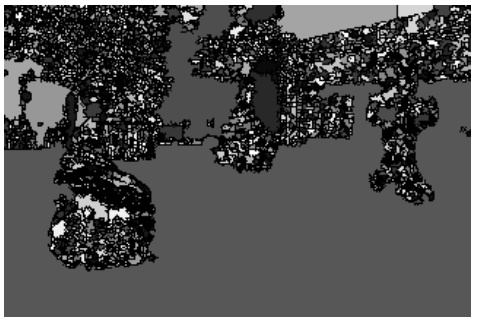


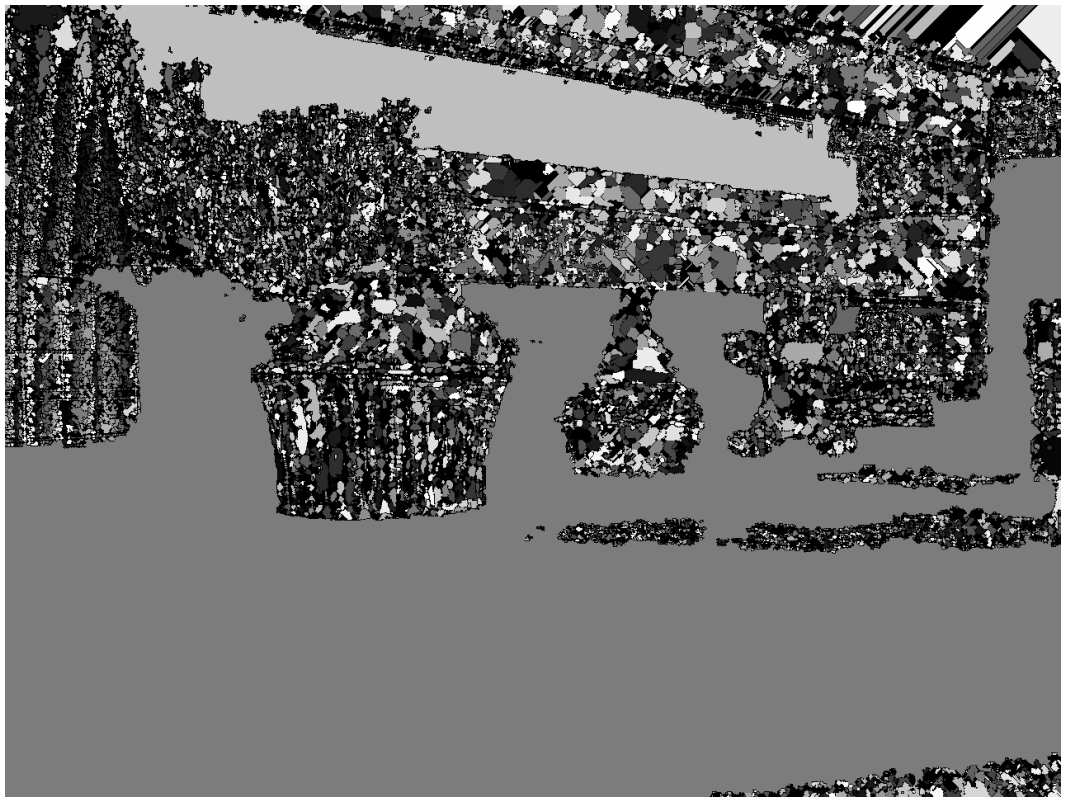
A (from LAB color space) watershed:

very large levels looks like the start of a good mask (excepting for green objects on green background)

<—- full size image

image binned to max integer dimension 350



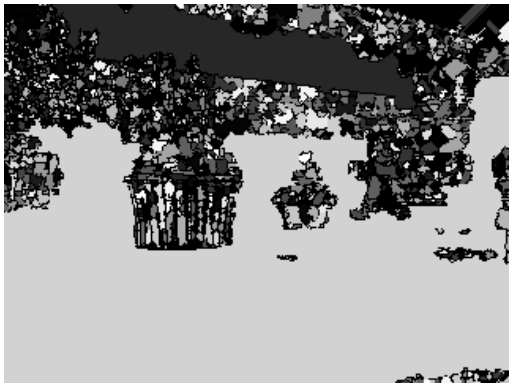


A (from LAB color space) watershed:

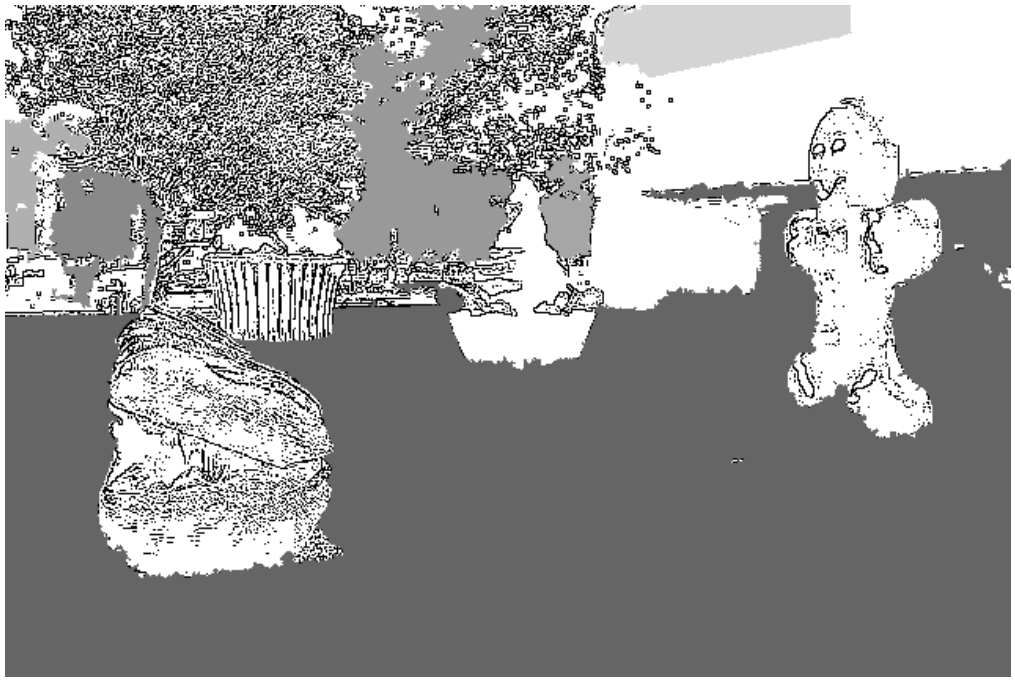
very large levels looks like the start of a good mask (excepting for green objects)

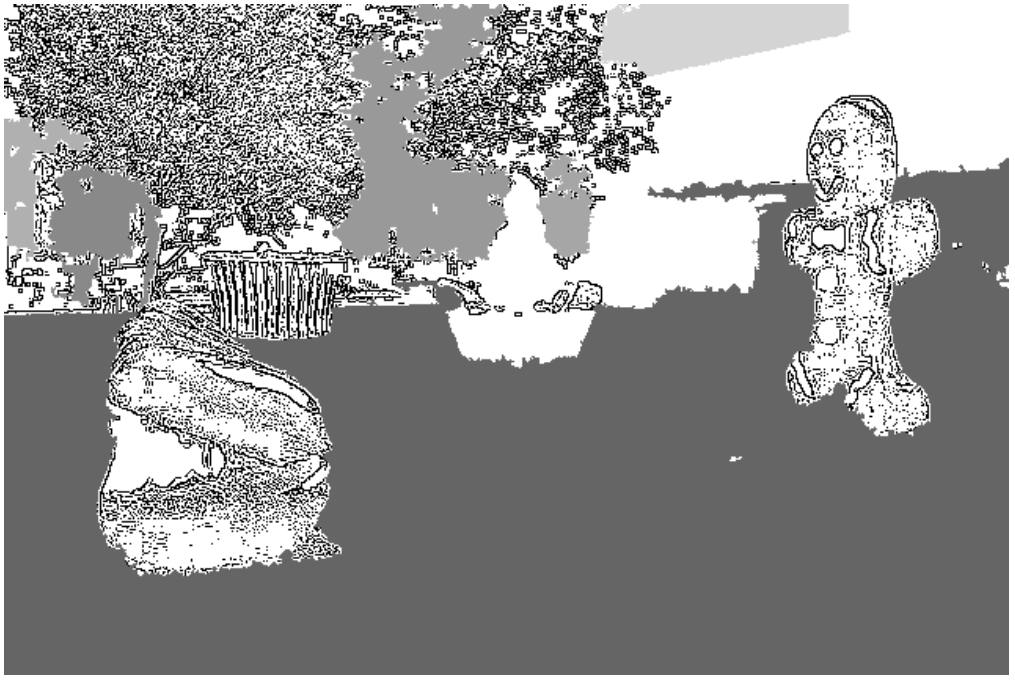
<—- full size image

image binned to max integer dimension 350



A watershed large regions as a mask
then O3 adaptive median or O2 adaptive median.



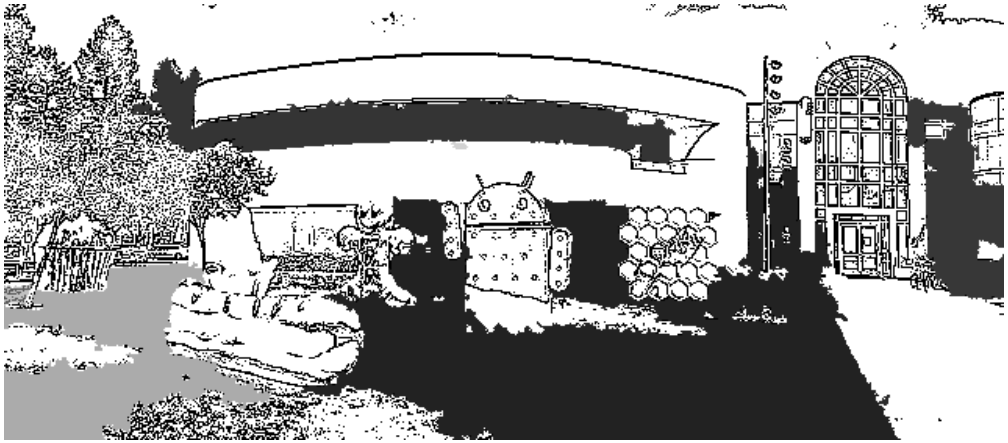


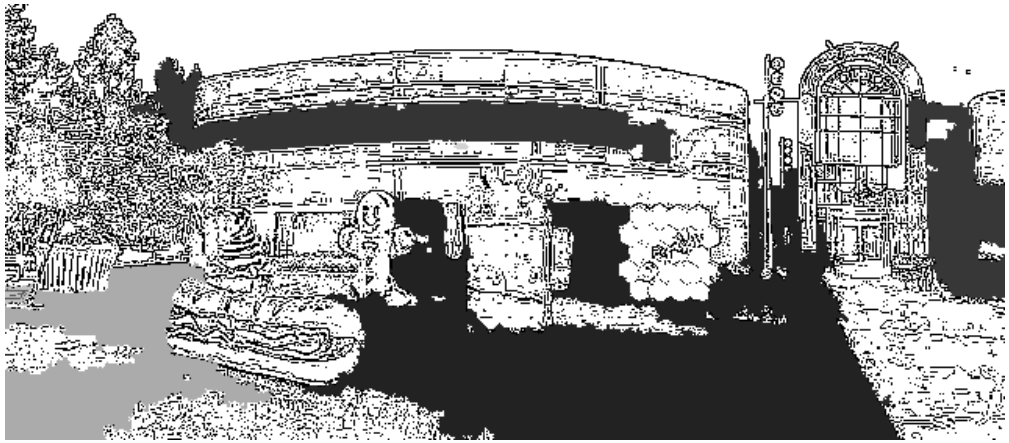
A watershed large regions as a mask
then O1 adaptive median.

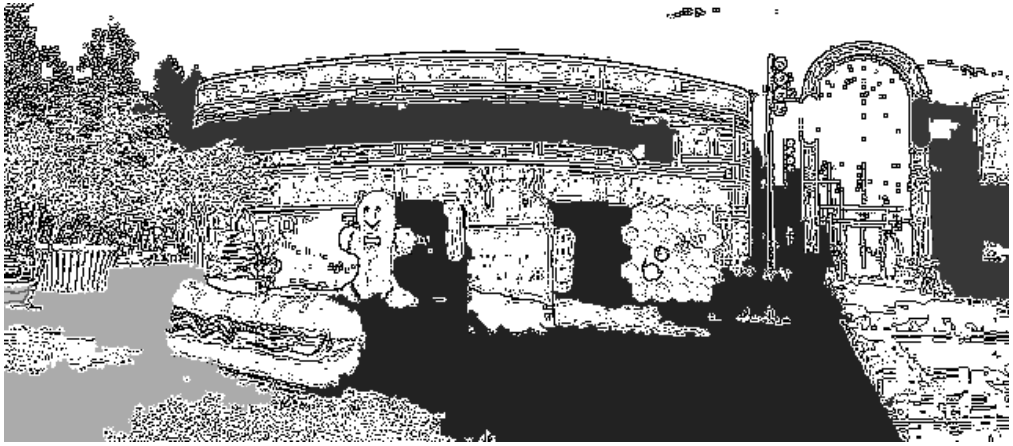
A watershed large regions as a mask
then O3 adaptive median or O2 adaptive median.

A watershed large regions as a mask
then O1 adaptive median.

A watershed large regions as a mask
then O3 adaptive median or O2 adaptive median.

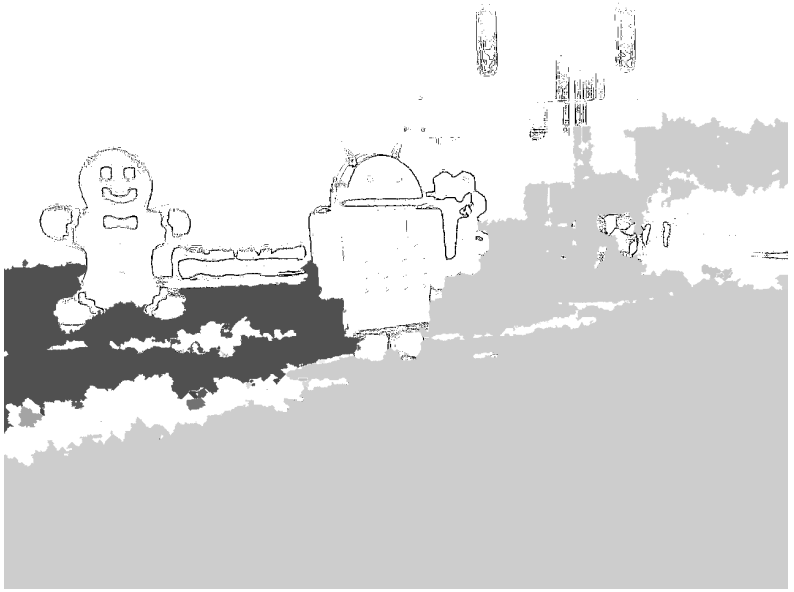




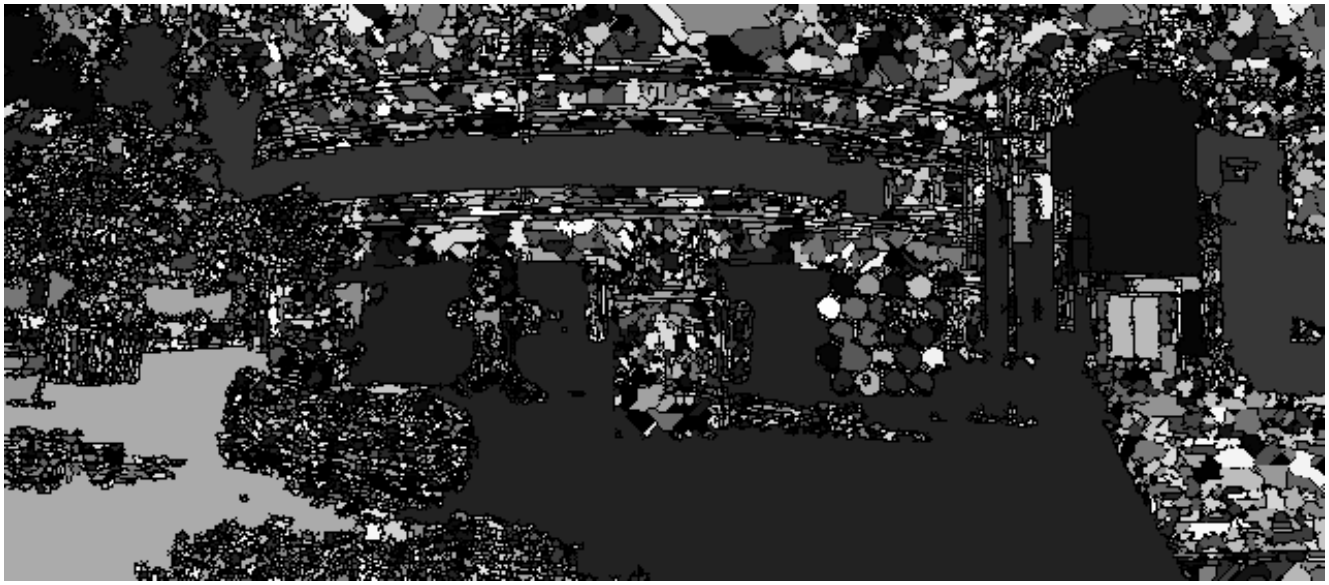


A watershed large regions as a mask
then O1 adaptive median.

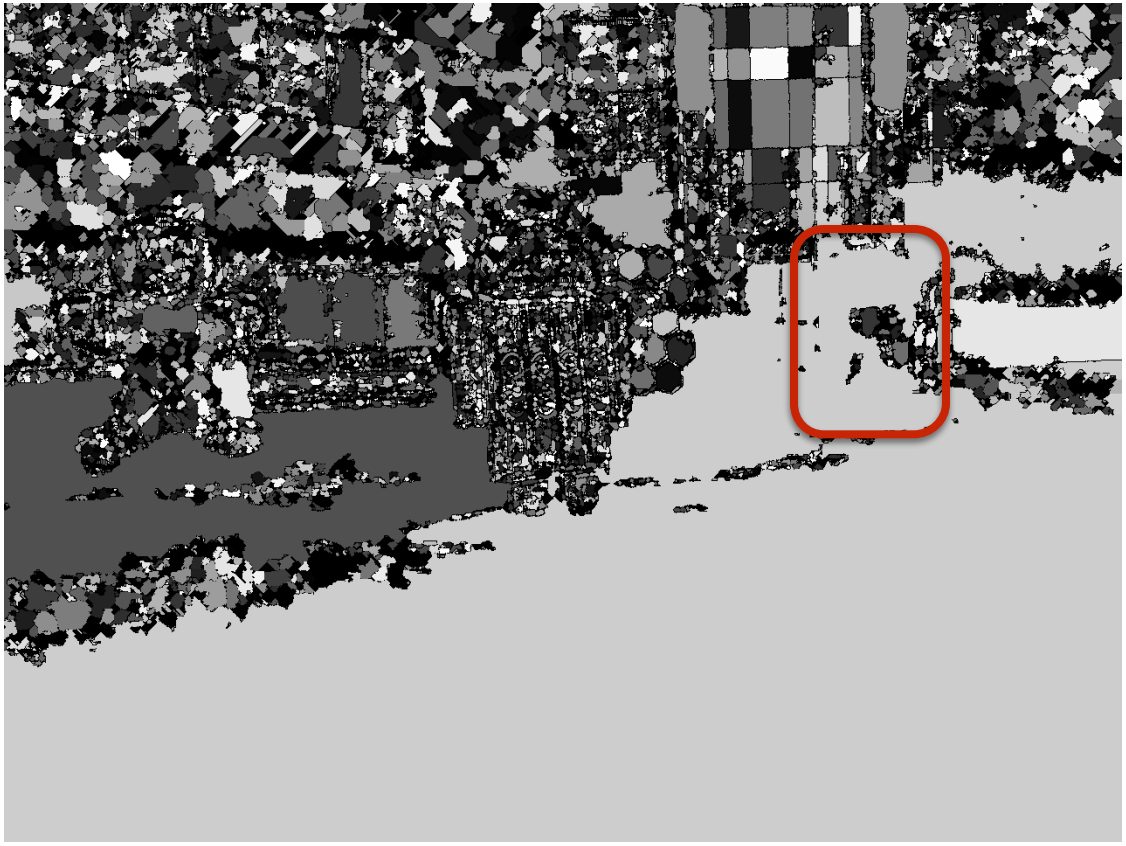
A watershed large regions as a mask
then O3 adaptive median or O2 adaptive median.



A watershed large regions as a mask
then O1 adaptive median.

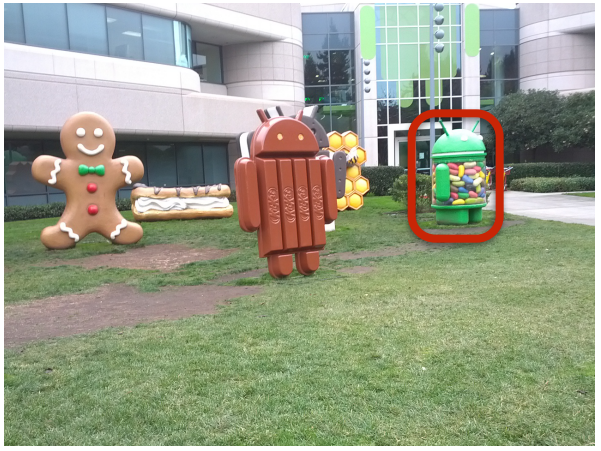


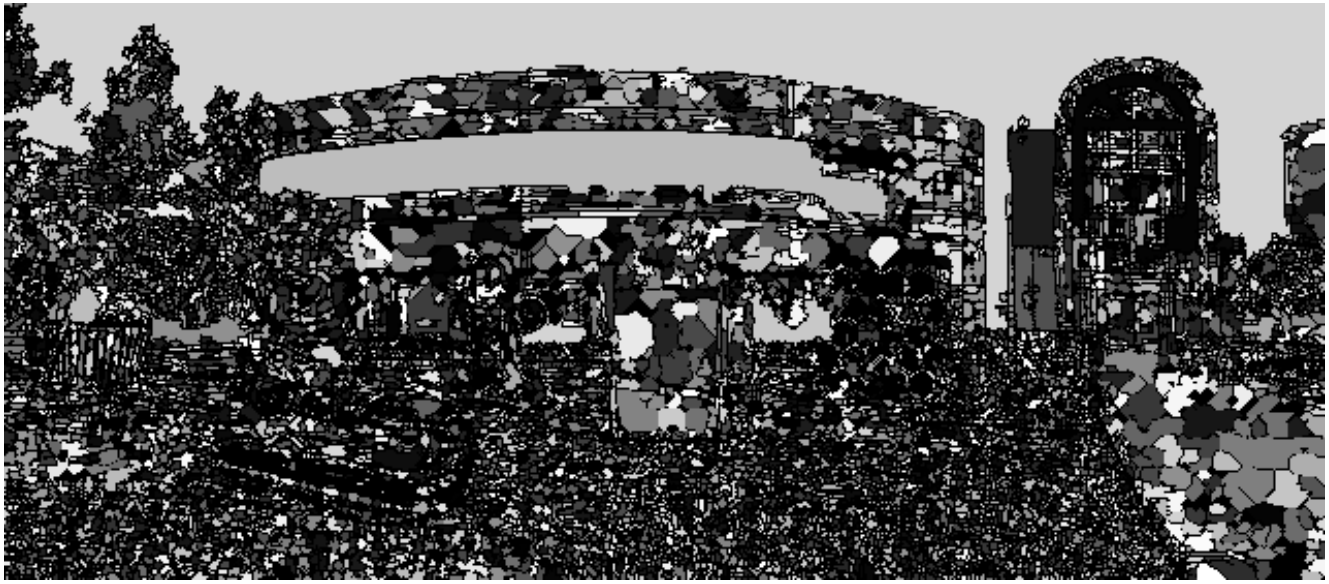
A (from LAB color space) watershed:



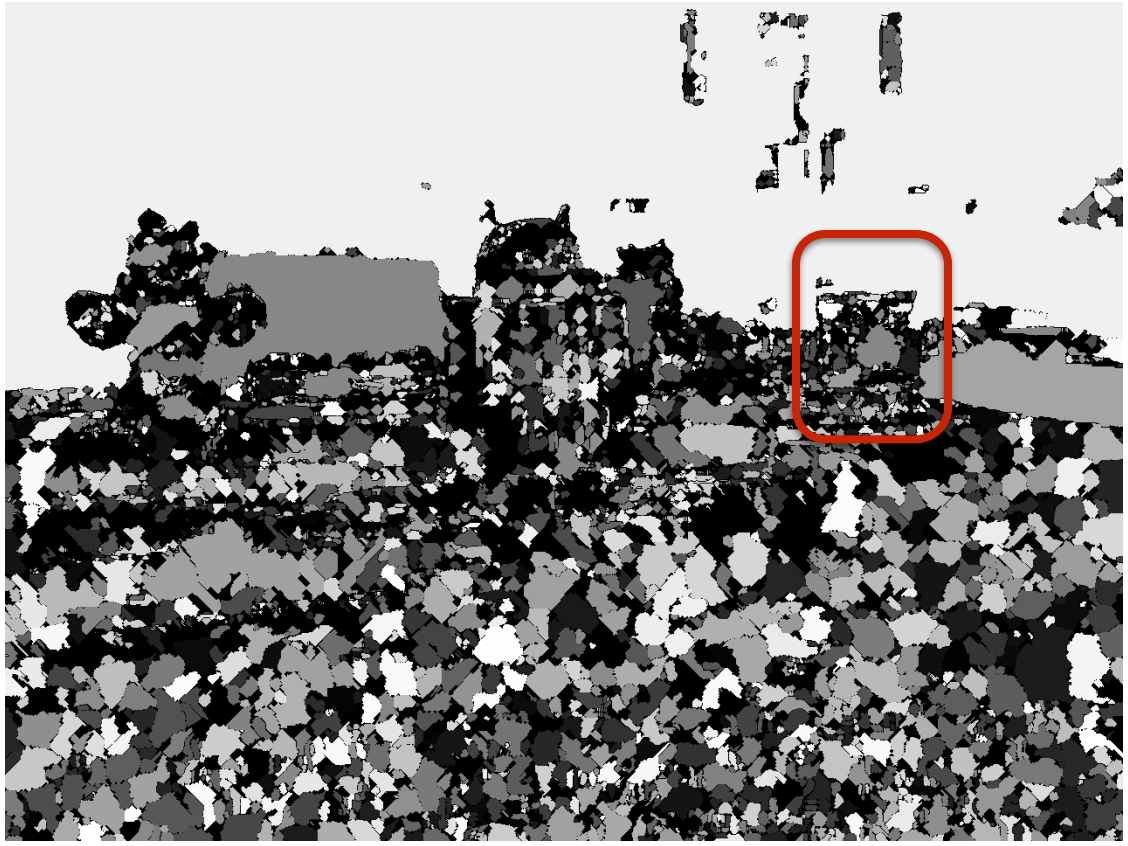
A (from LAB color space) watershed:

part of green droid is lost in grass watershed



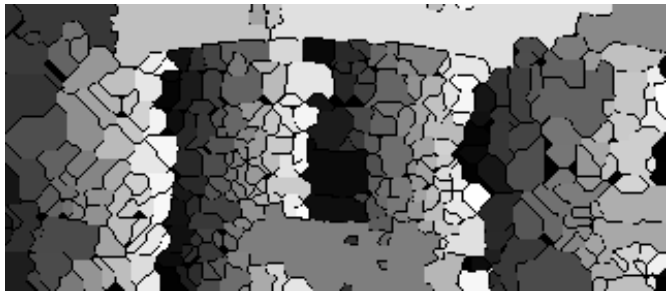


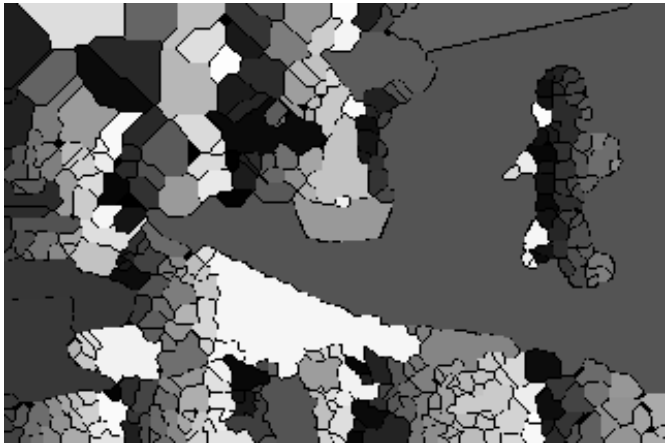
O2 watershed:

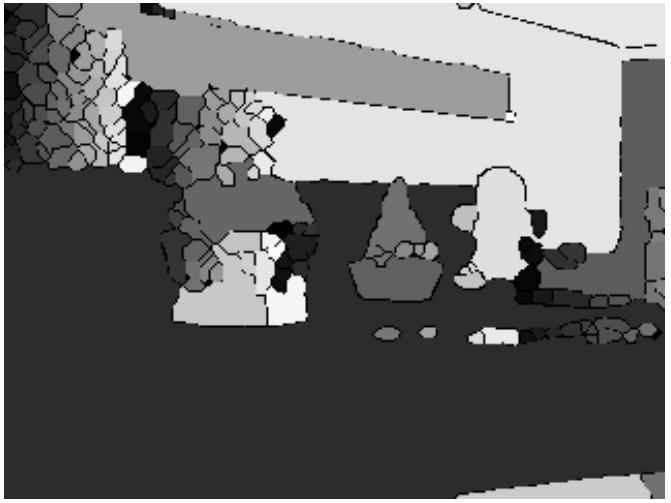


O2 watershed:

have created an image to use watershed algorithm on as the start of a color segmentation algorithm.
A greyscale image is created and a small sigma first derivative gaussian is applied to the image.
Thresholding for a fraction of the number of counts is performed to remove the lowest intensity pixels,
then the image is inverted and combined with an image that is the O1 adaptive median image.
A watershed algorithm is performed on that combined image to give the following results.
The results are very granular, so an algorithm to merge similar adjacent cells has been made too.



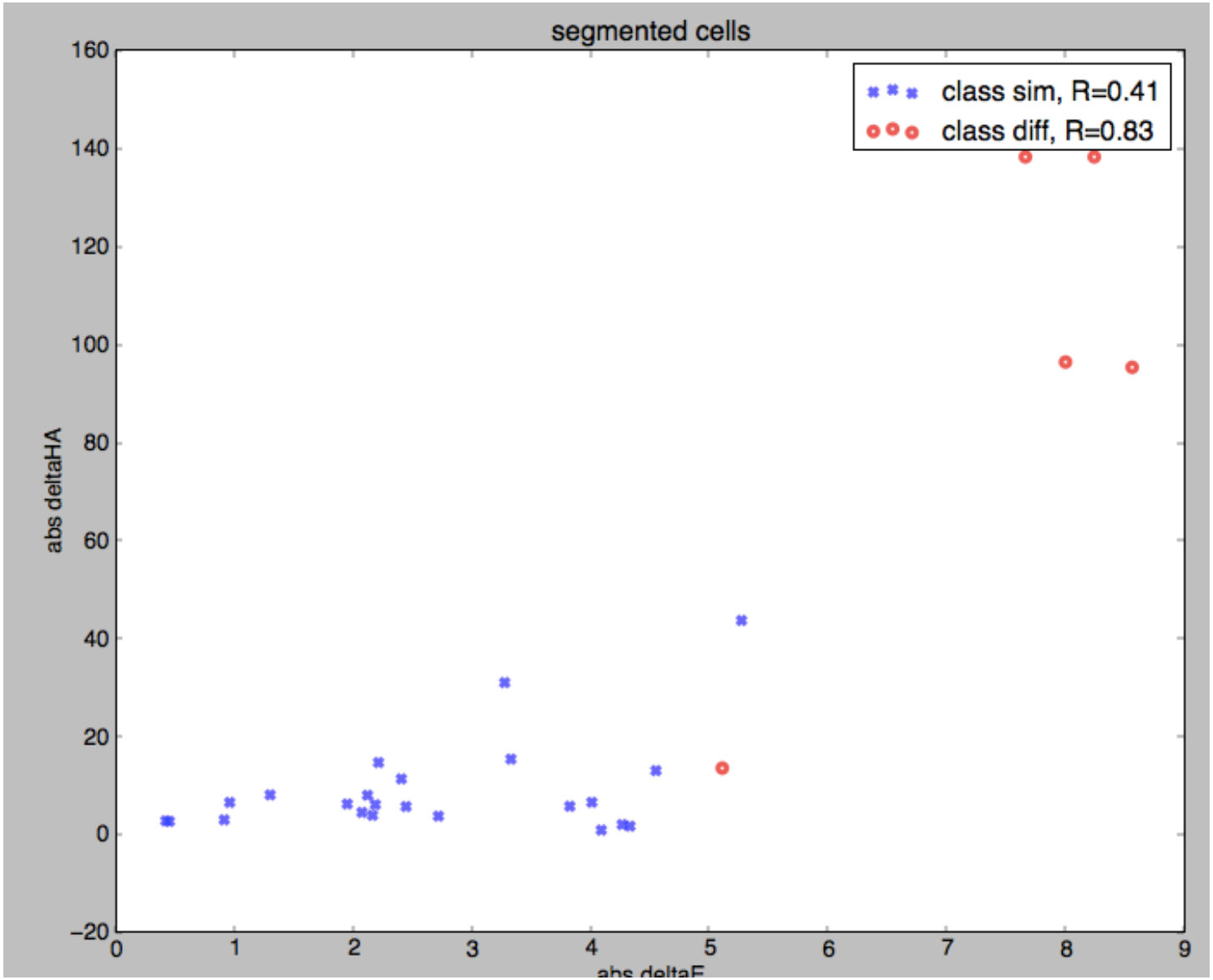


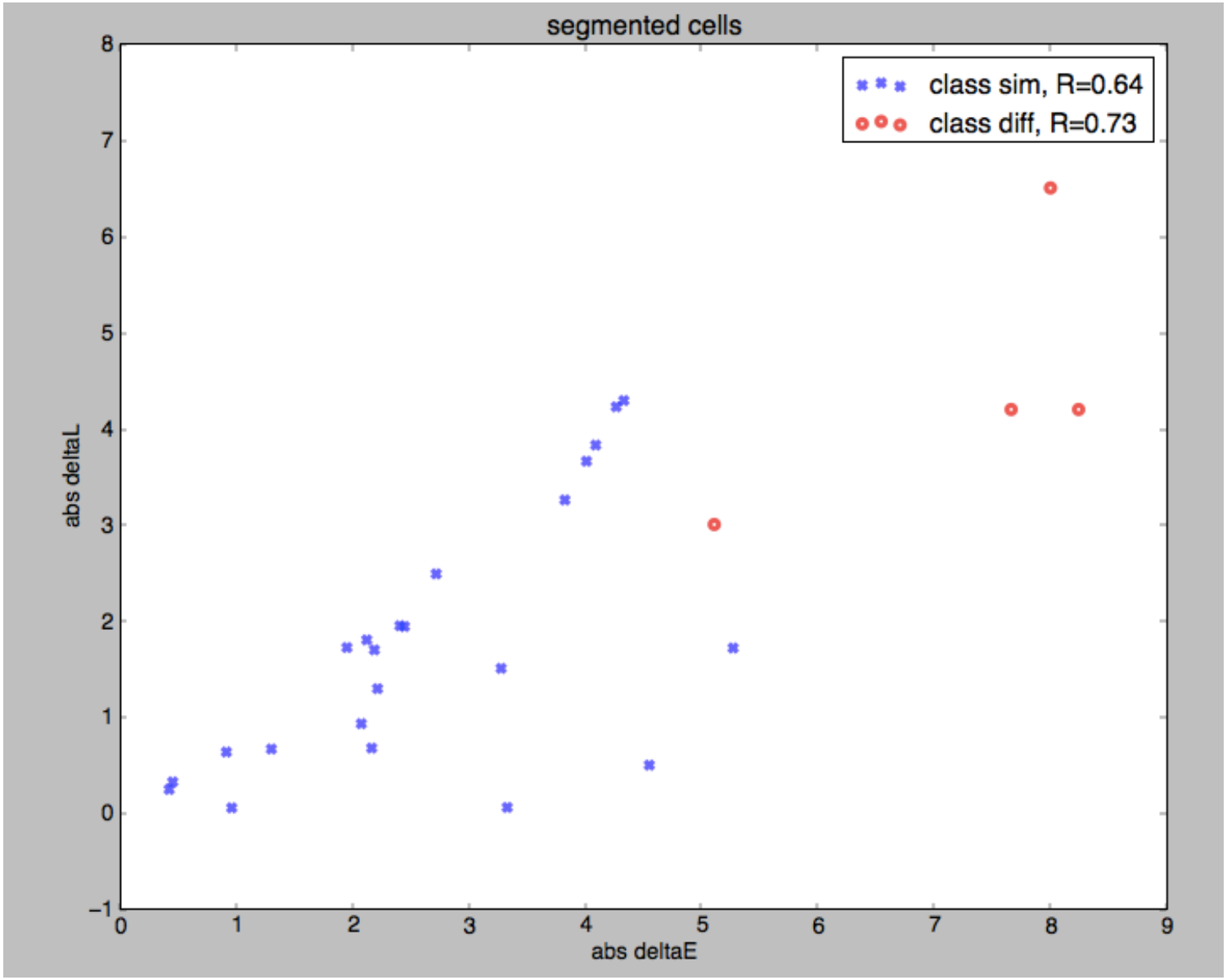


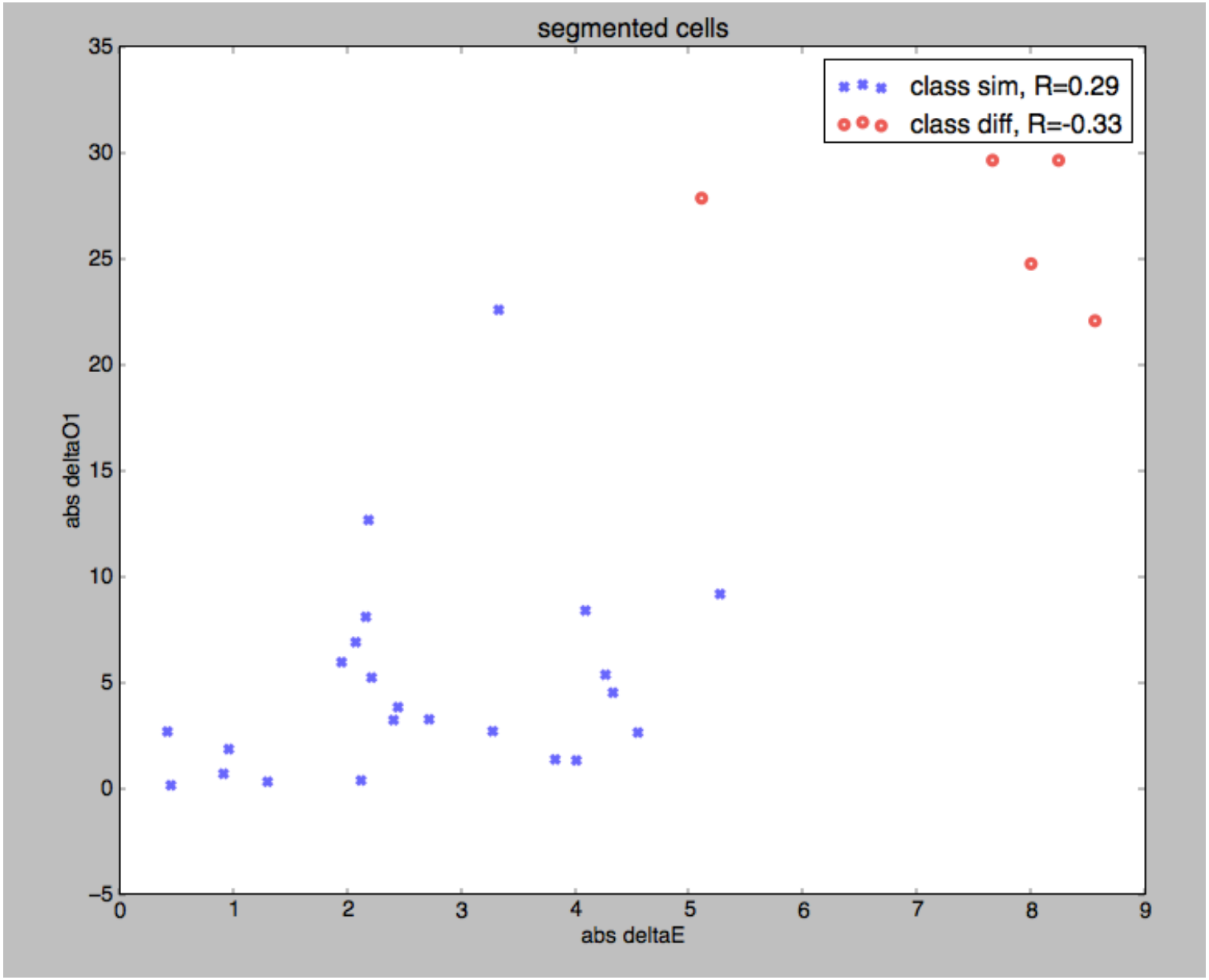
A file of similar and different pairs of adjacent cells and their colors was made as 2 ‘classes’ for LDA of the colors to find the best combination of colors for separating the classes.

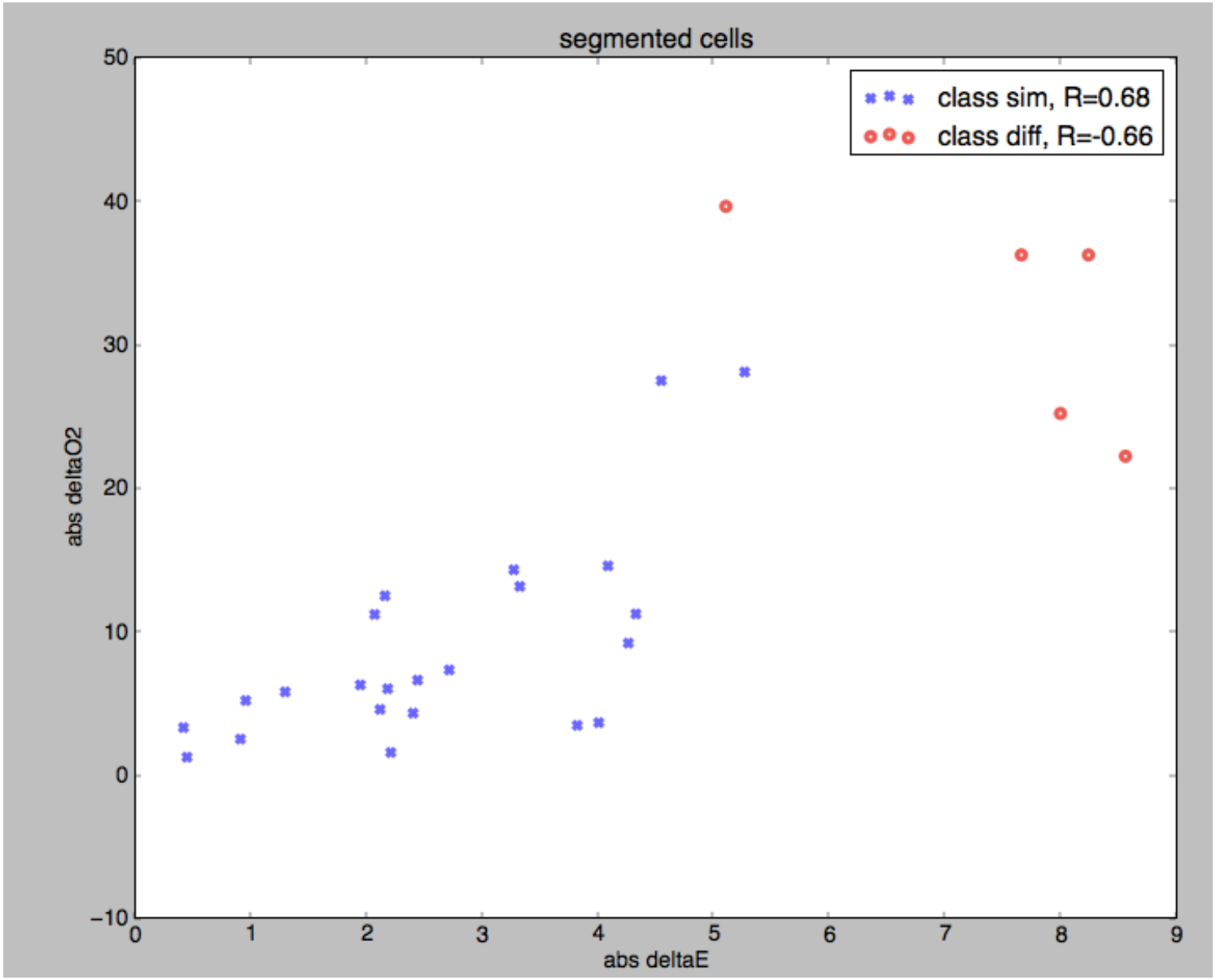
The best results were to use deltaE 1994 and delta opponent colors.

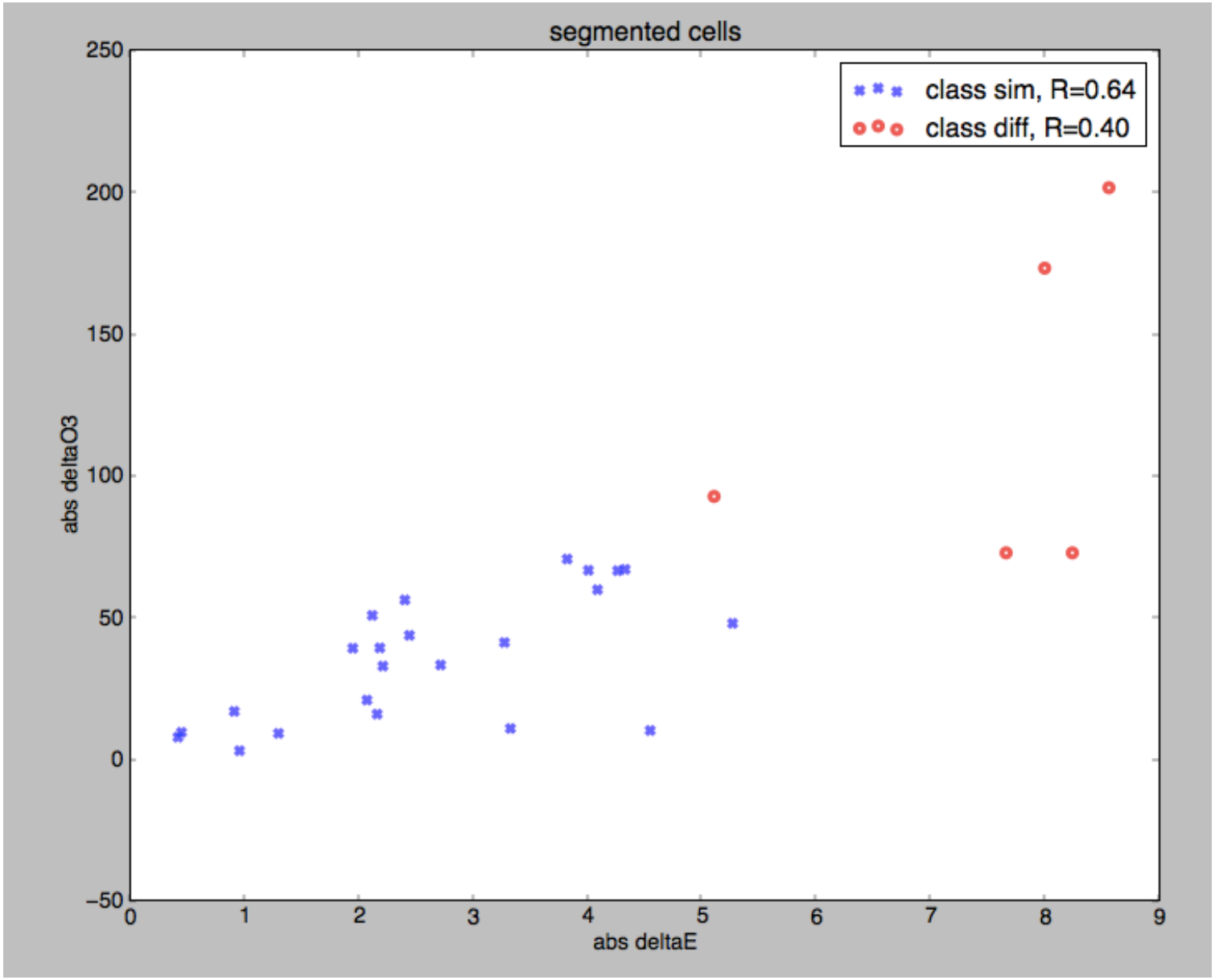
The next pages are diagrams of the results.

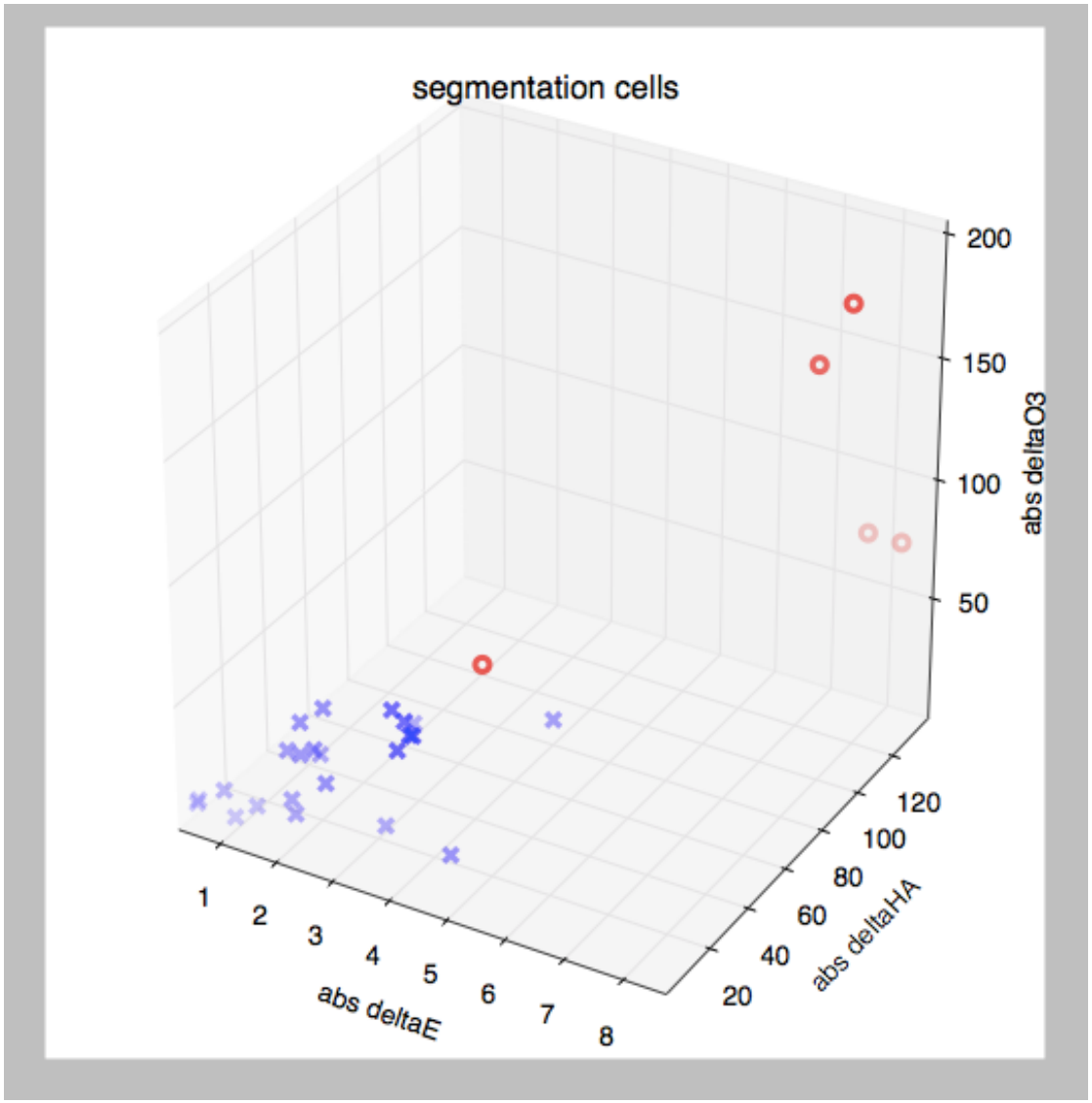

segmented cells
class sim, R=0.41
class diff, R=0.83
abs deltaHA
abs deltaE
160
140
120
100
80
60
40
20
0
−20
0
1
2
3
4
5
6
7
8
9

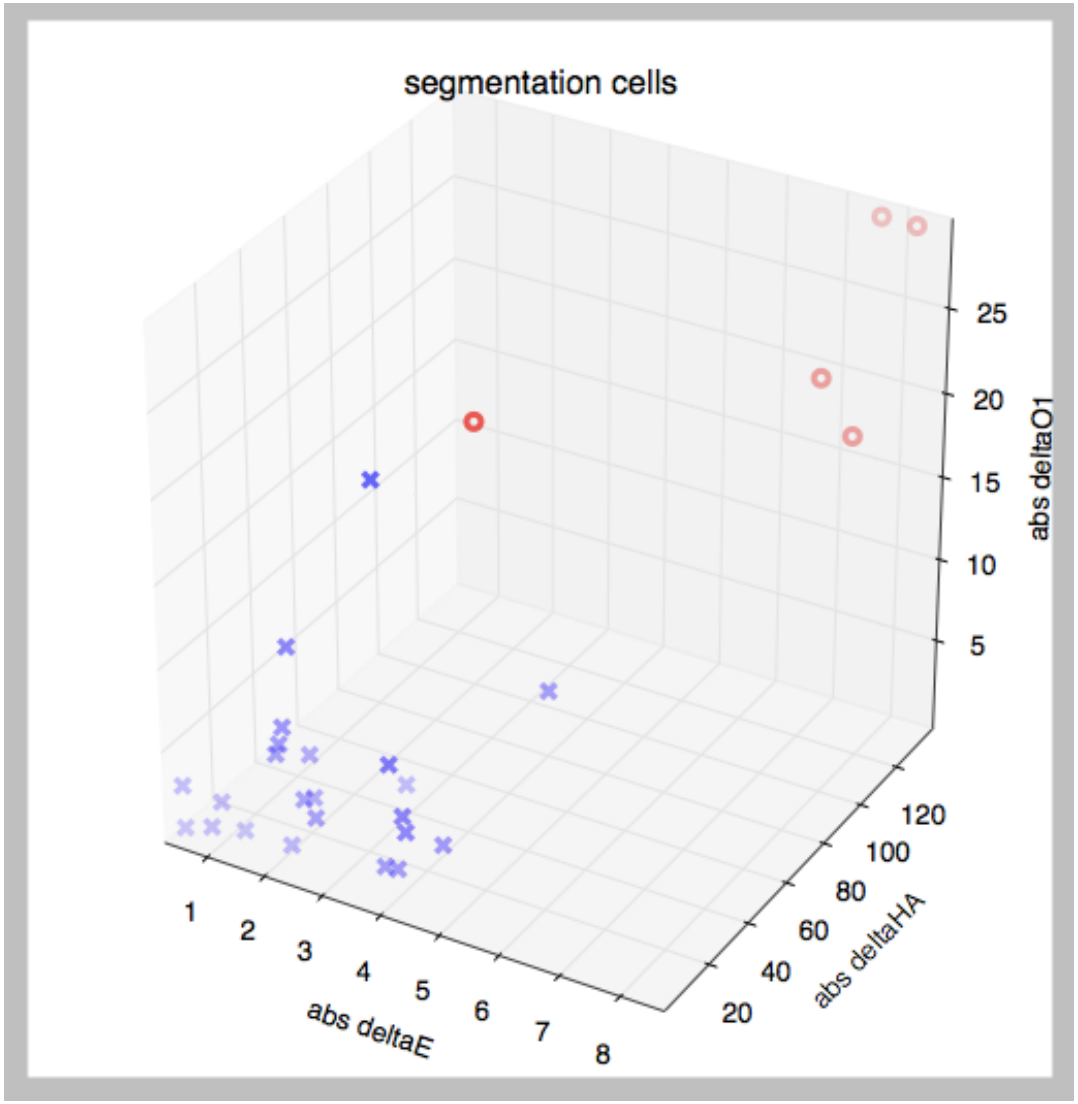

segmented cells
class sim, R=0.64
class diff, R=0.73
abs deltaL
abs deltaE
8
7
6
5
4
3
2
1
0
−1
0
1
2
3
4
5
6
7
8
9


segmented cells
class sim, R=0.29
class diff, R=-0.33
abs deltaO1
abs deltaE
35
30
25
20
15
10
5
0
−5
0
1
2
3
4
5
6
7
8
9

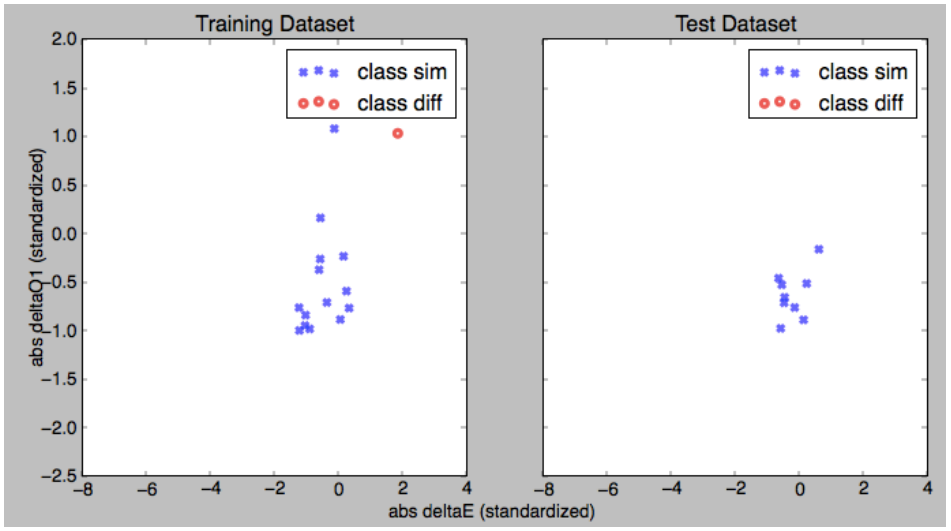

segmented cells
class sim, R=0.68
class diff, R=-0.66
abs deltaO2
abs deltaE
50
40
30
20
10
0
−10
0
1
2
3
4
5
6
7
8
9

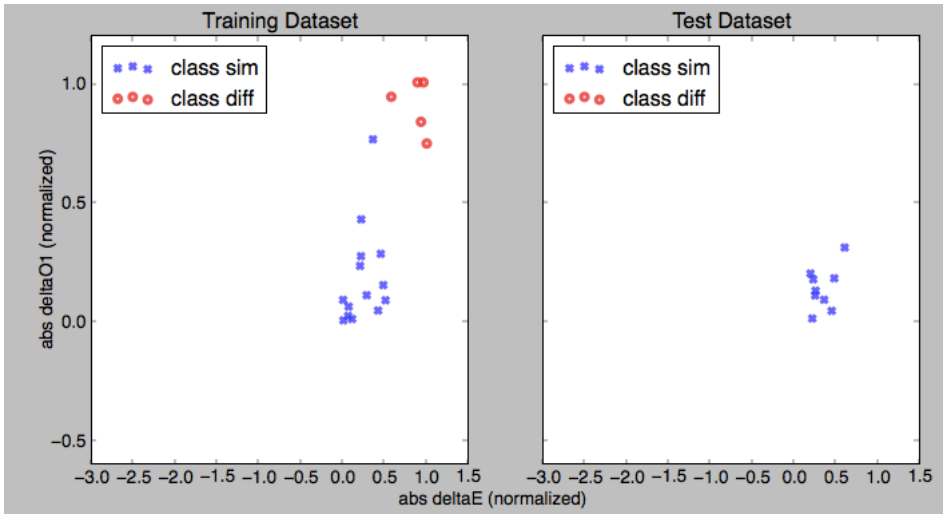

segmented cells
class sim, R=0.64
class diff, R=0.40
abs deltaO3
abs deltaE
250
200
150
100
50
0
−50
0
1
2
3
4
5
6
7
8
9


segmentation cells
200
150
100
50
abs deltaO3
120
100
80
60
40
20
abs deltaHA
1
2
3
4
5
6
7
8
abs deltaE

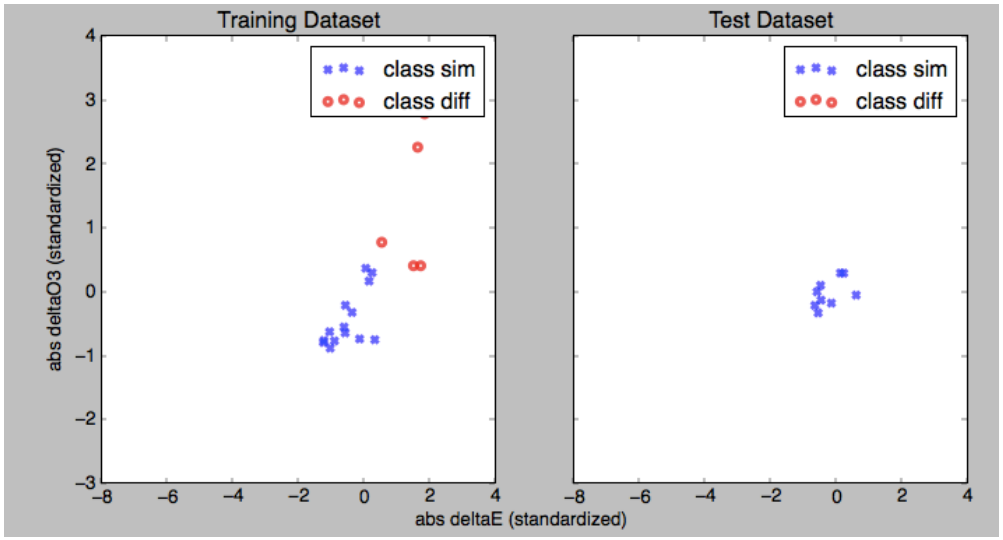

segmentation cells
abs deltaE
1
2
3
4
5
6
7
8
abs deltaHA
20
40
60
80
100
120
abs deltaO1
5
10
15
20
25

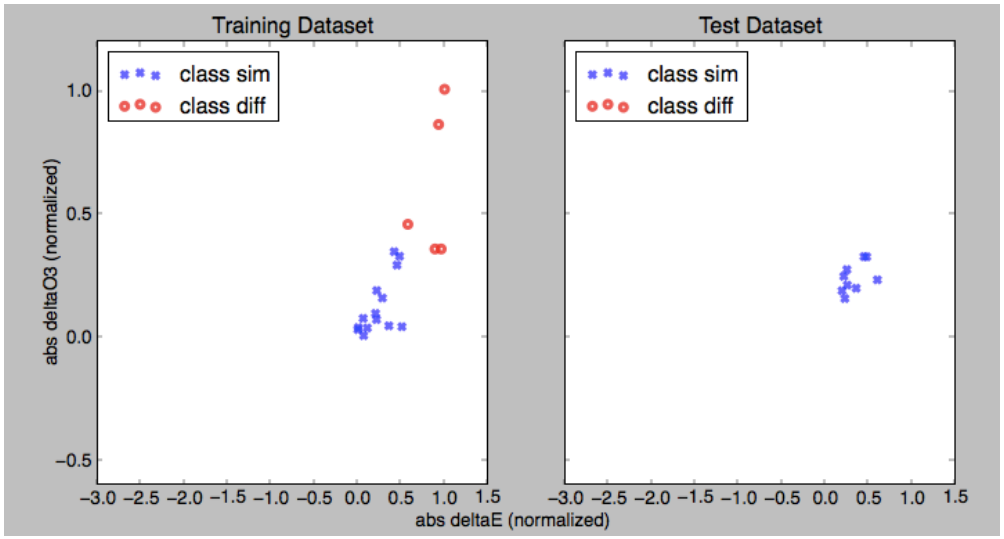
deltaE and delta O1



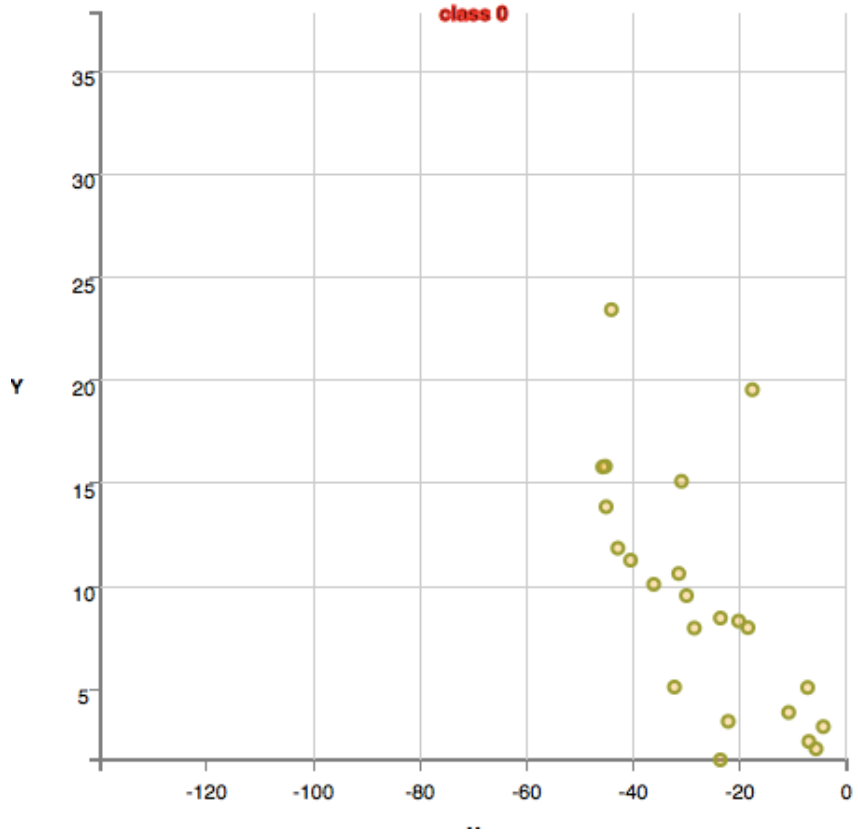


deltaE and delta O3

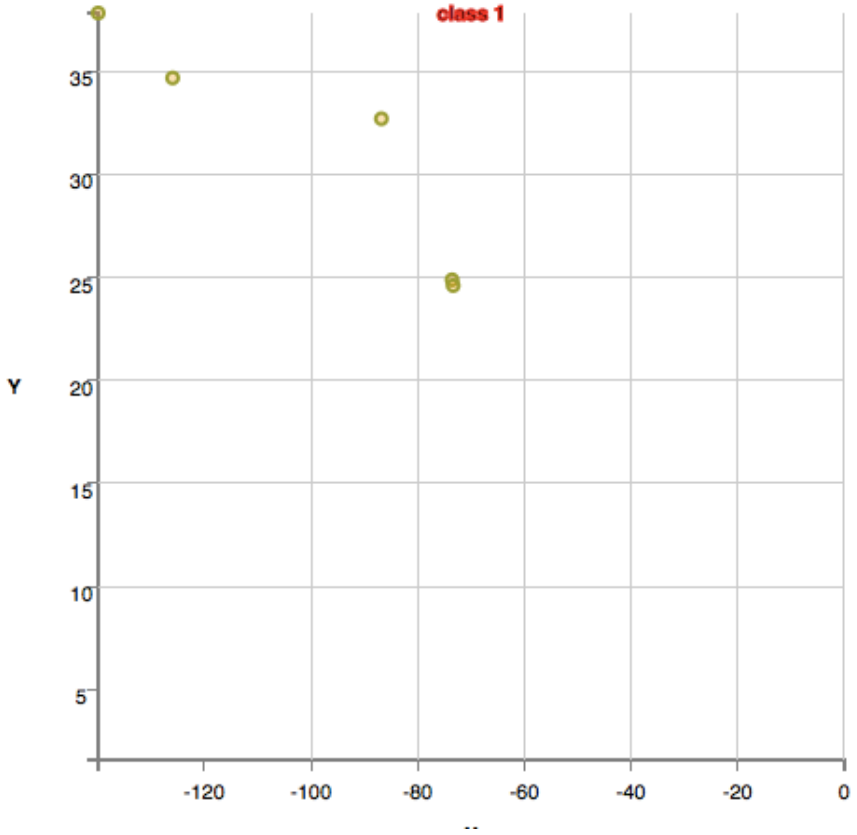




similar cells



different cells



result of LDA applied to feature classes using deltaE 1994, delta O1, delta O2, and delta O3.

(more points will be used when adjacency map is improved).

```
INFO: transfrmation matrix =
Type = dense , numRows = 2 , numCols = 4
 0.343  -0.575  -0.439  -0.599
-0.482  -0.381   0.771   0.166
```