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.

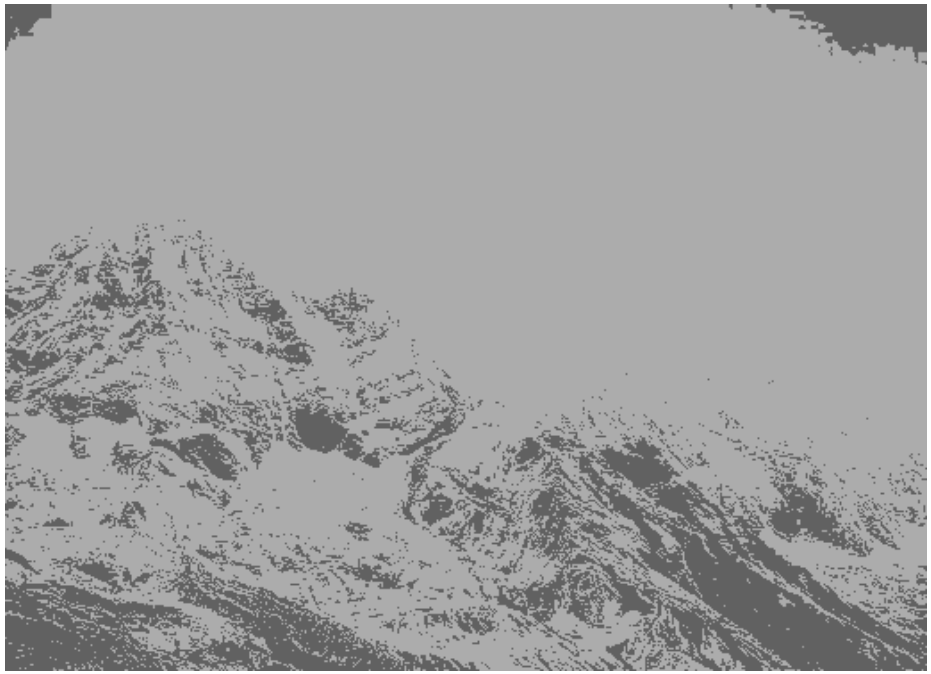
summary of next pages:

- 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.

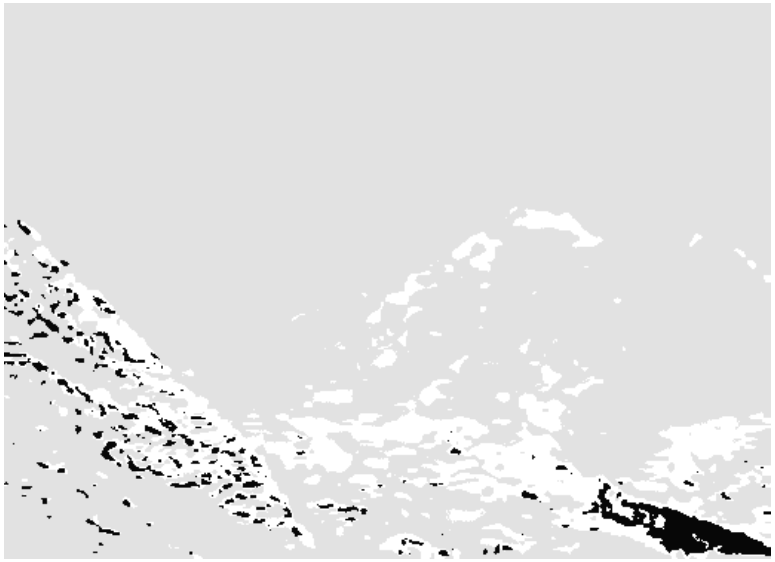
```
int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

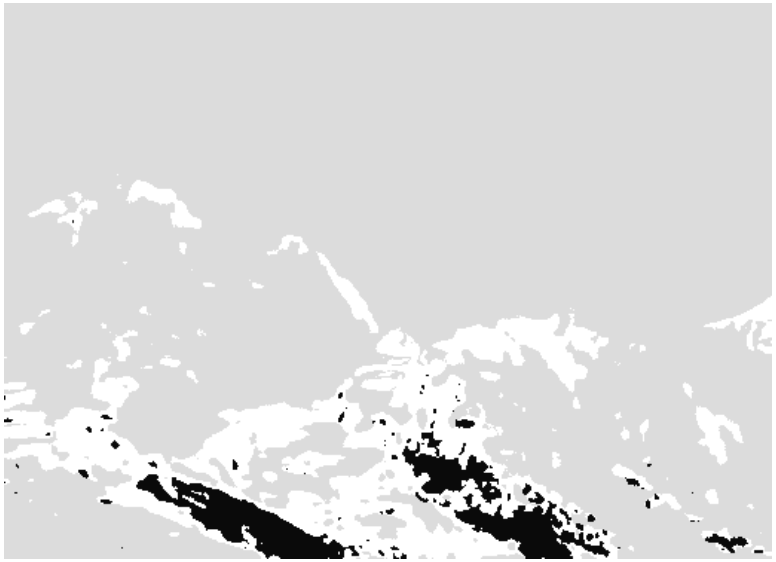


```
int kBands = 3; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

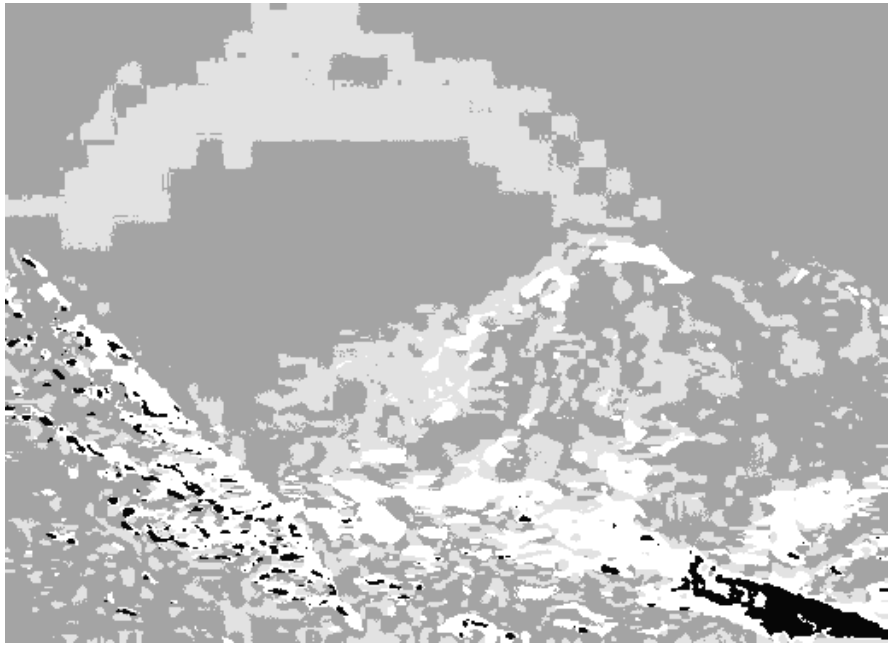
int kBands = 8; imageSegmentation.applyUsingKMPP(gsImg1, kBands);

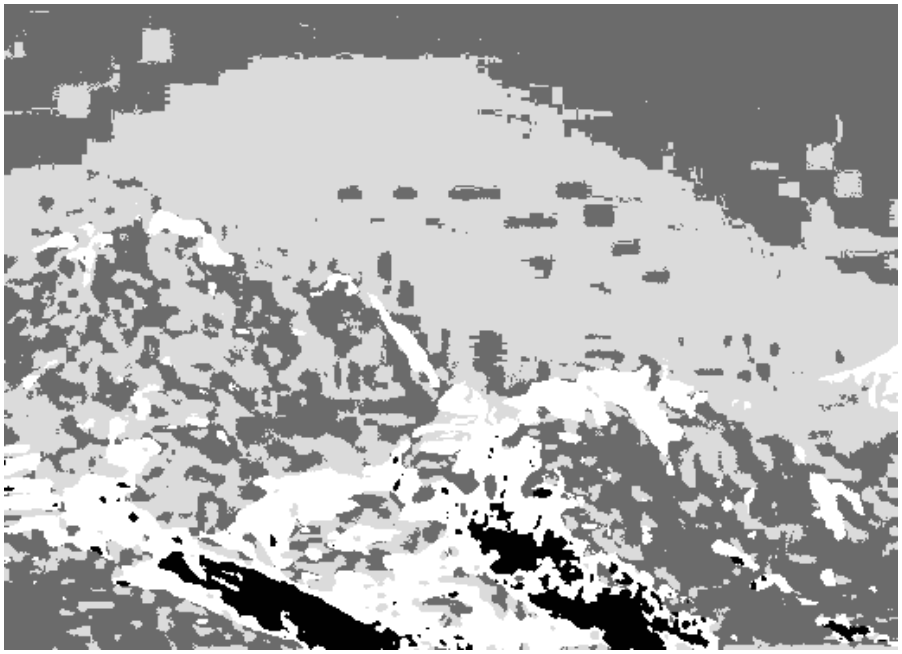
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);



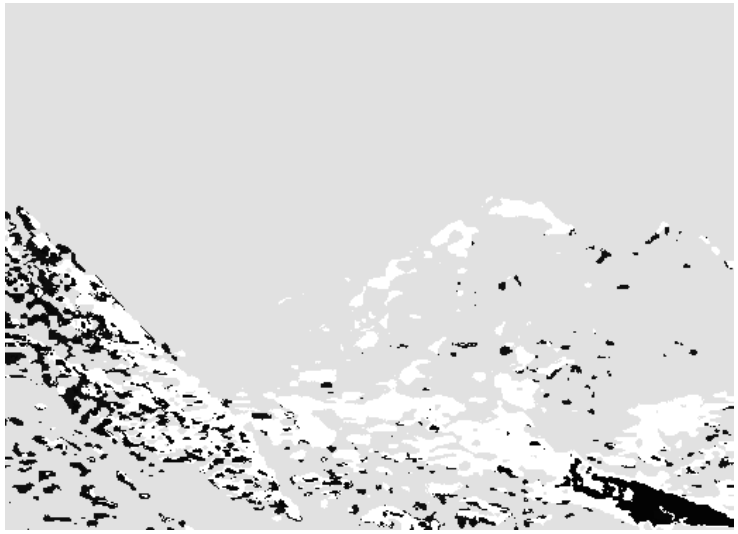


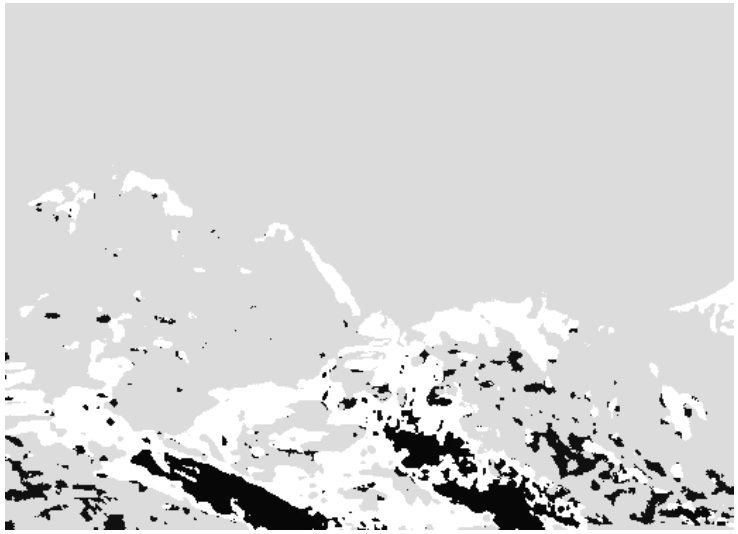
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);



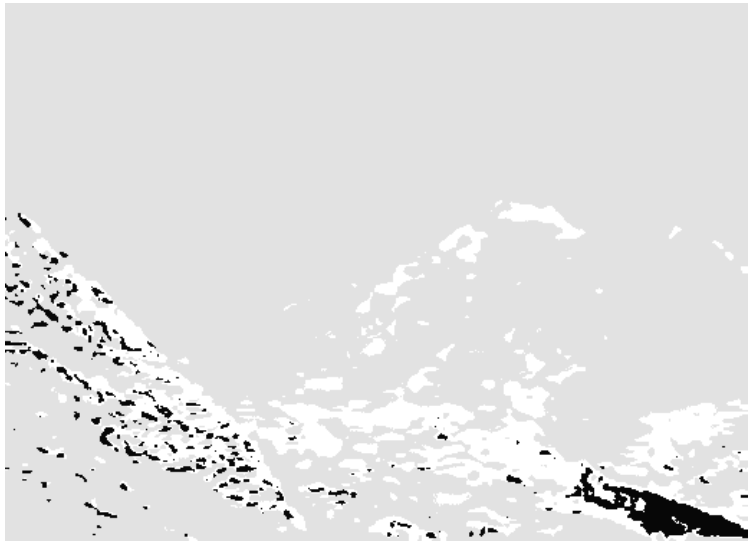


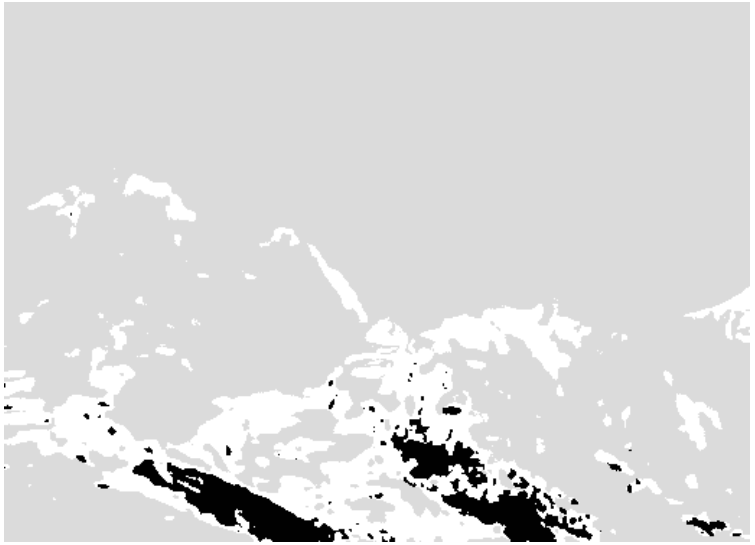
int kBands =8; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);



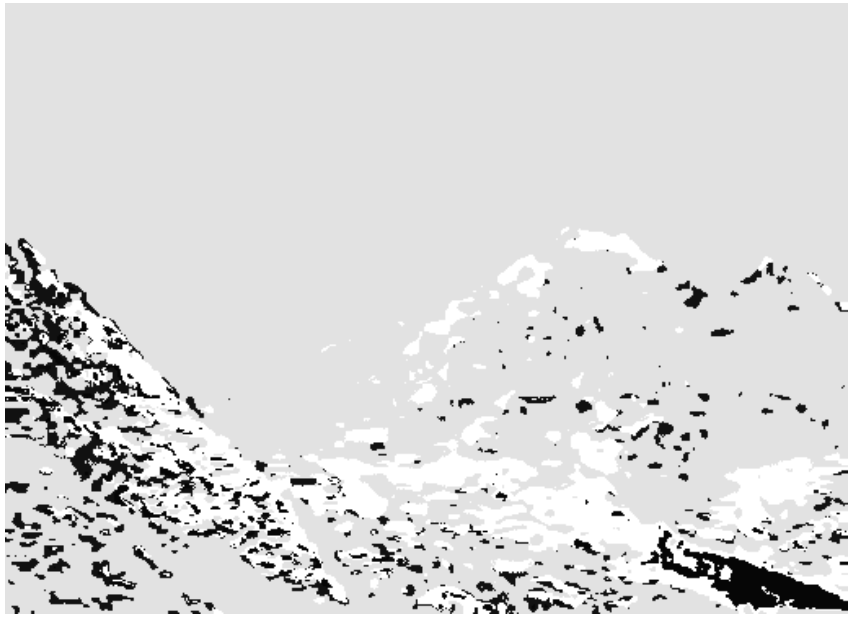


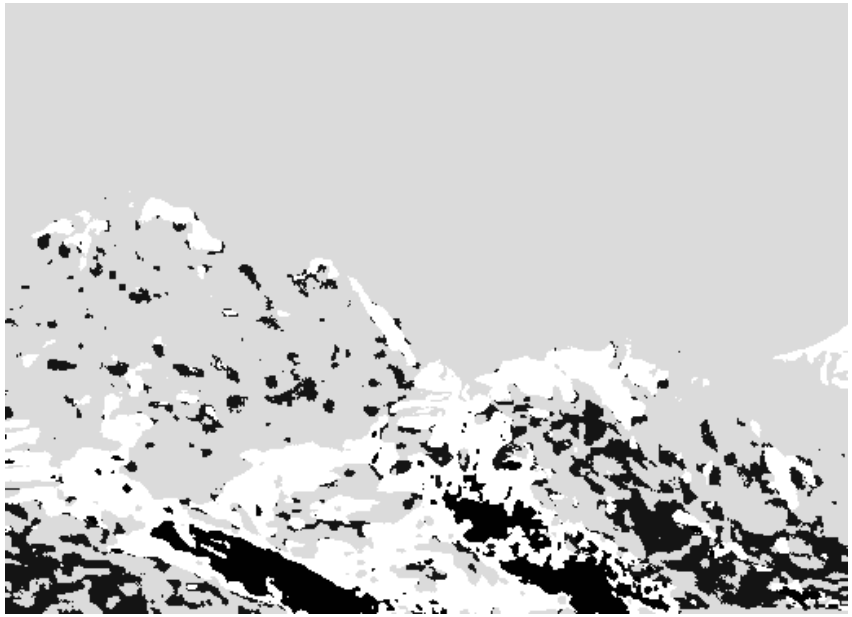
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);
```



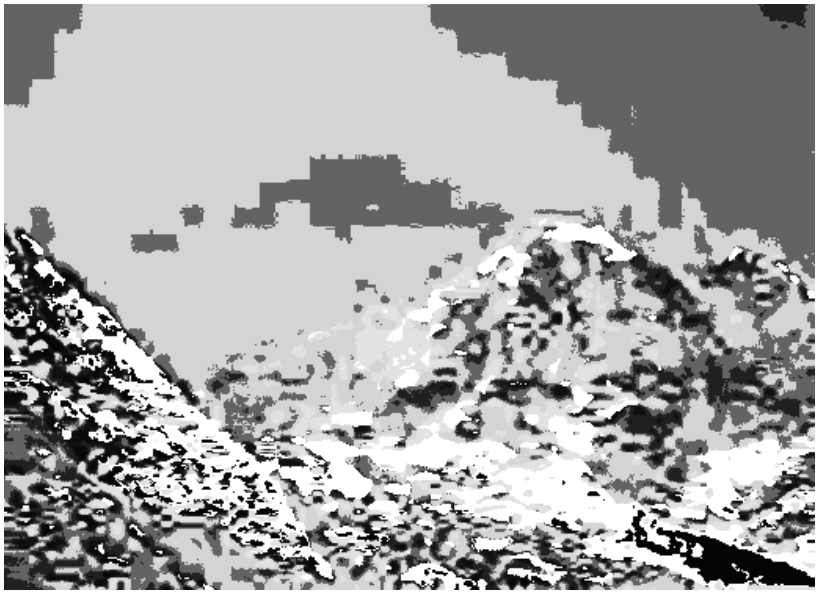


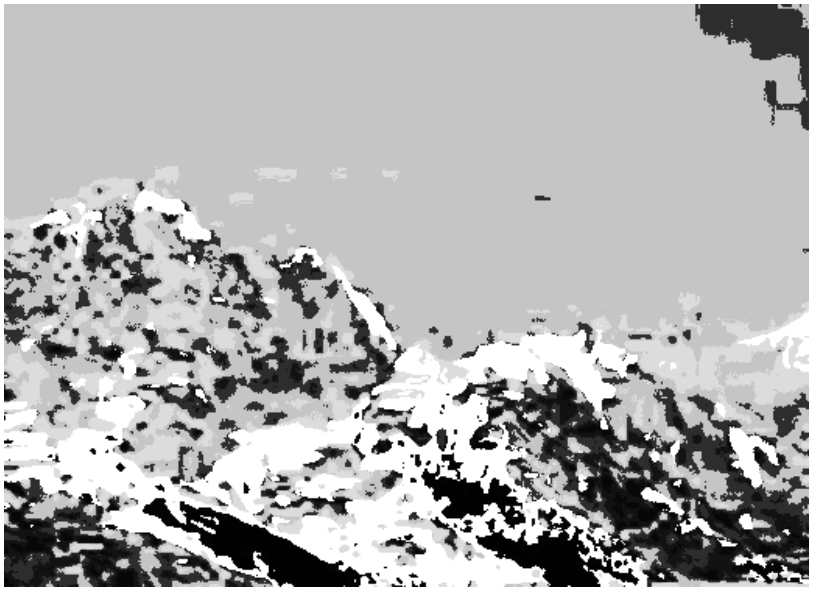
```
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);
```



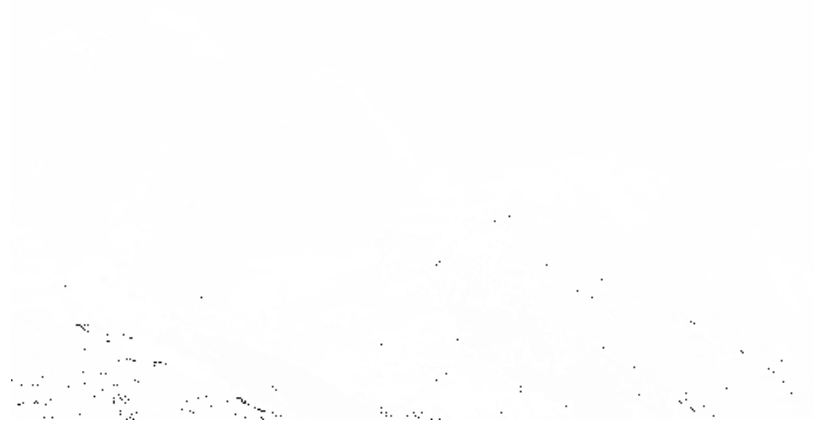


```
int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);
```

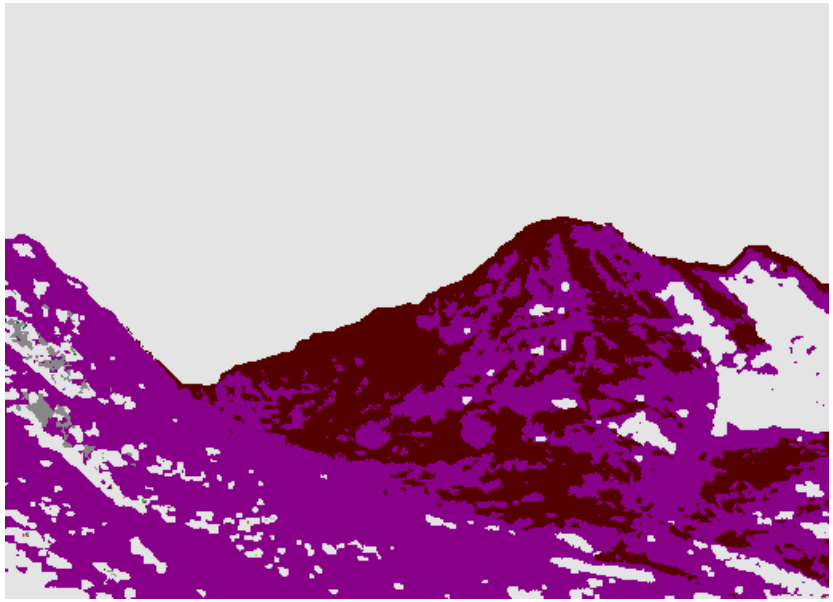


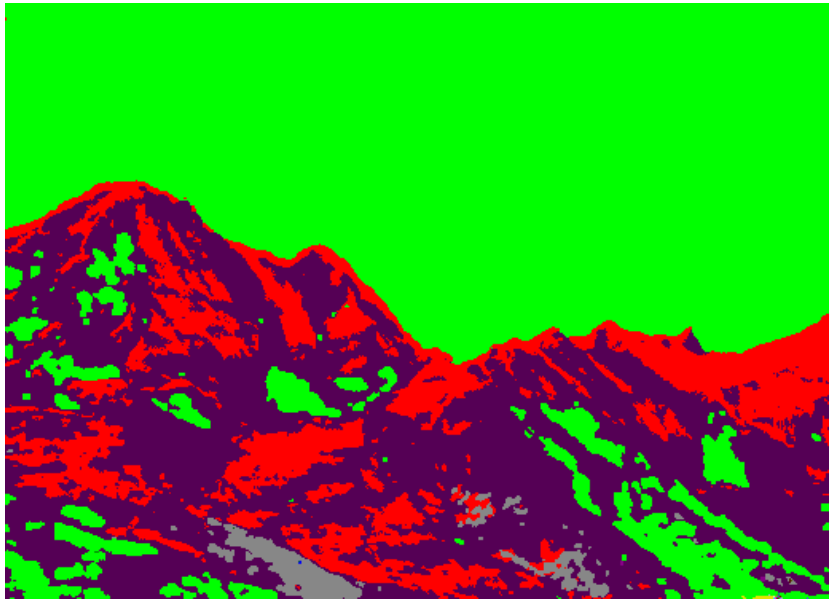


```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistogram(gsImg1, kBands);
```

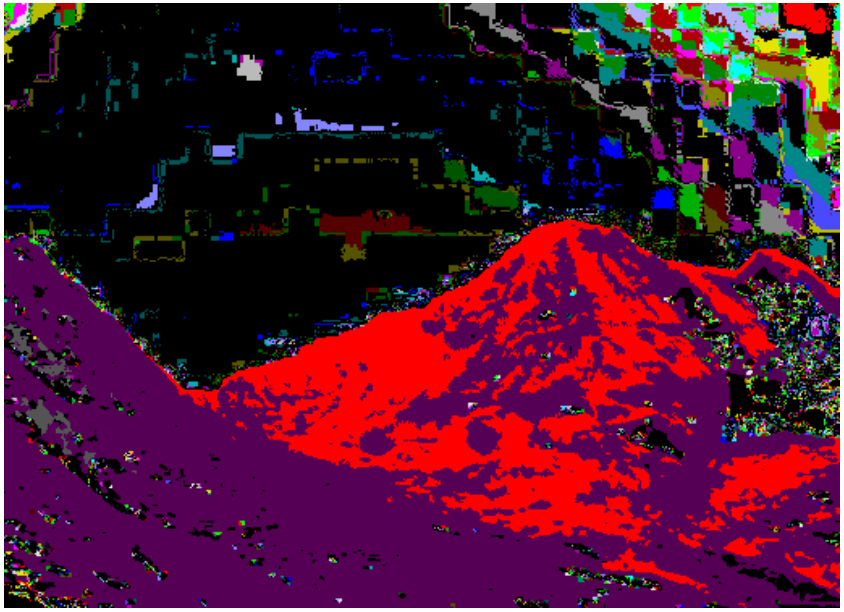


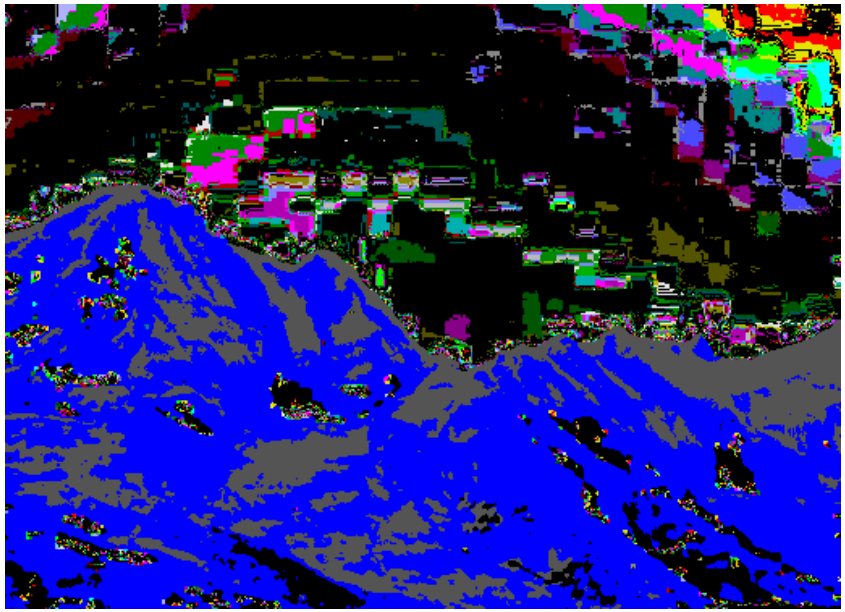
imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);



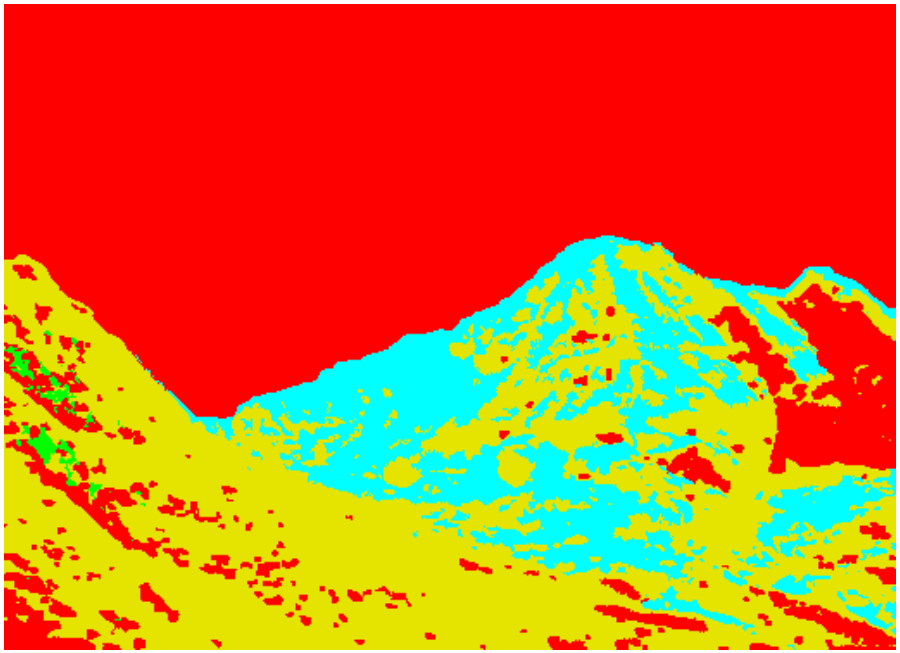


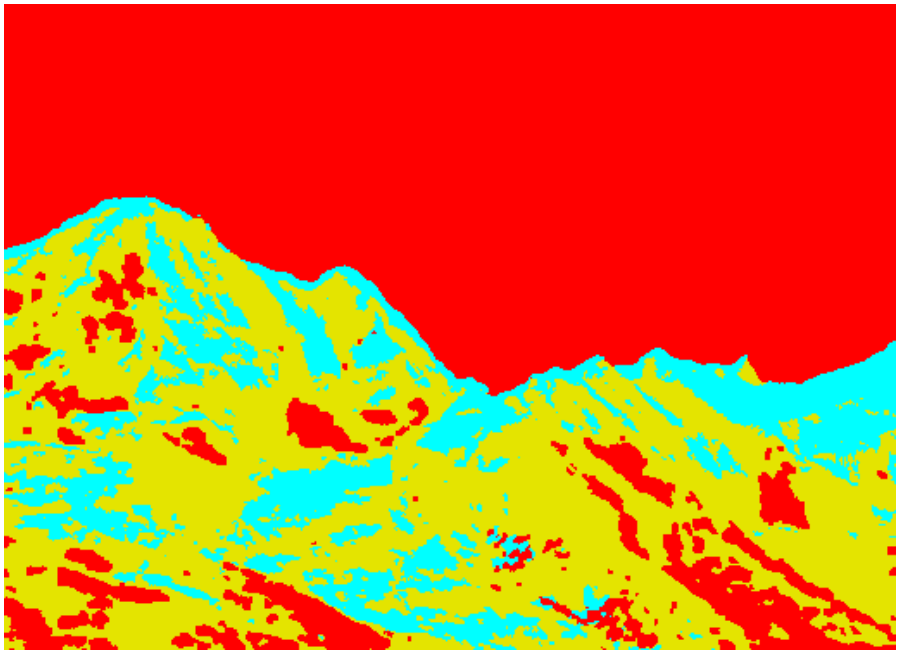
imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);



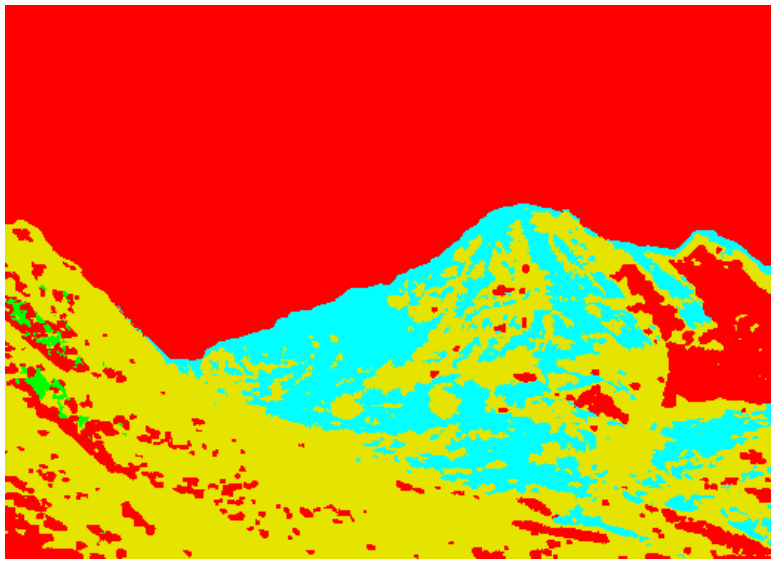


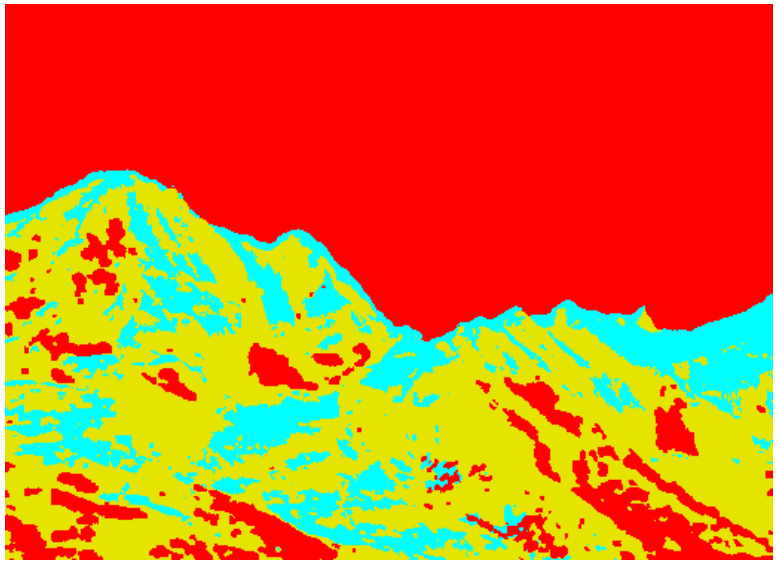
```
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, true);
```





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





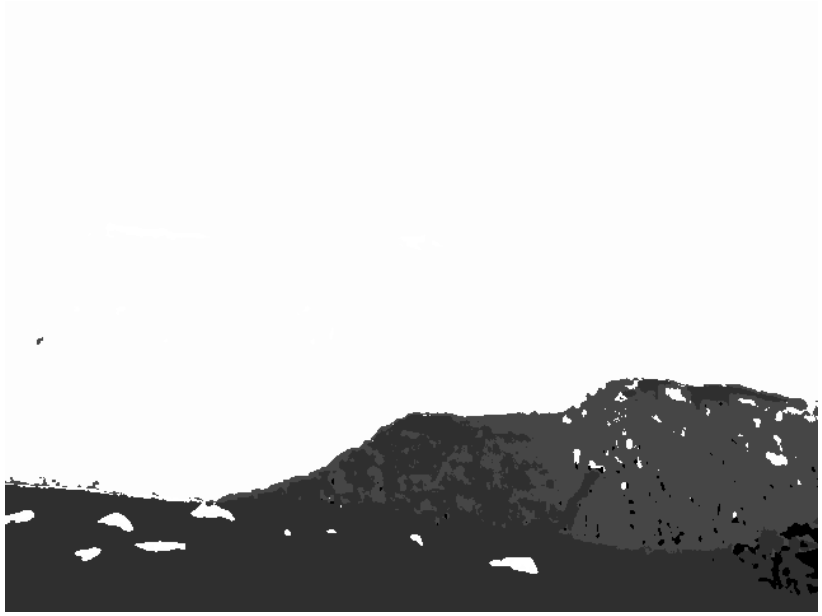
```
int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

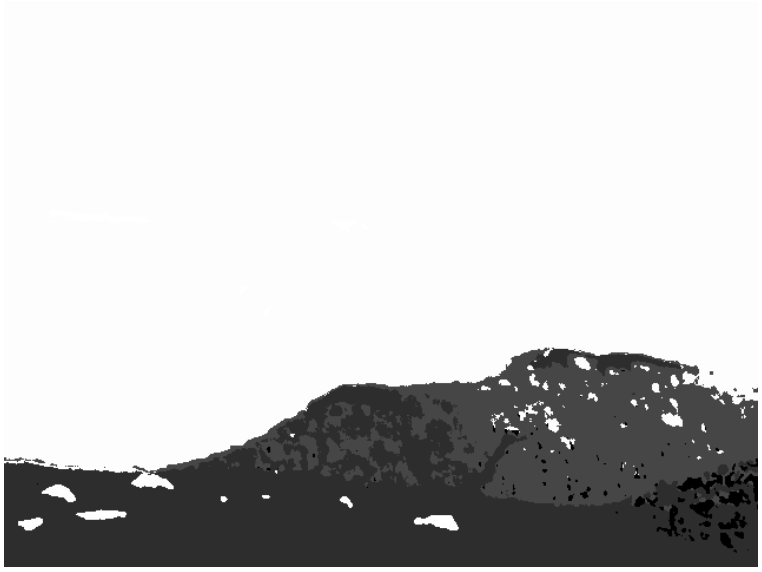
```
int kBands = 3; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```

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

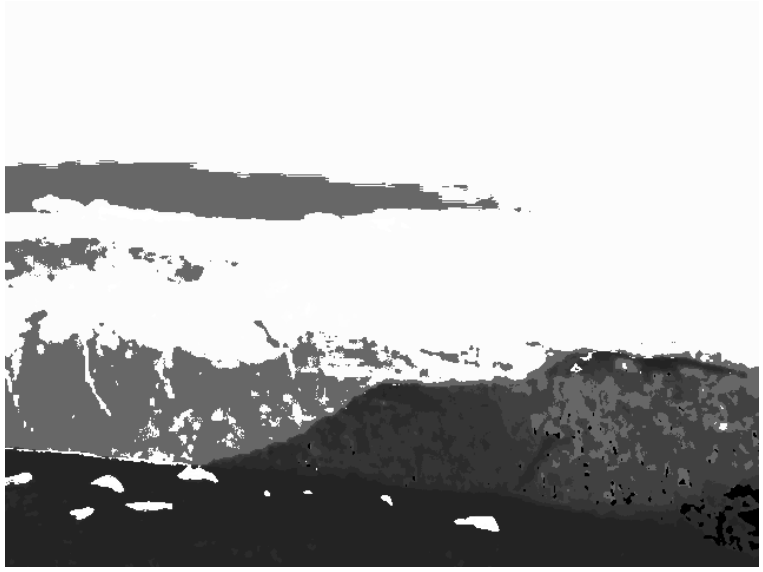
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);
```

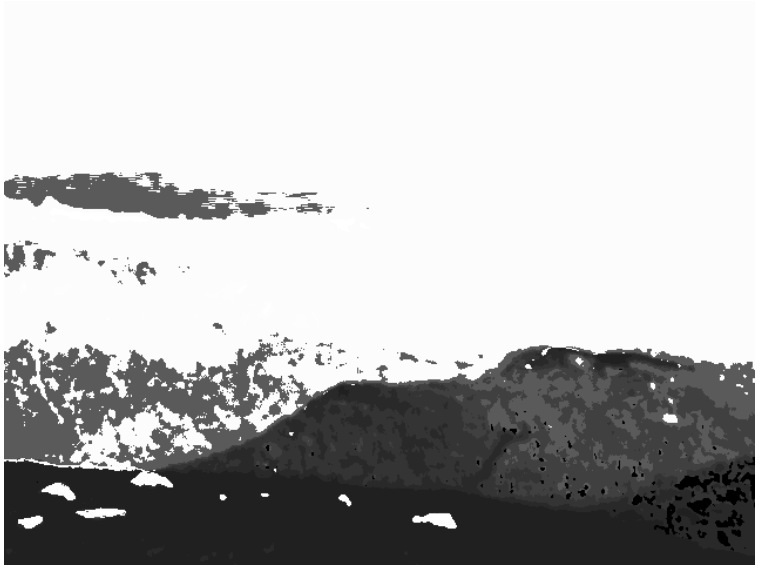
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);





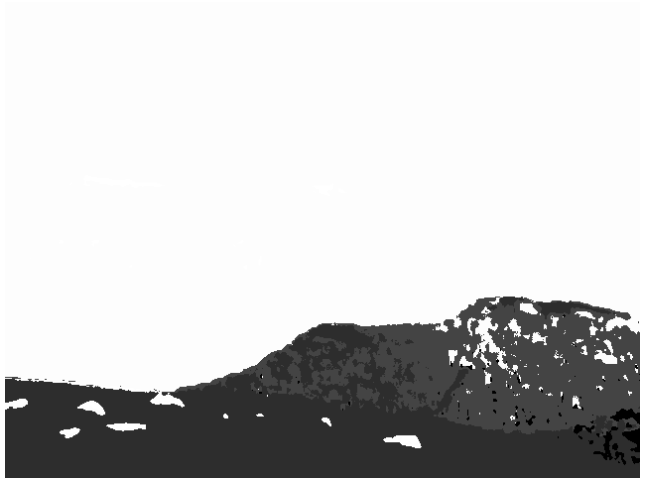
int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);

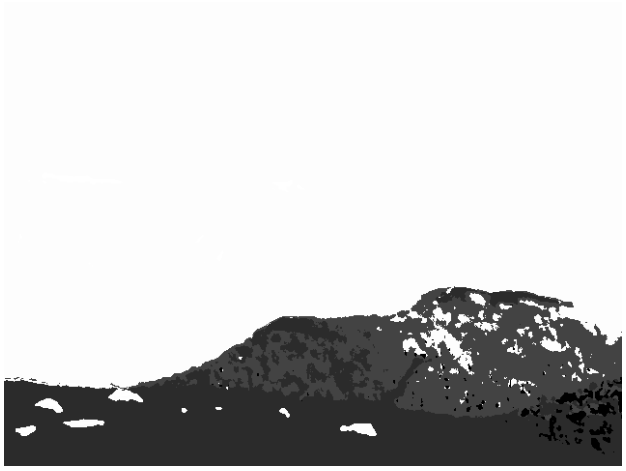




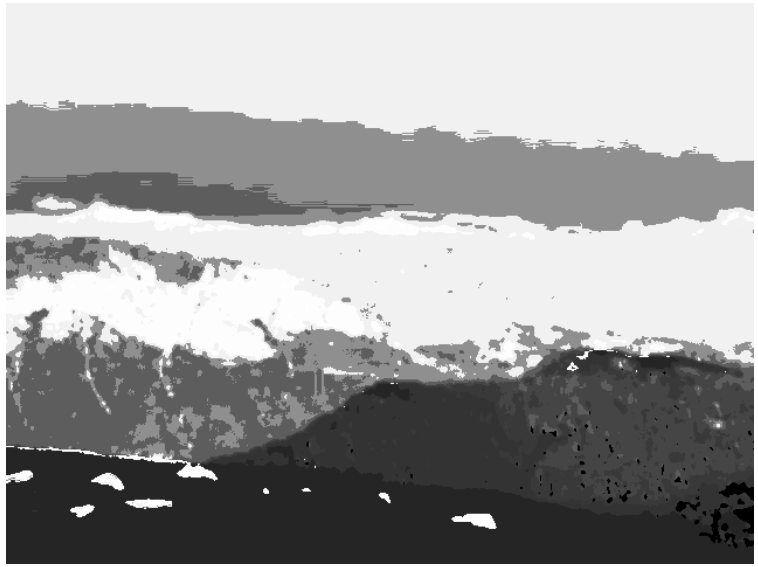
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

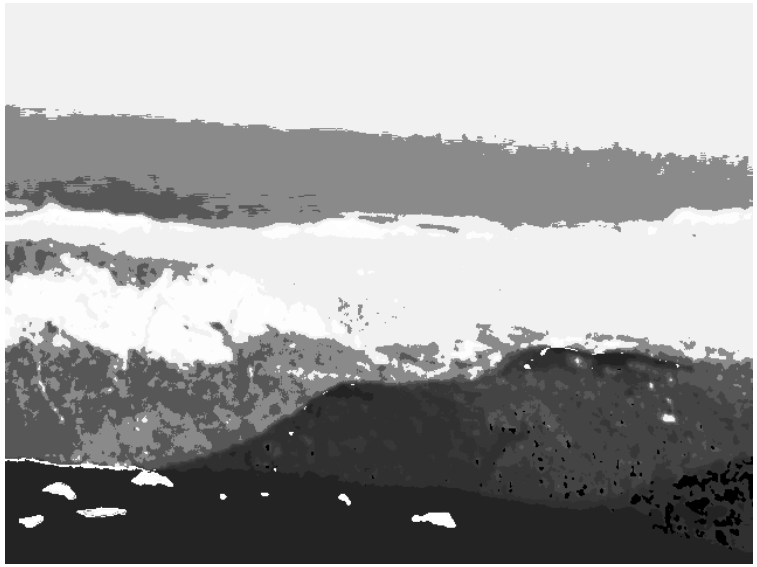
int kBands =3; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);





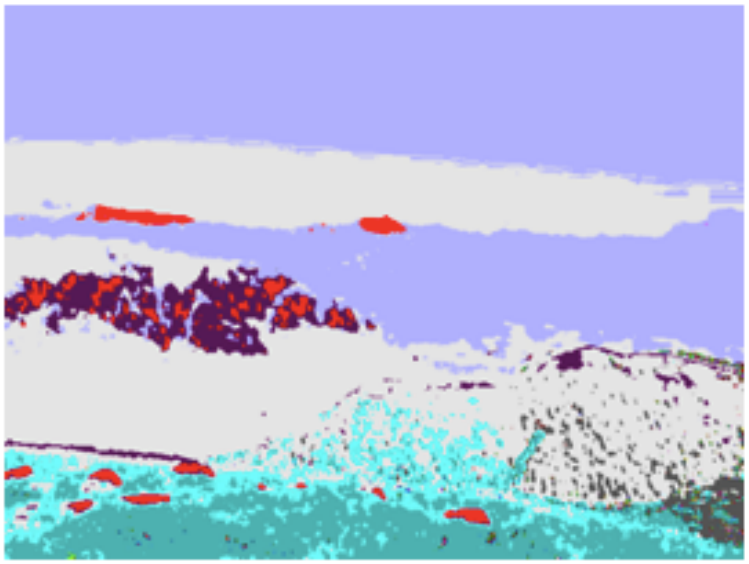
int kBands =8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq

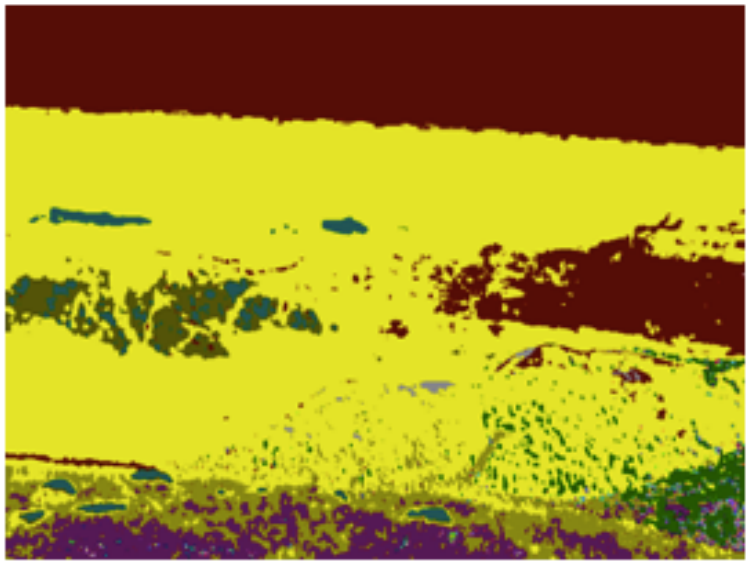




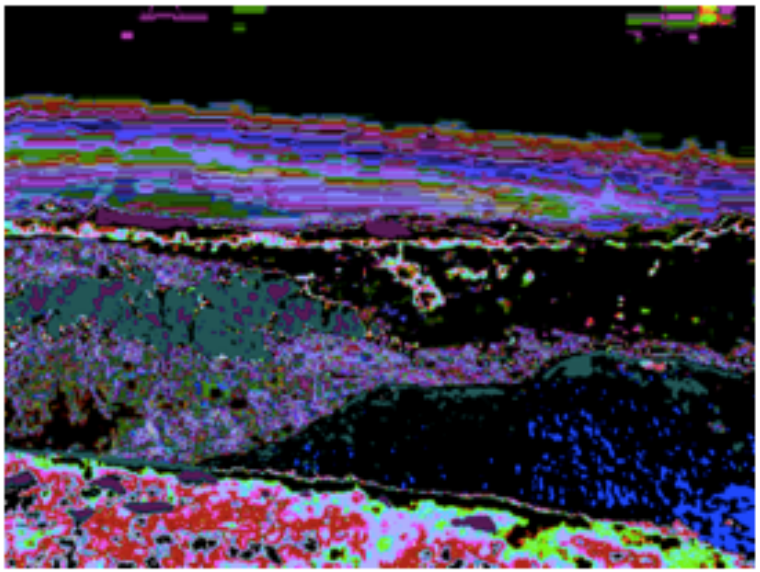
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistogram(gsImg1, kBands);

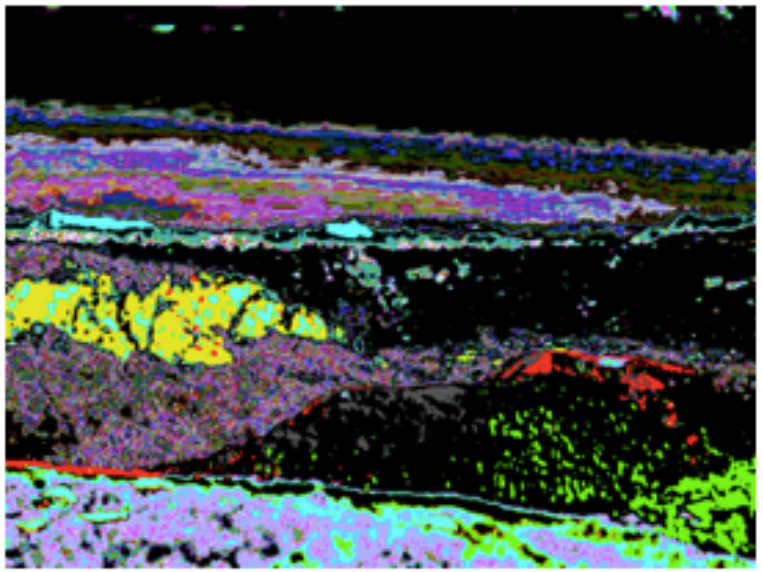
imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);



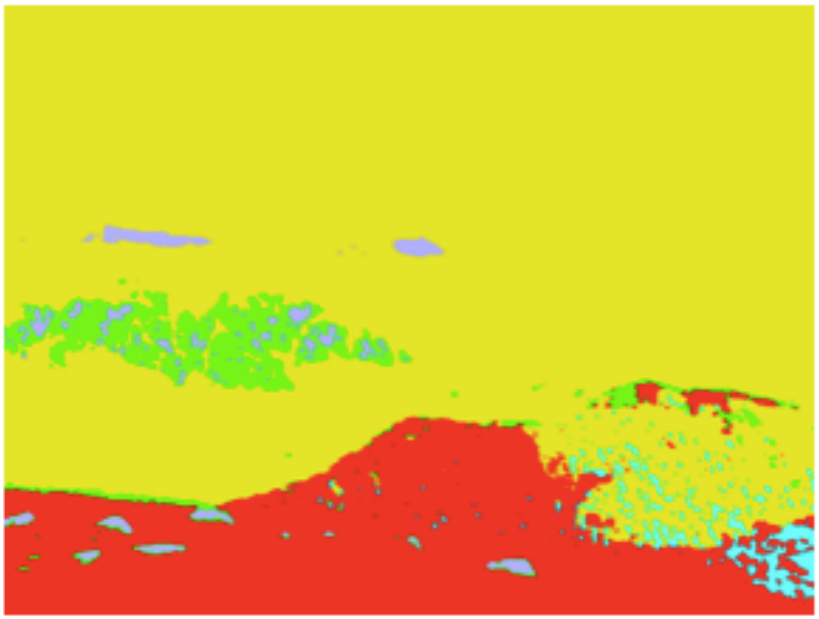


imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);

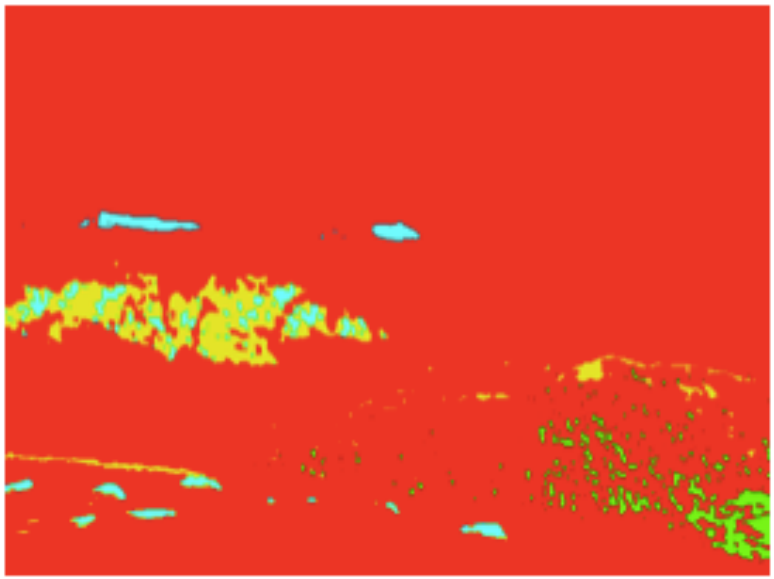


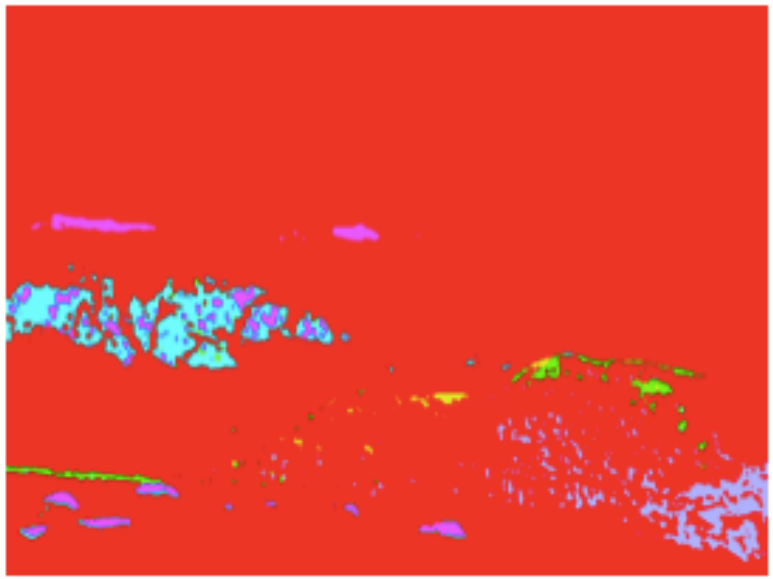


imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, true);



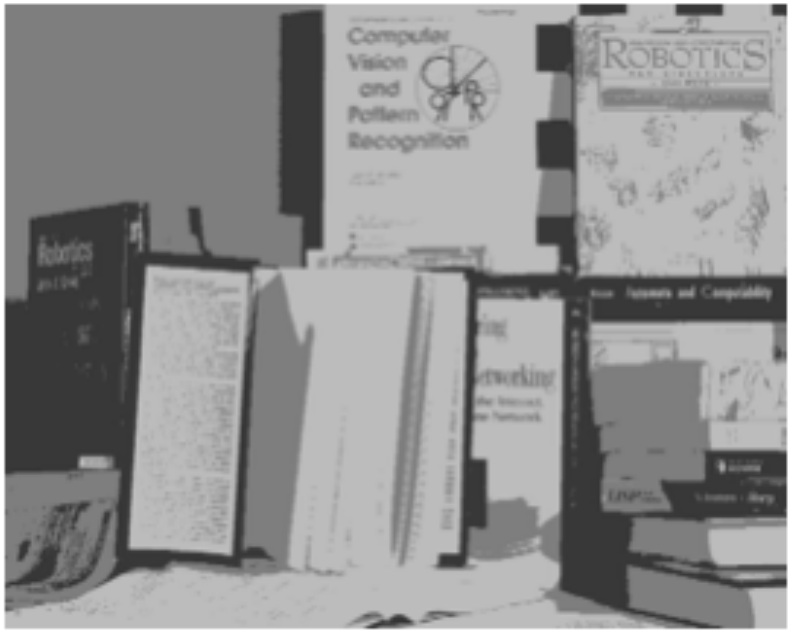
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, 0.2f, true);

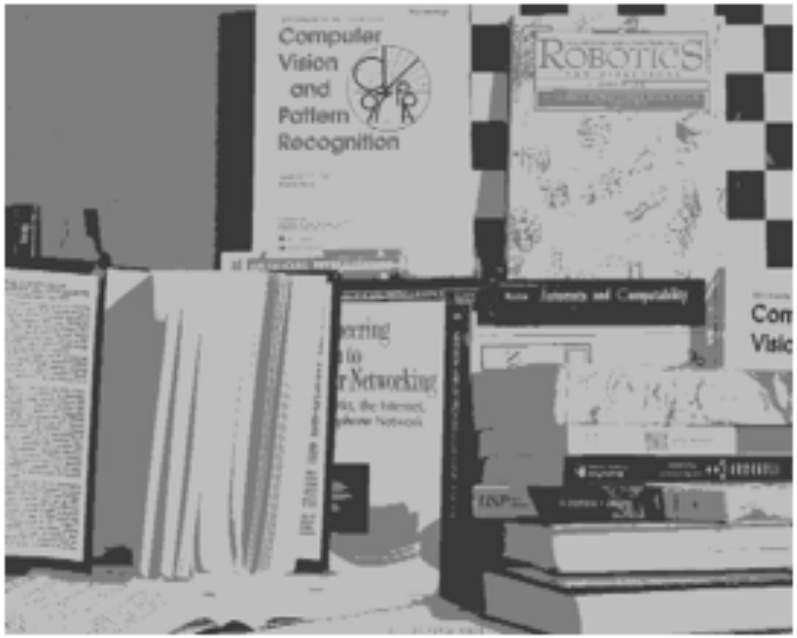




int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);

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



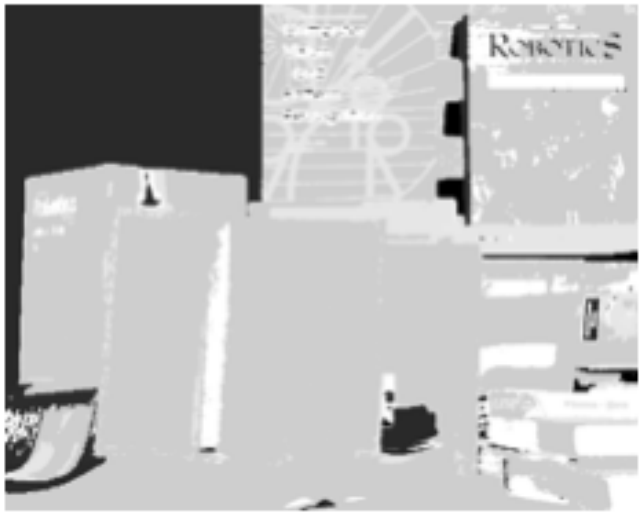


int kBands = 8; imageSegmentation.applyUsingKMPP(gsImg1, kBands);

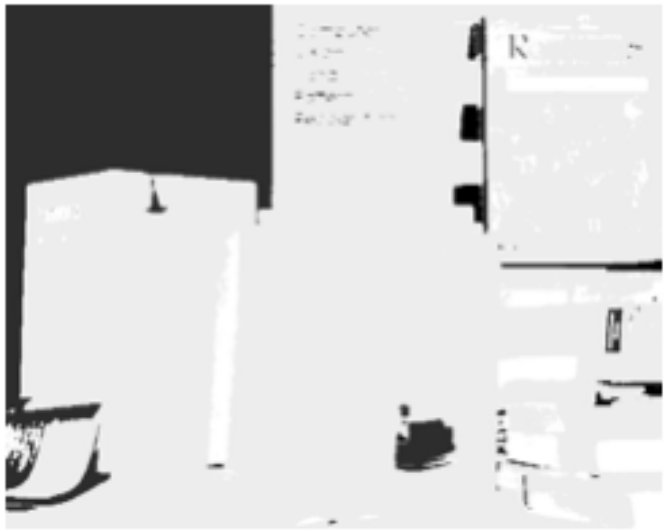
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);

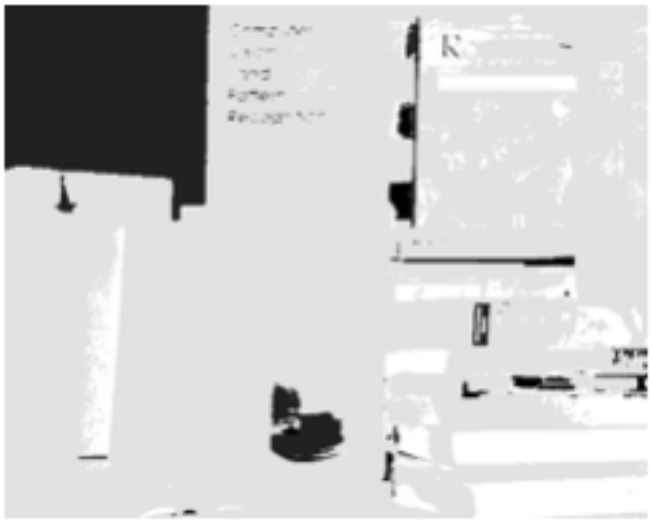
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);

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

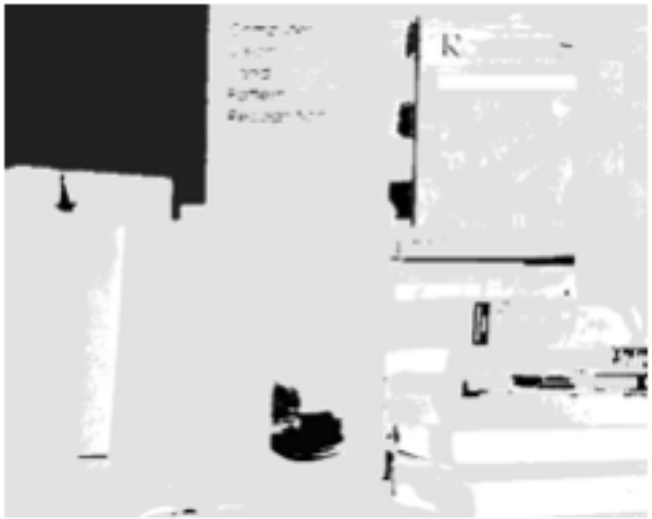


int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

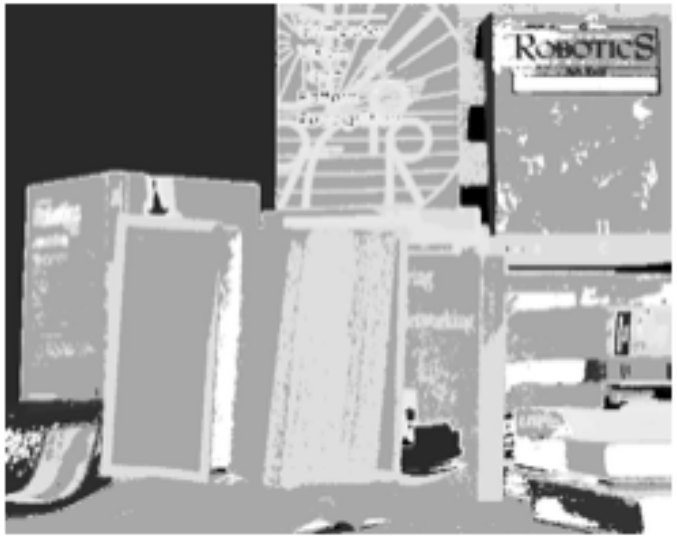


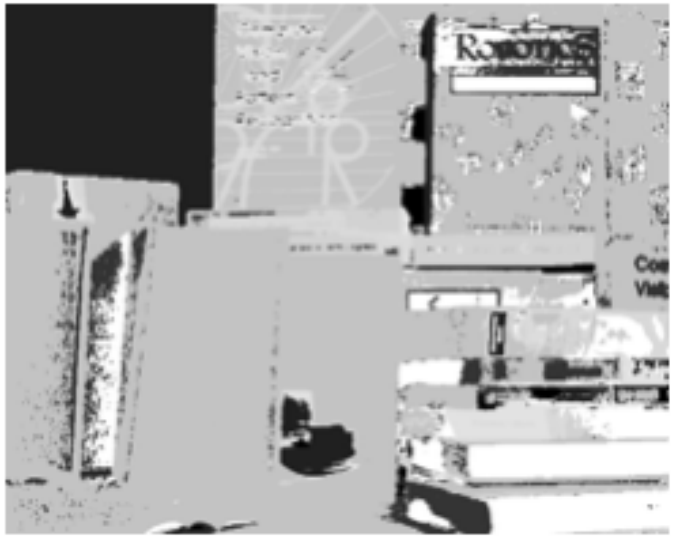


int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

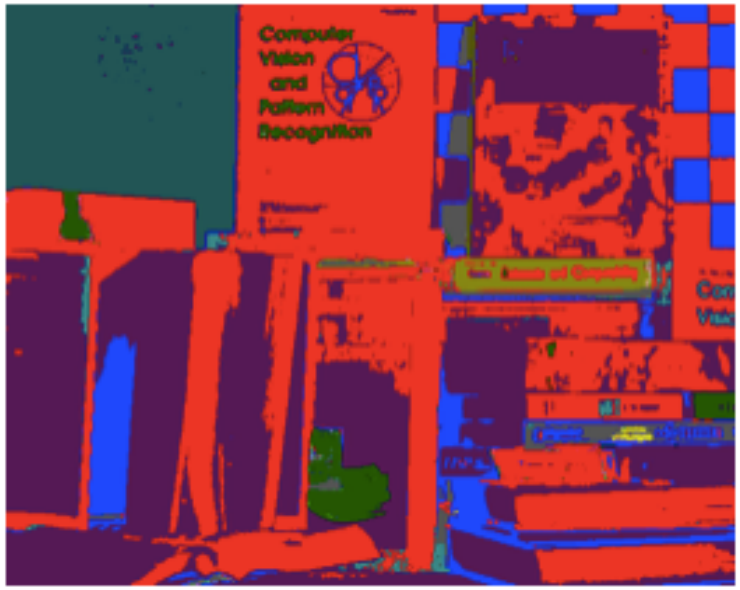


int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1, kBands);

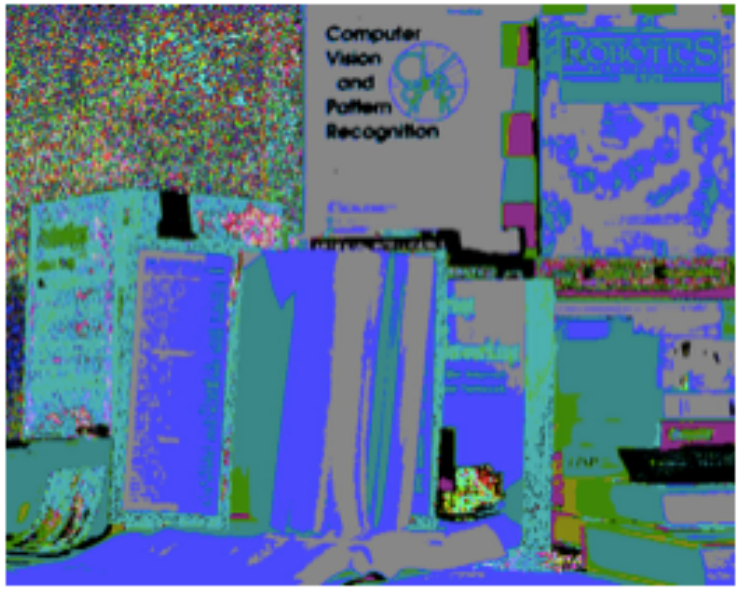


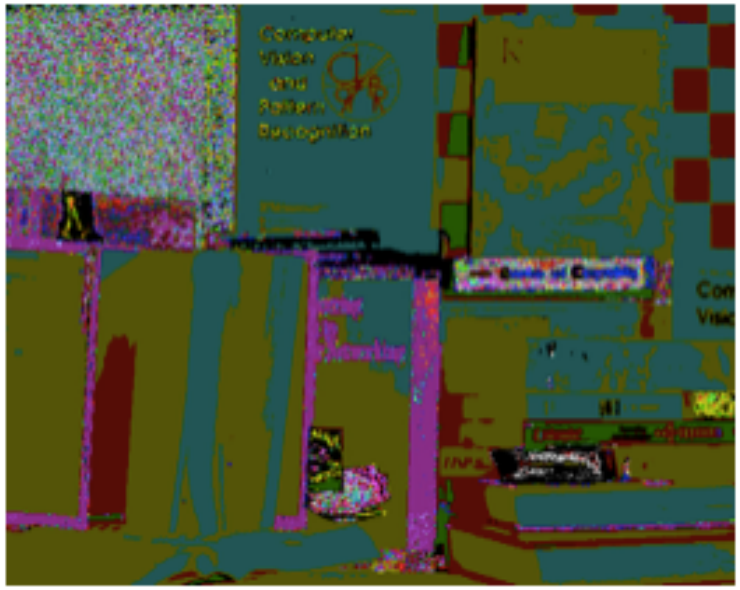


imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);

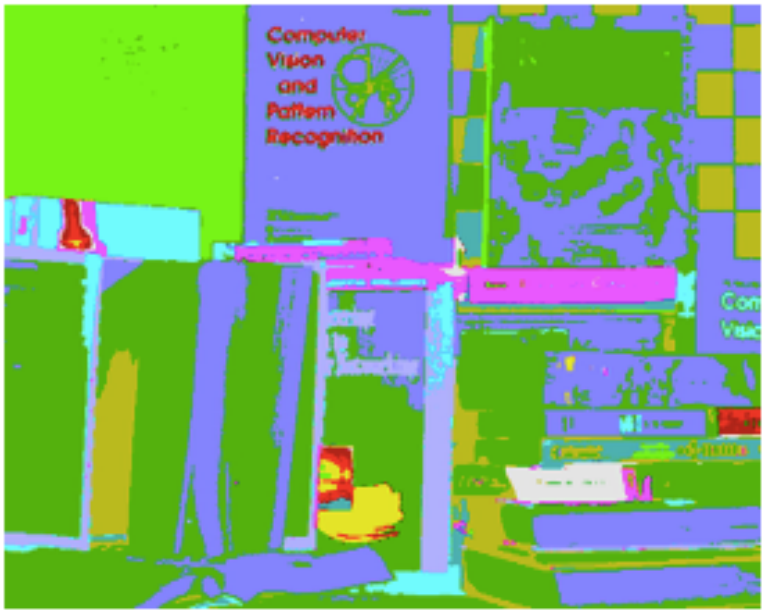


imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);

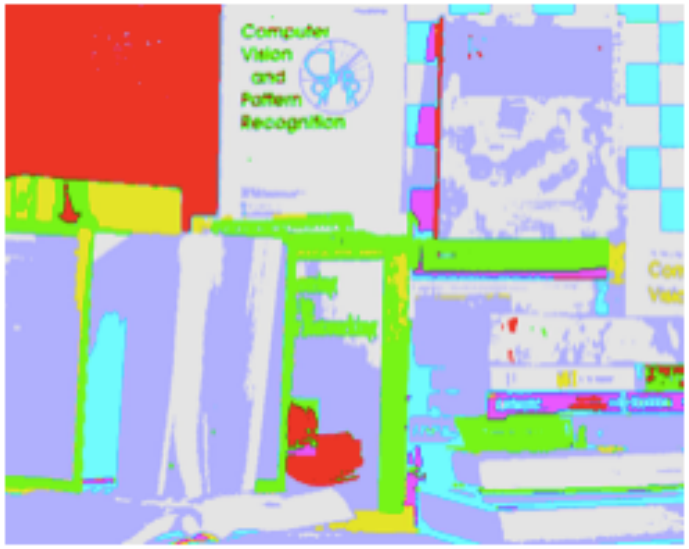




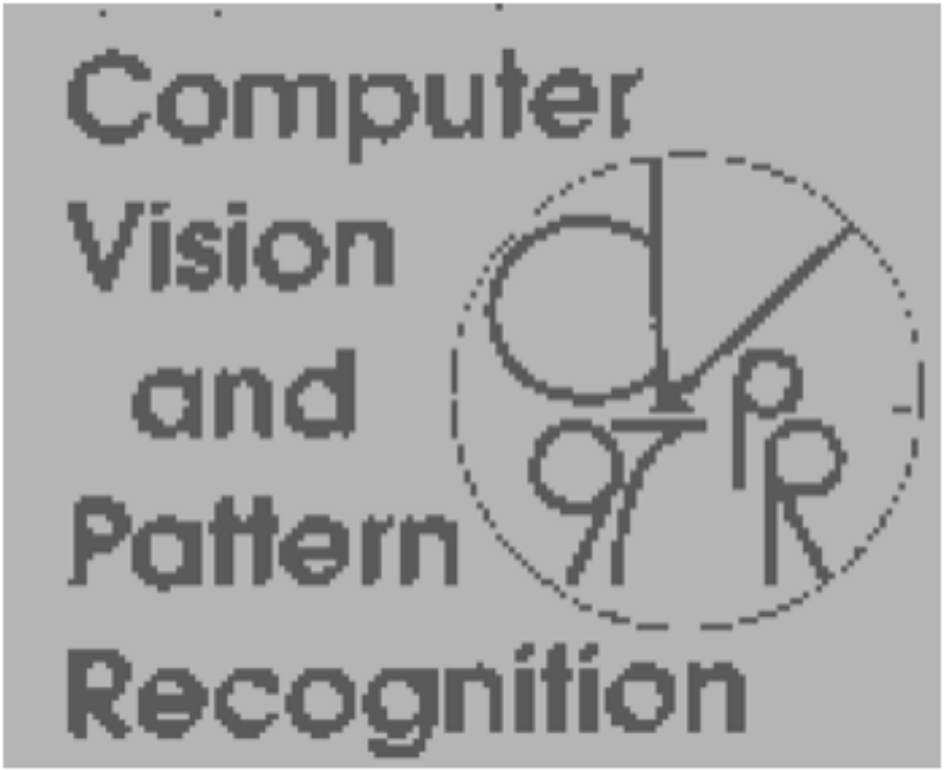
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, true);

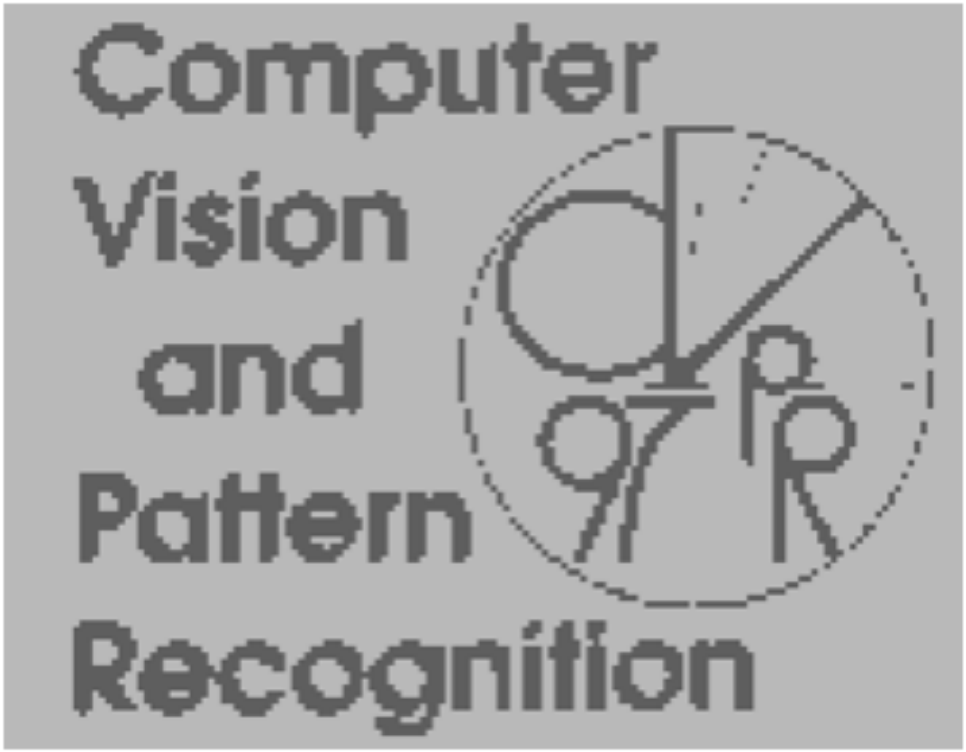


imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gsImg1, 0.2f, true);

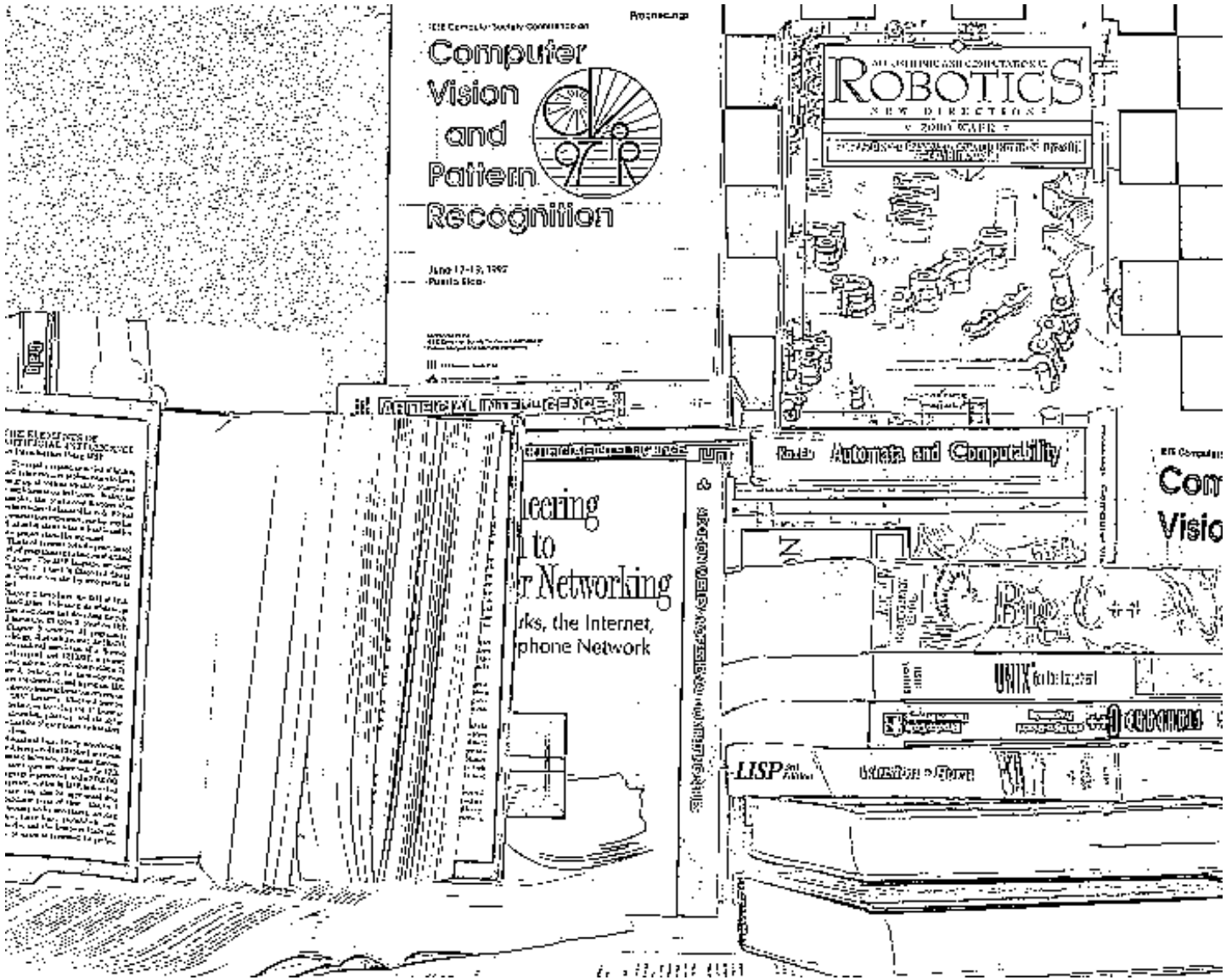


int kBands = 2; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
zoom in

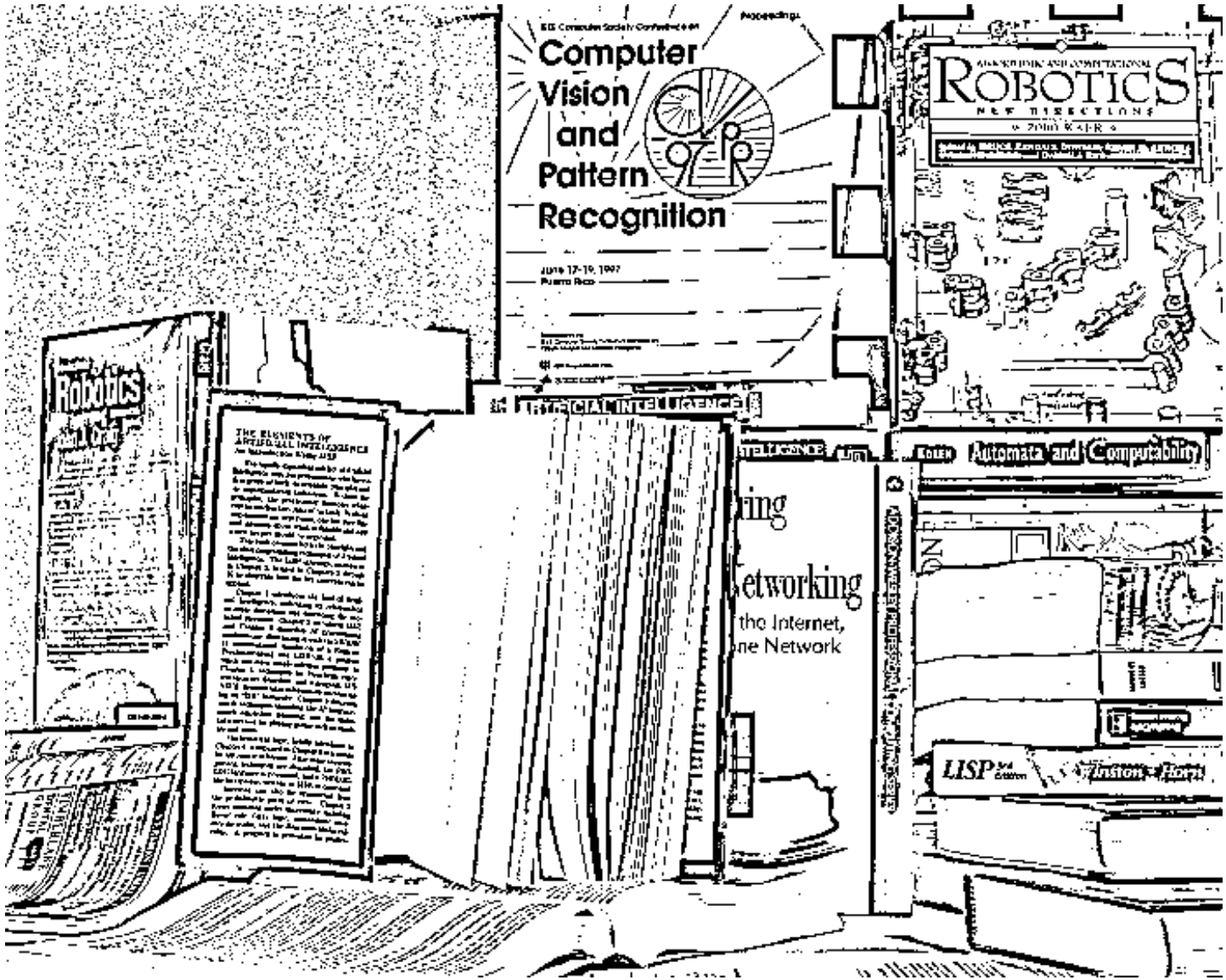




adaptive means, h=1

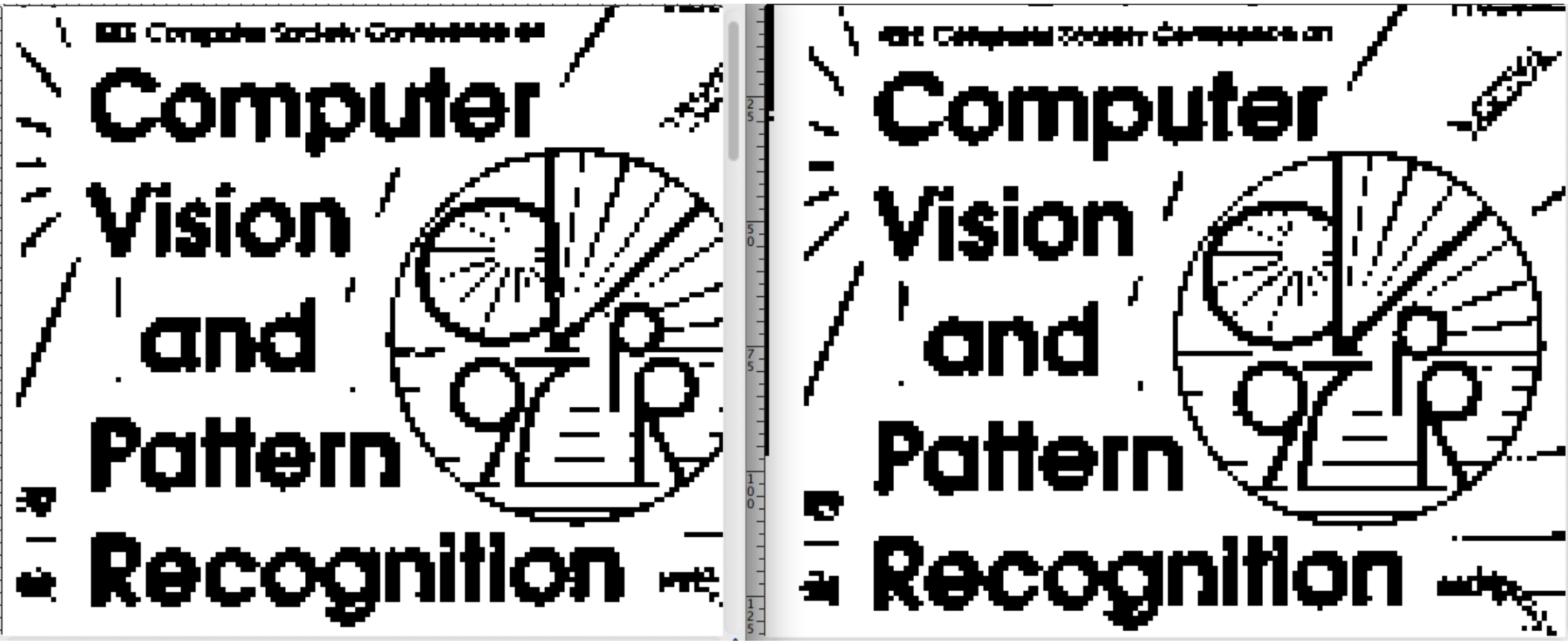


adaptive means, h=3

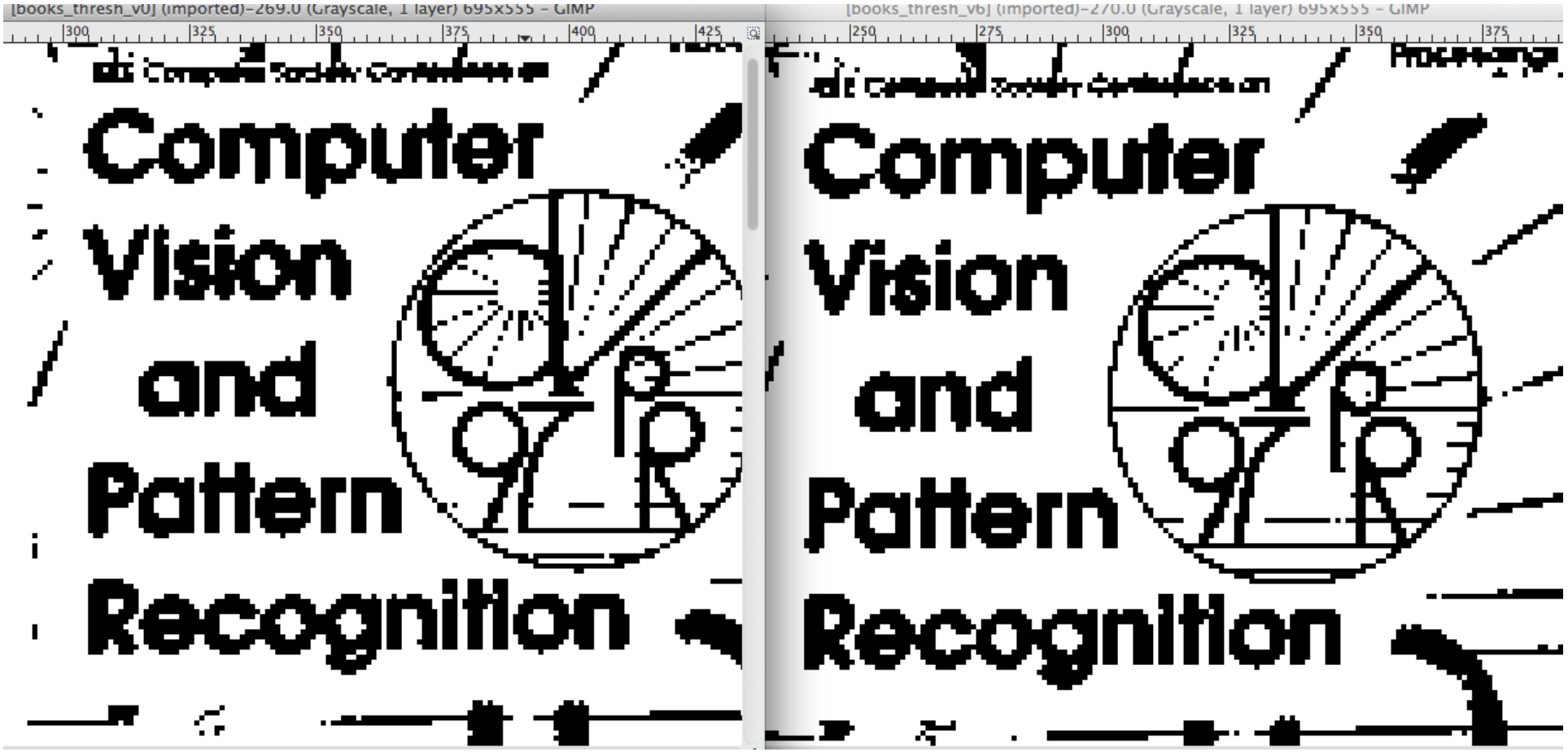


adaptive means, h=5

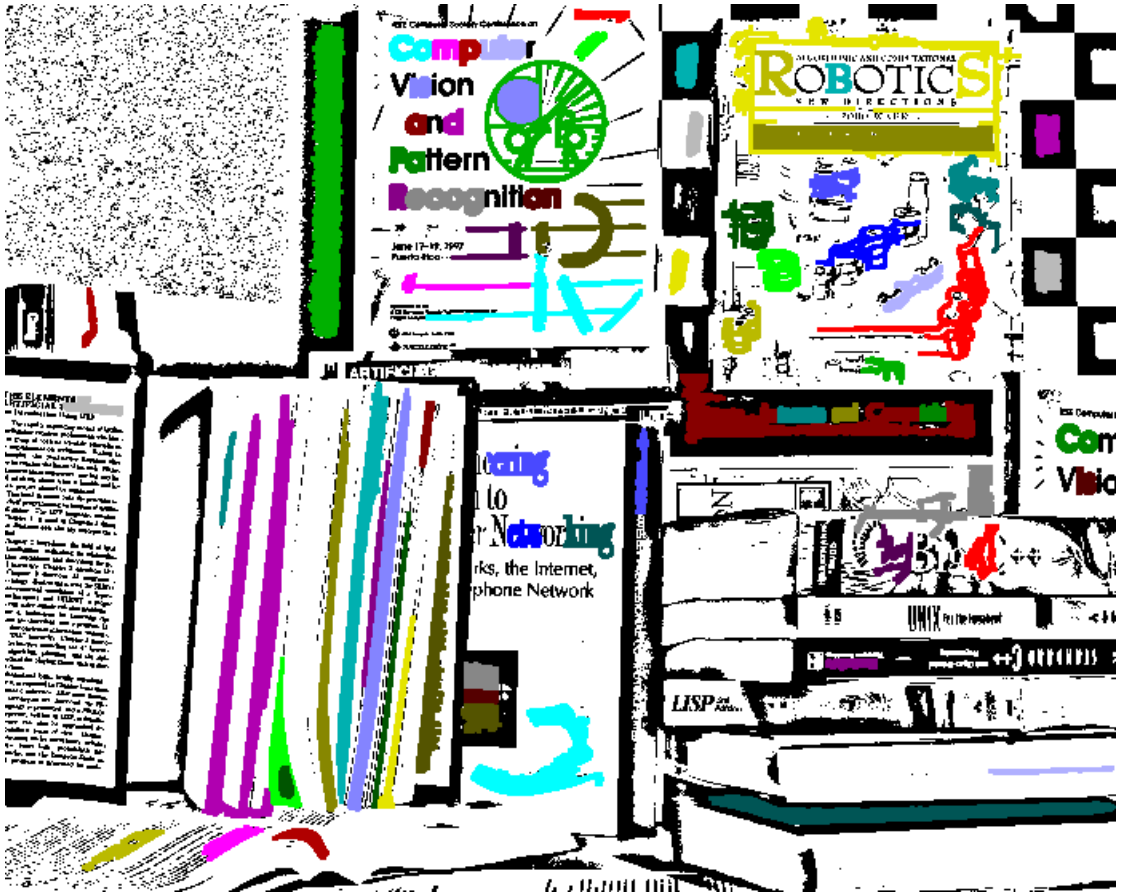
adaptive means, h=7



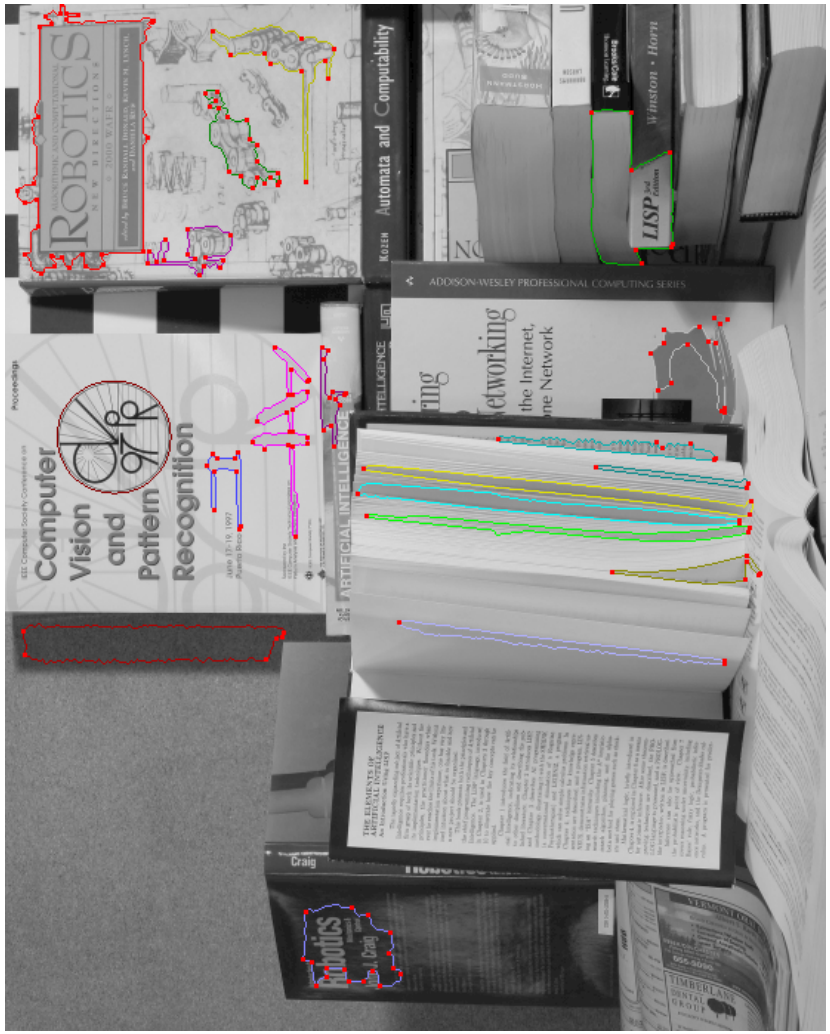
adaptive means, h=11 for adaptive to join close letters

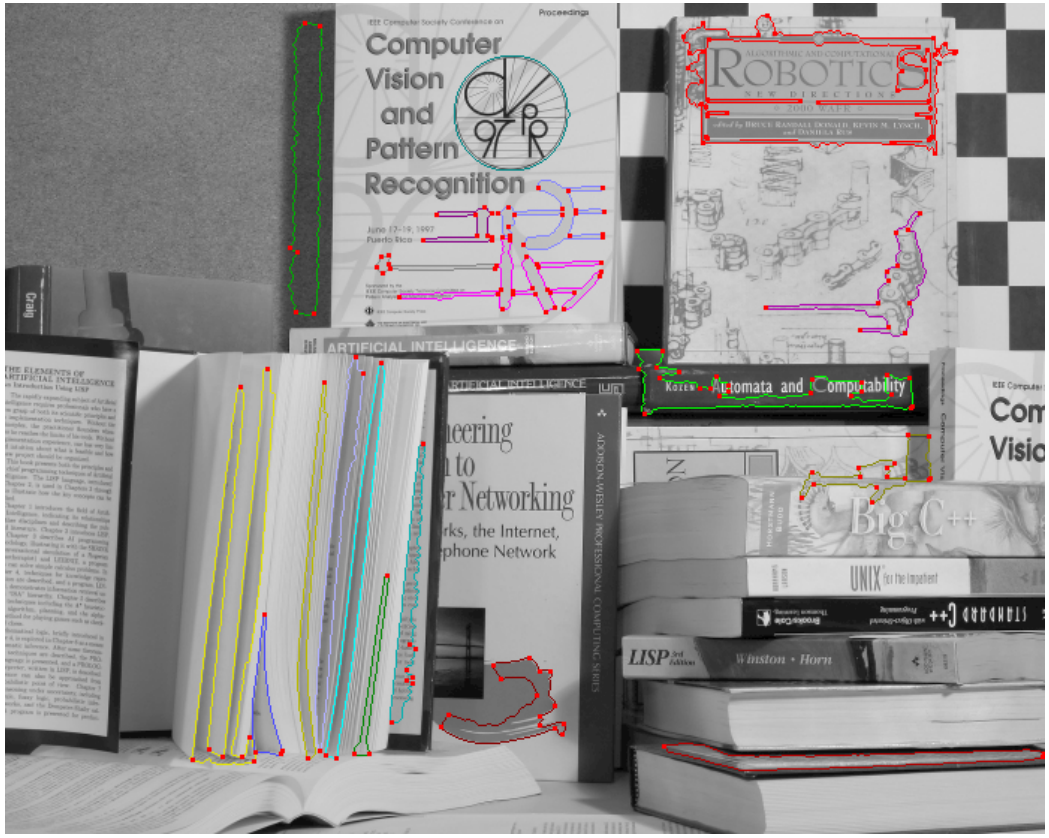


adaptive means, h=11 blobs (extr using scale calc code)



adaptive means, h=11 blob corners

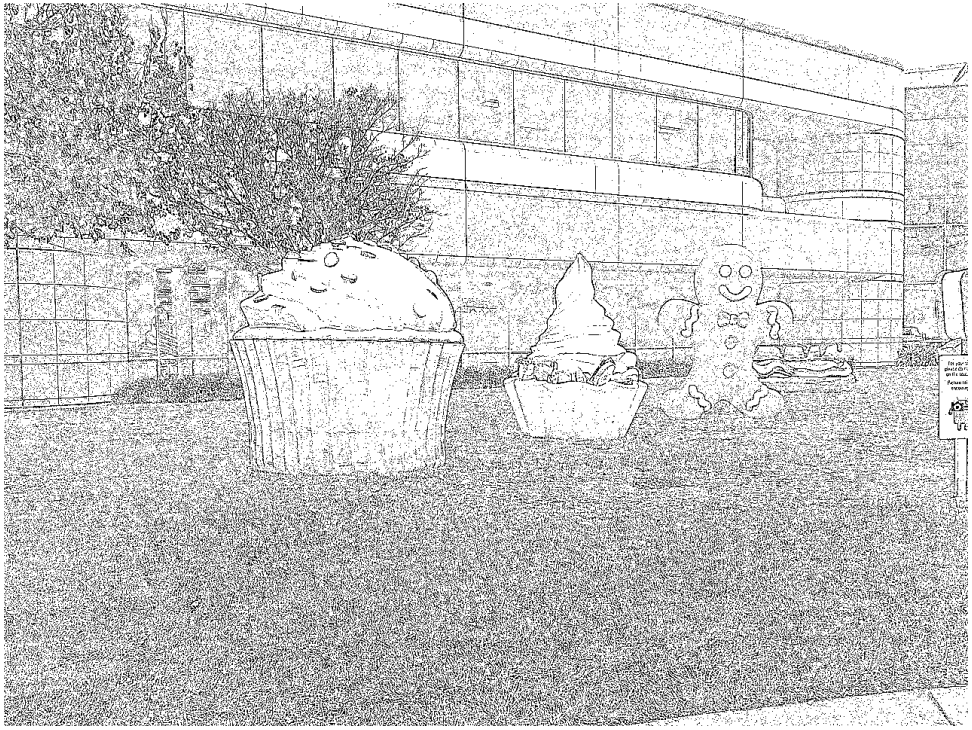




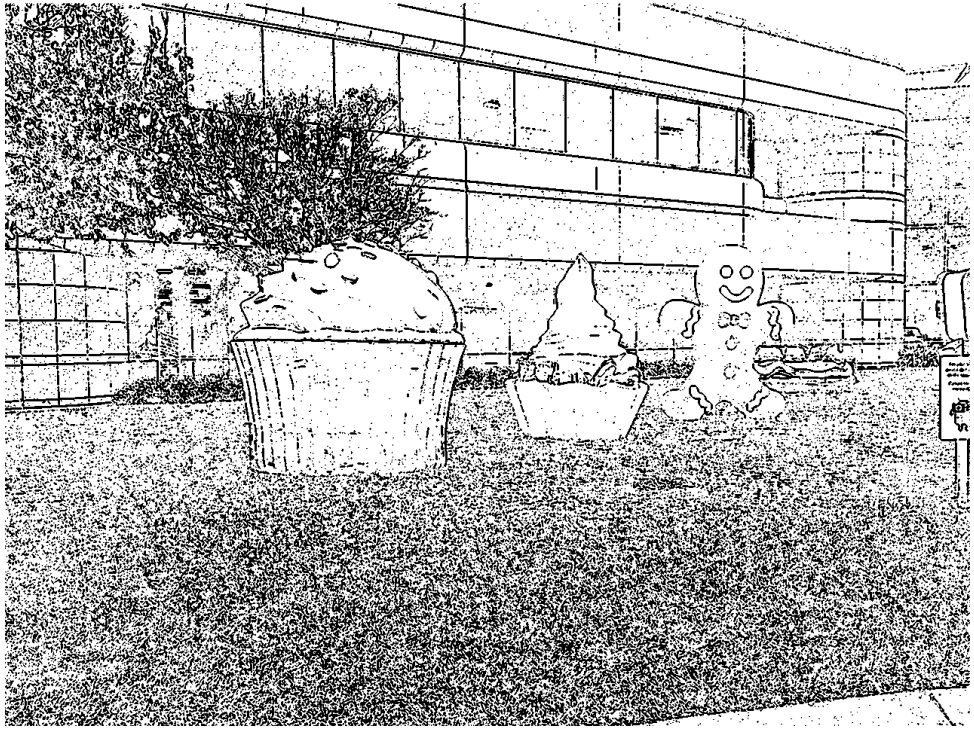
the letters are filtered out due to large nPerimeter compared to nInternal

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

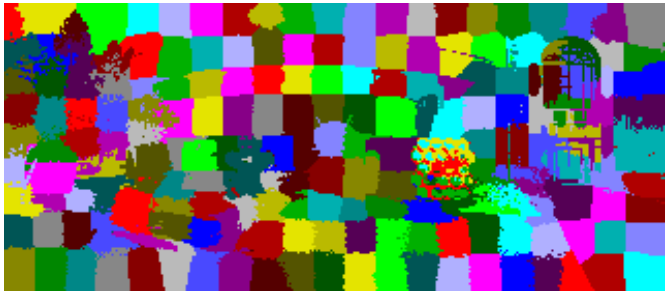
adaptive means, h=1

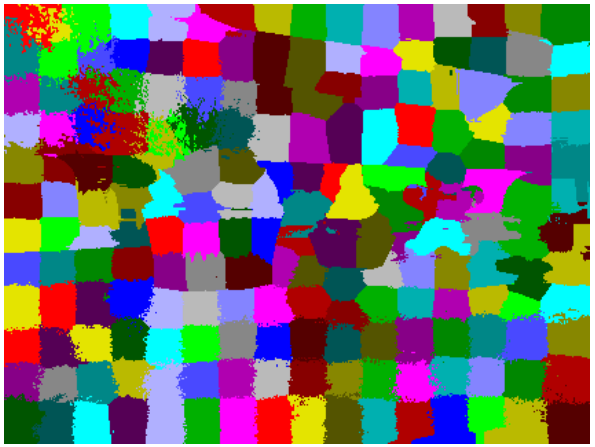
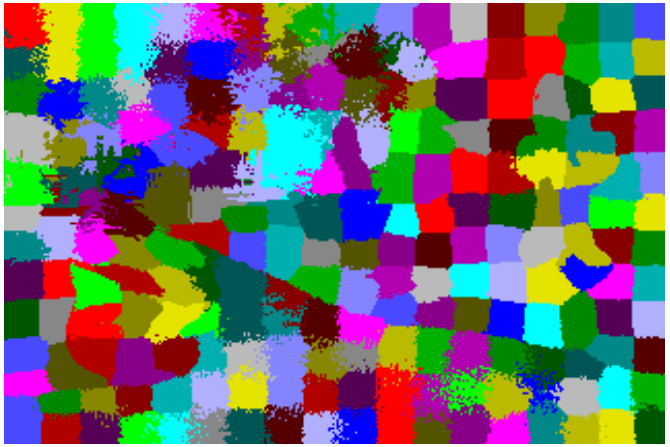


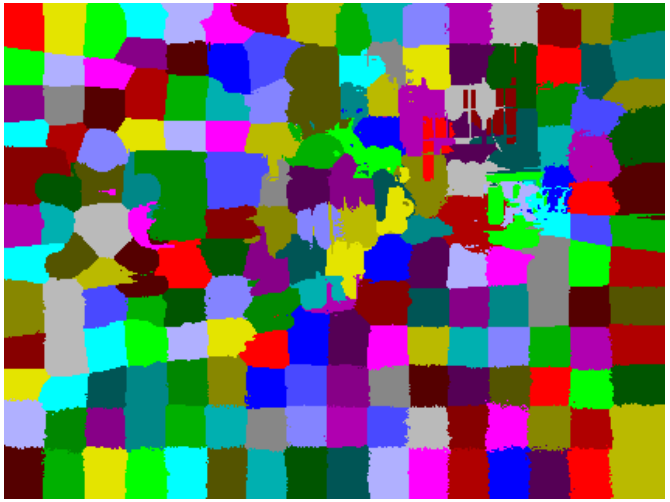
two median smooth with h=2



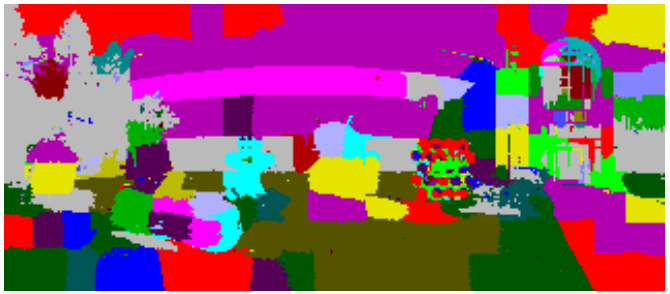
SLIC super pixels algorithm

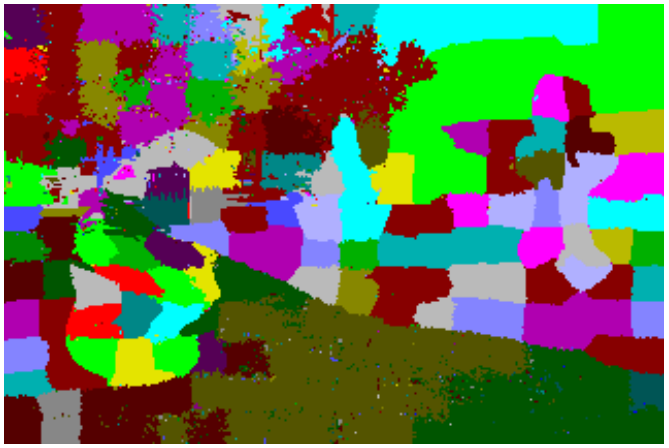


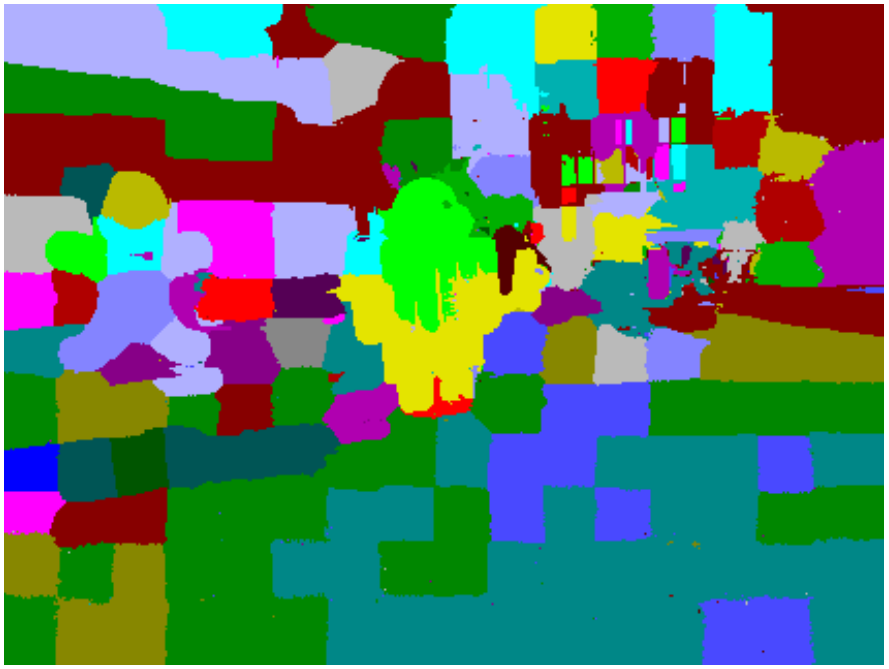



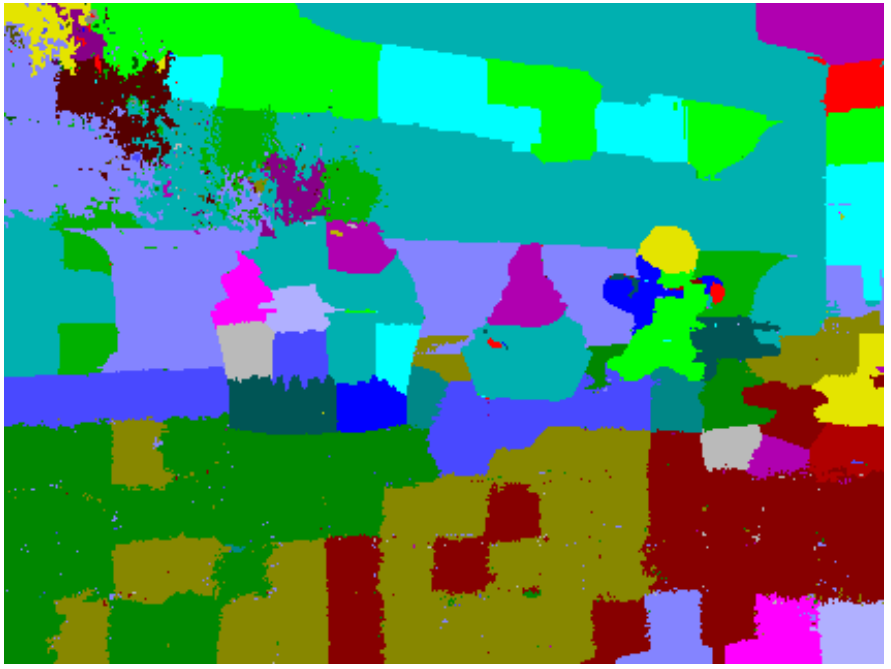




CIE XY Theta Clustering of the super pixels









Partial Shape Matching of the gingerbread man

extract ordered borders of shapes, blur them by one or two sigma,
then ordered bipartite matching algorithm.