

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

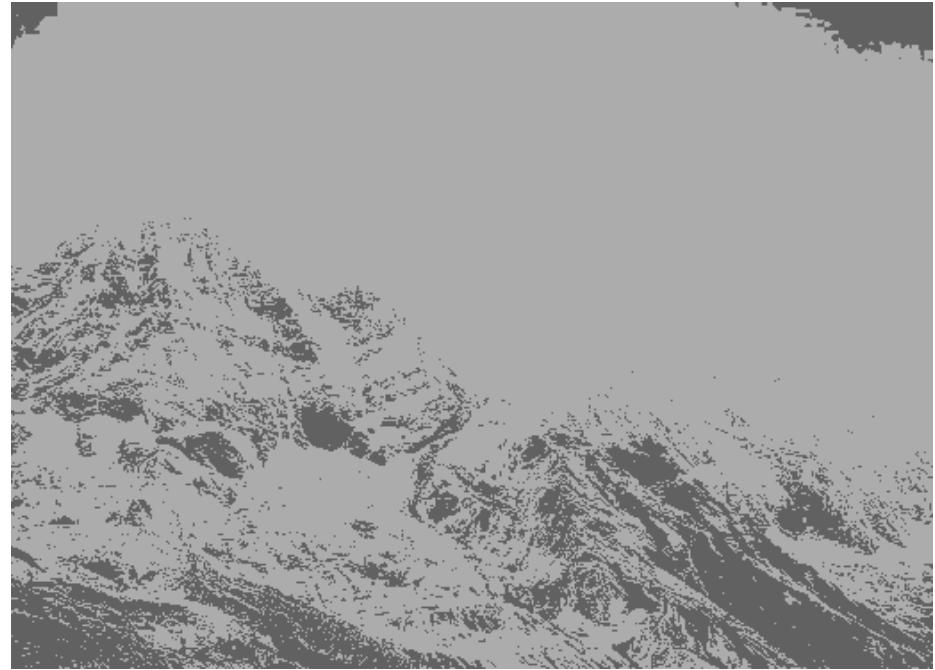
for Venturi, best was PolarCIEXYAndFrequency

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);
```



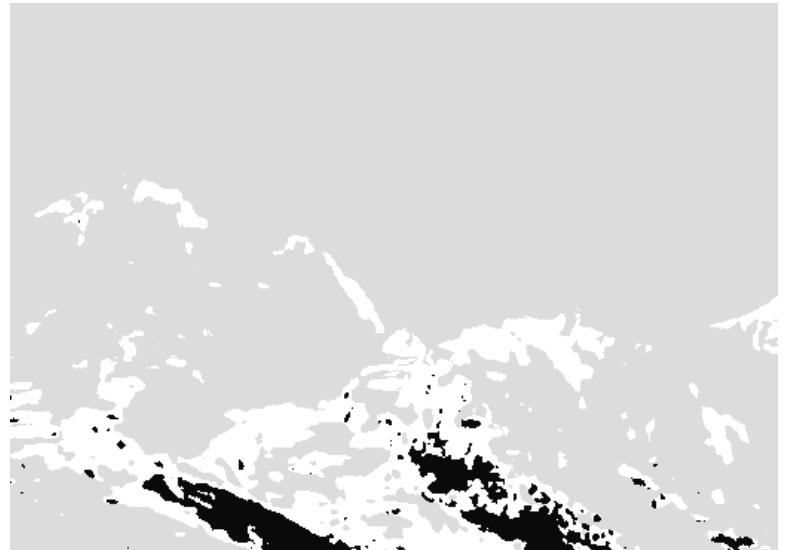
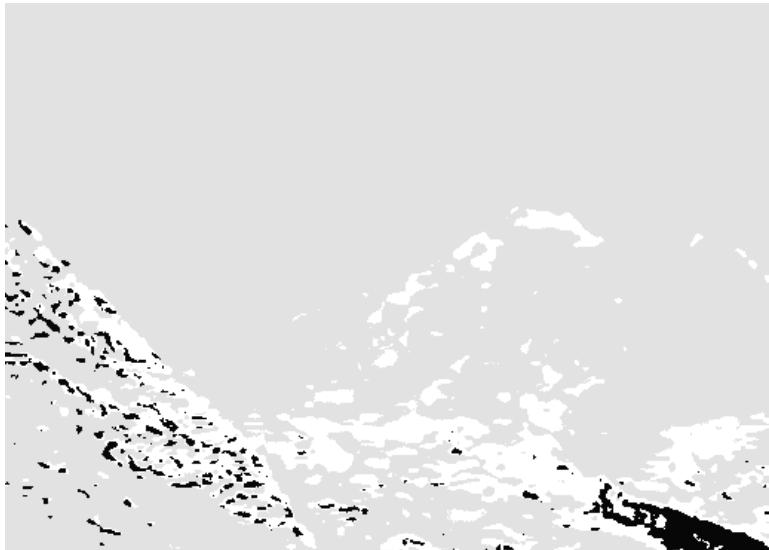
```
int kBands = 3; imageSegmentation.applyUsingKMPP(gsImg1, kBands);
```



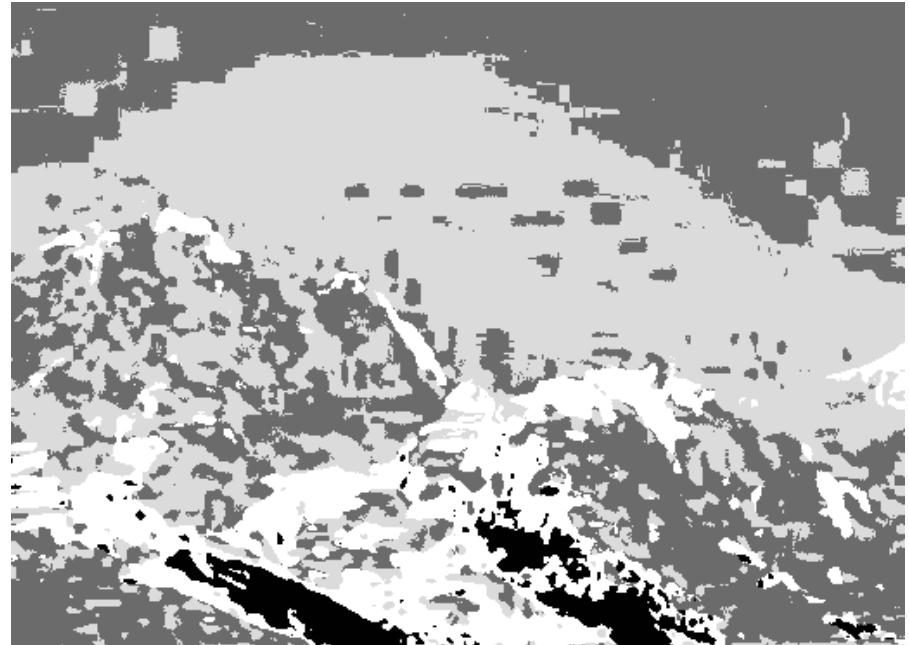
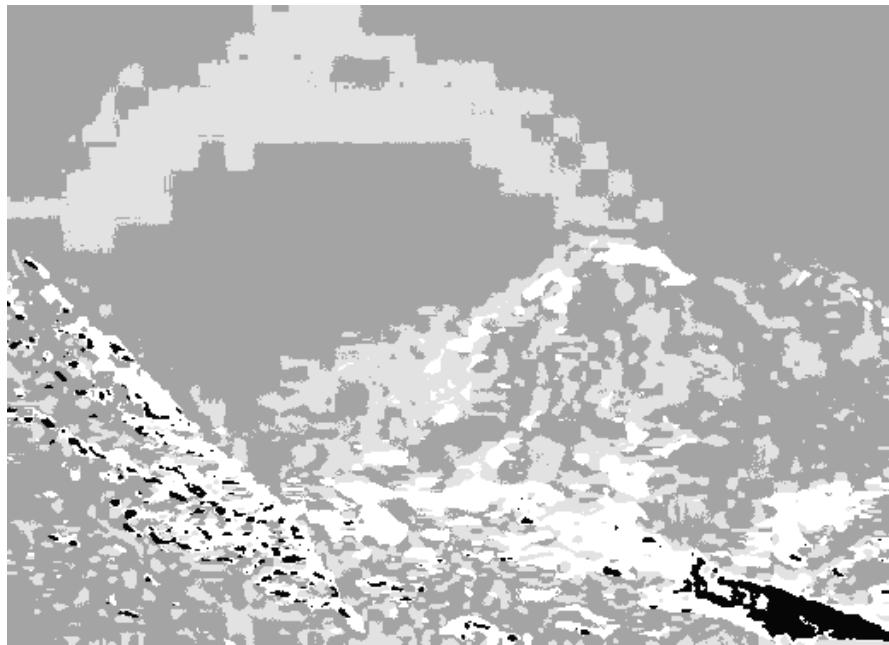
```
int kBands = 8; imageSegmentation.applyUsingKMPP(gslImg1, kBands);
```



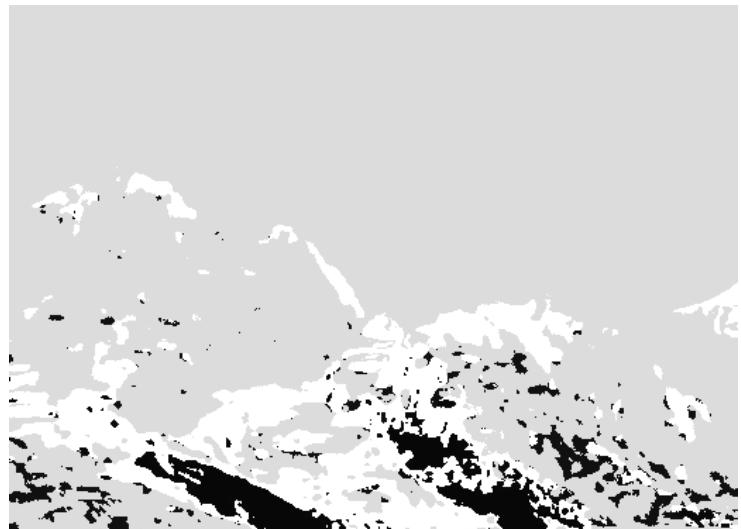
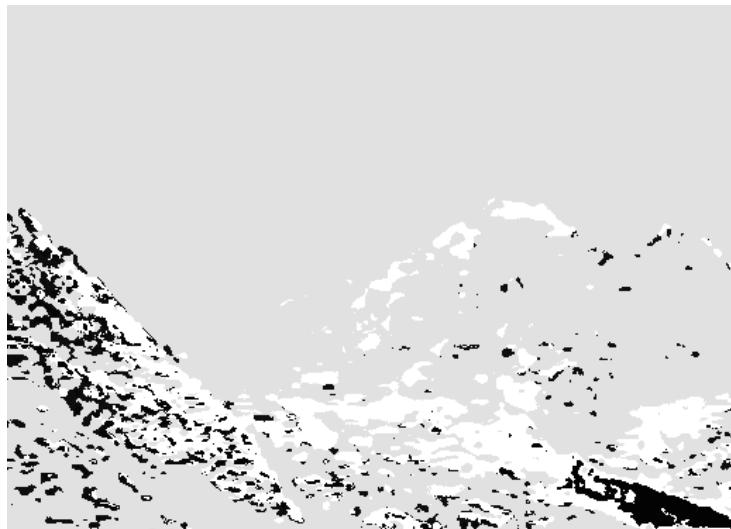
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gslImg1, kBands);
```



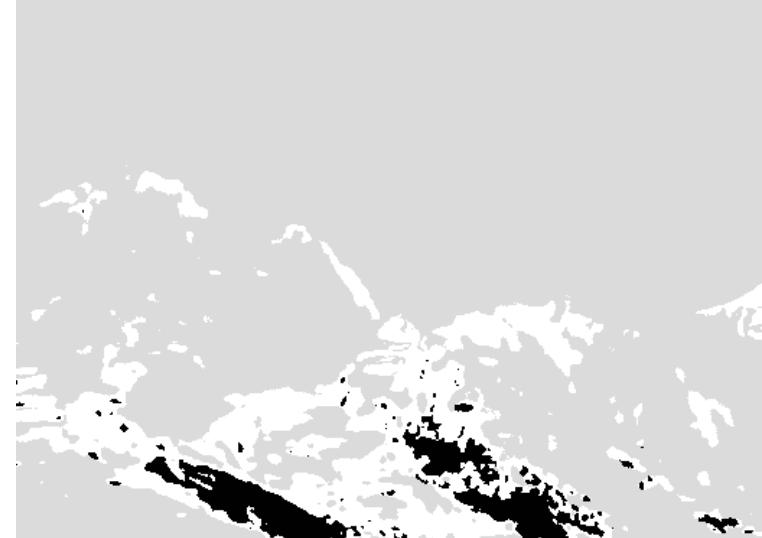
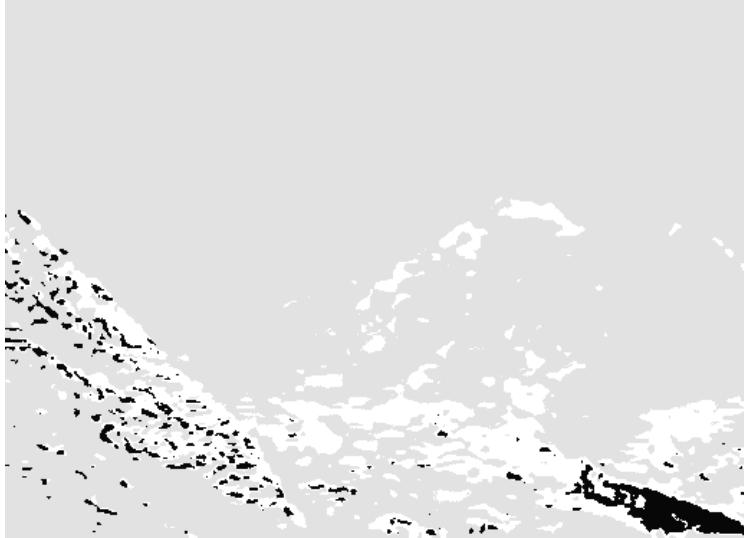
```
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);
```



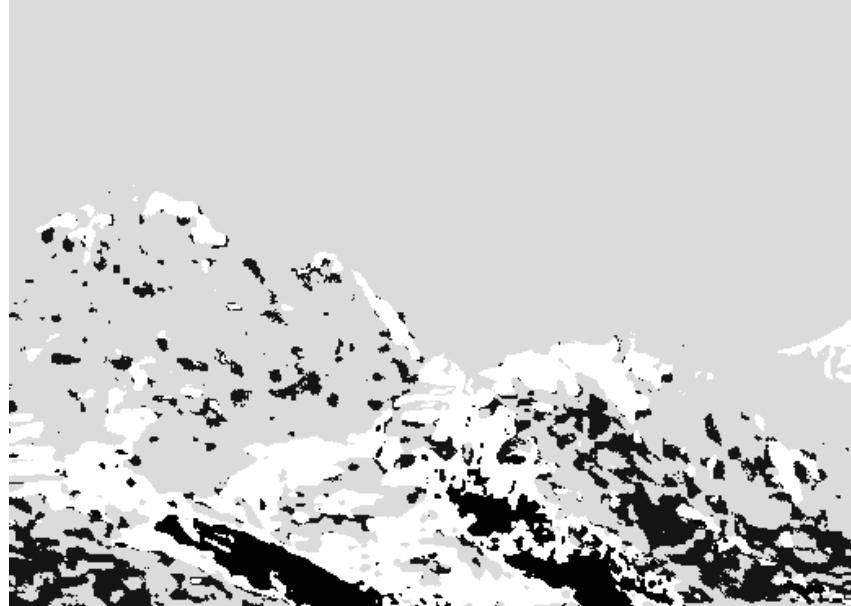
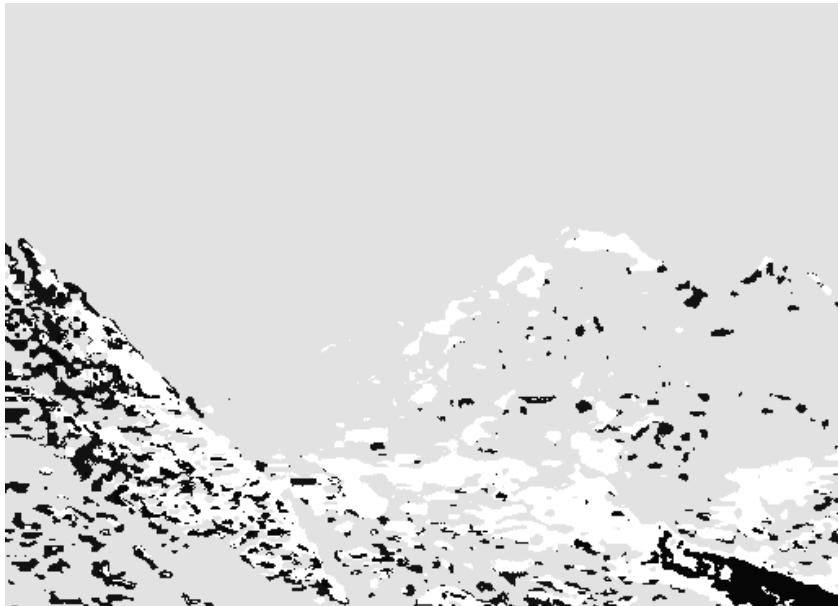
```
int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);
```



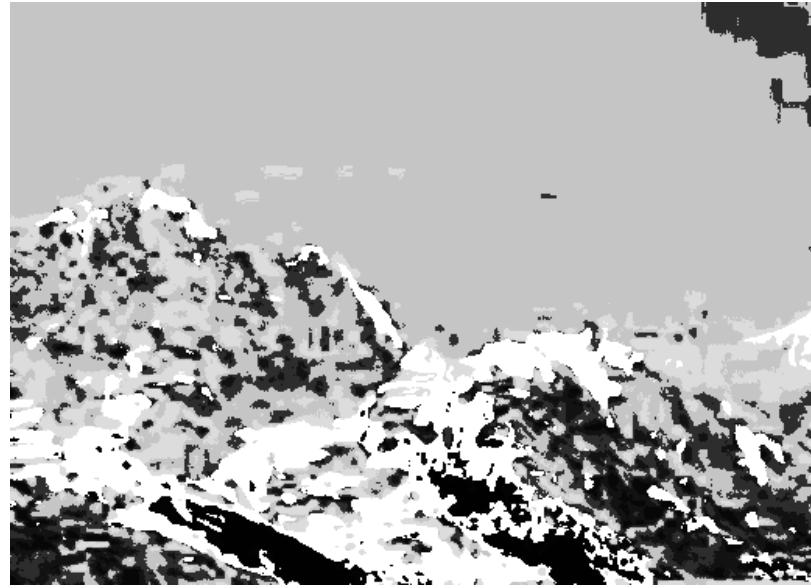
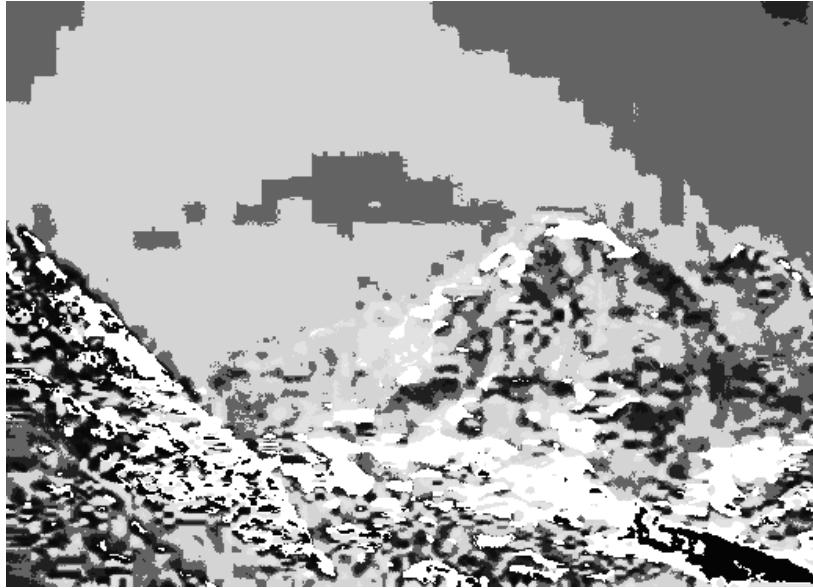
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1,  
kBands);
```



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

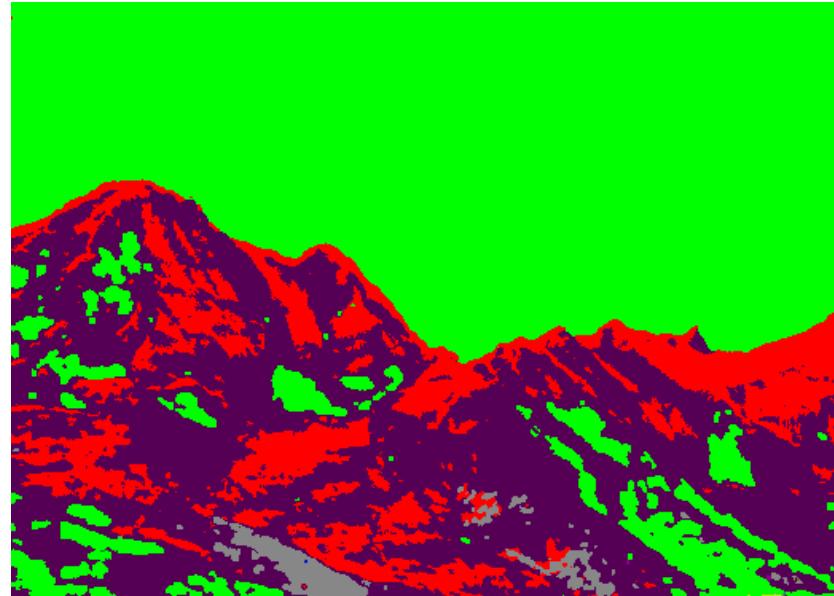
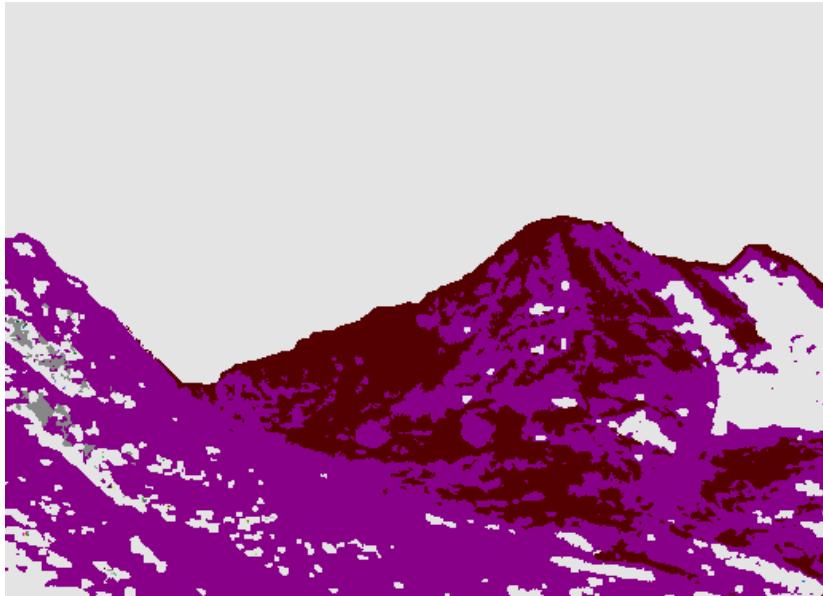


```
int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1,  
kBands);
```

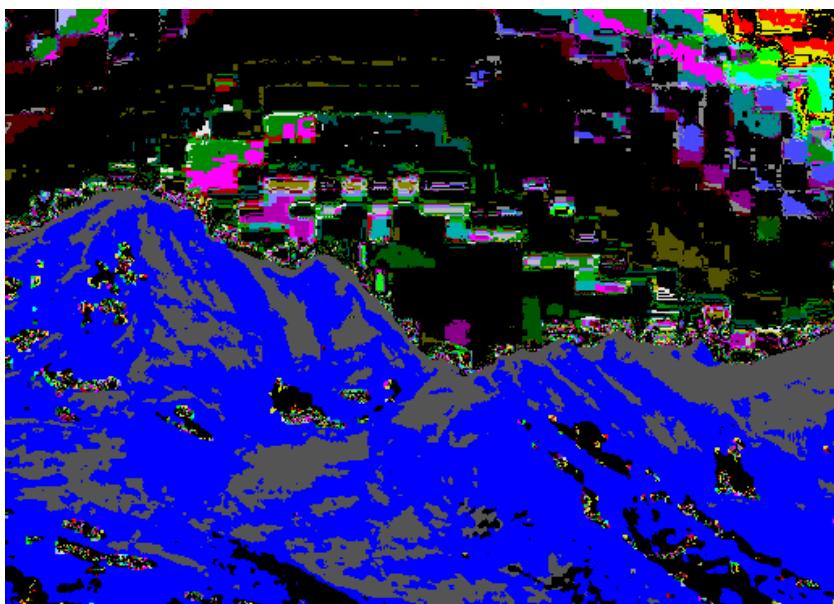
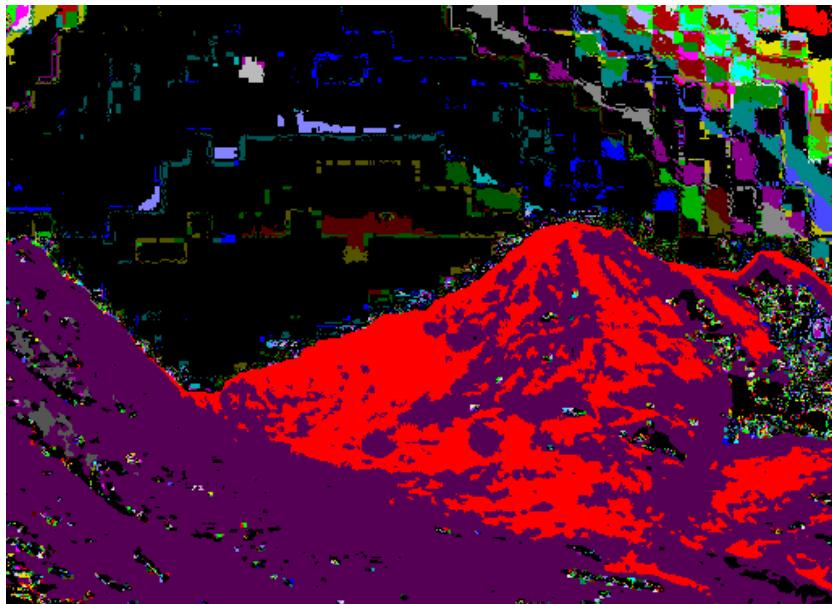


```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistogram(gsImg1, kBands);
```

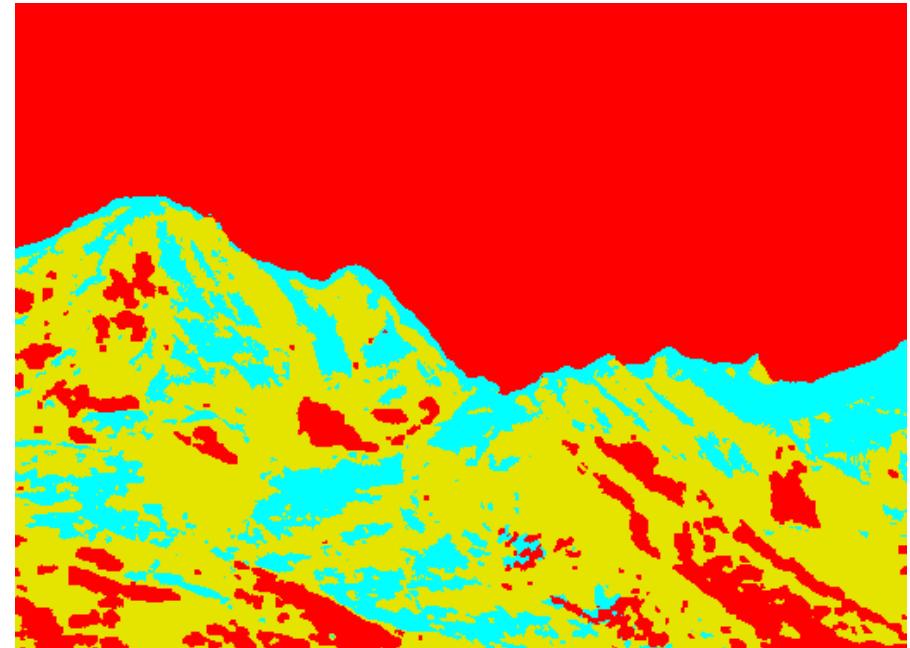
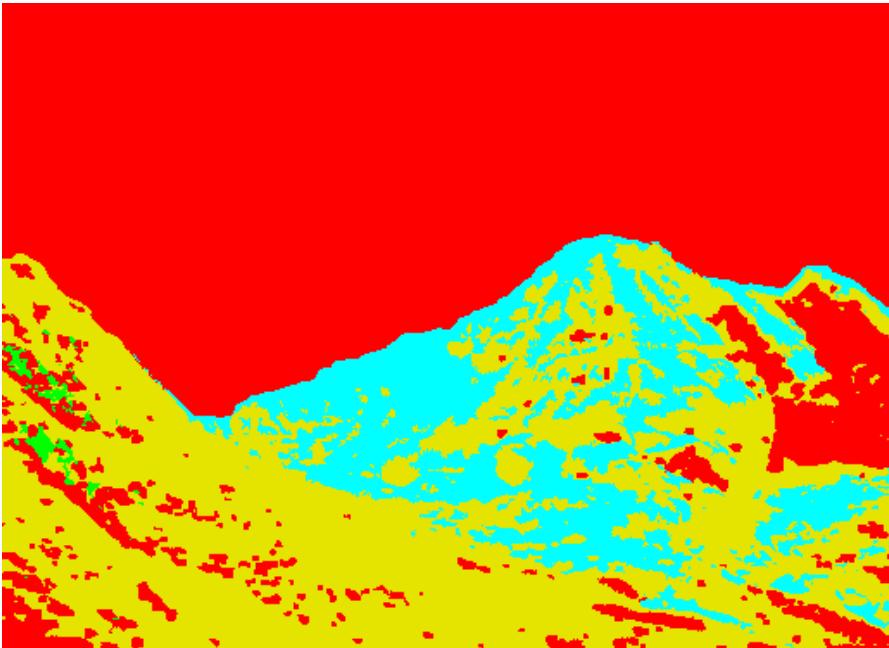
```
imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);
```



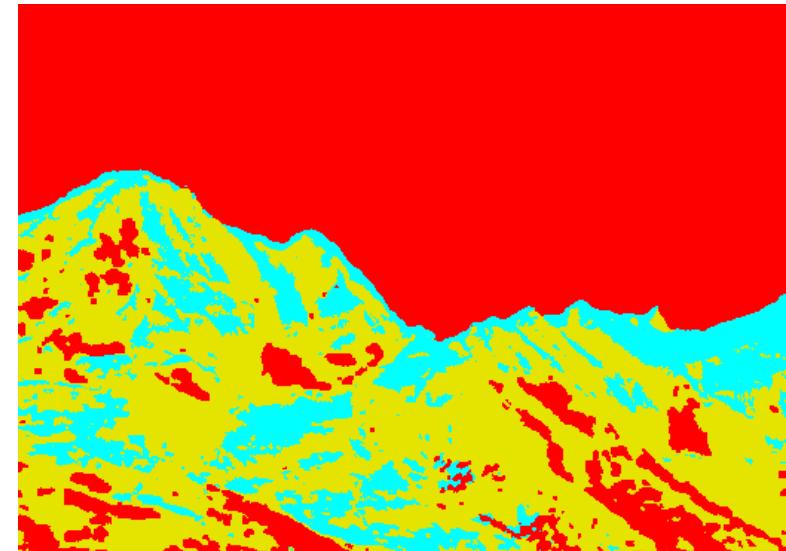
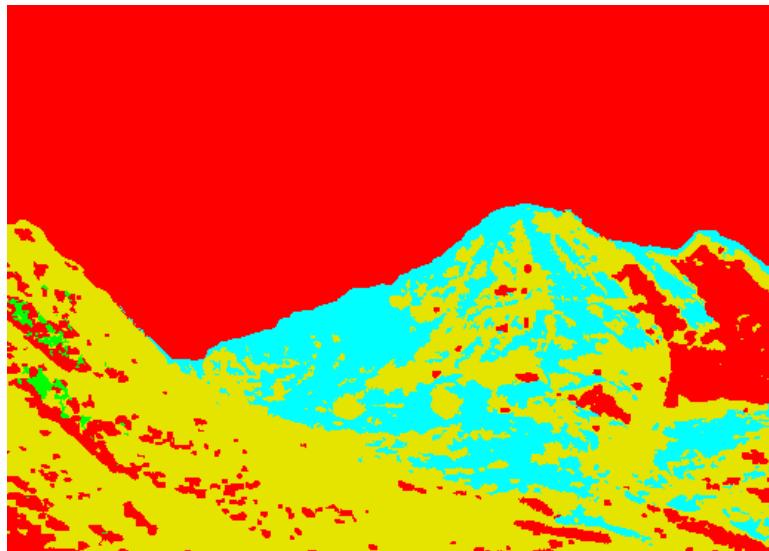
```
imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);
```



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



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



```
int kBands = 2; imageSegmentation.applyUsingKMPP(gslImg1, kBands);
```



```
int kBands = 3; imageSegmentation.applyUsingKMPP(gslImg1, kBands);
```



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



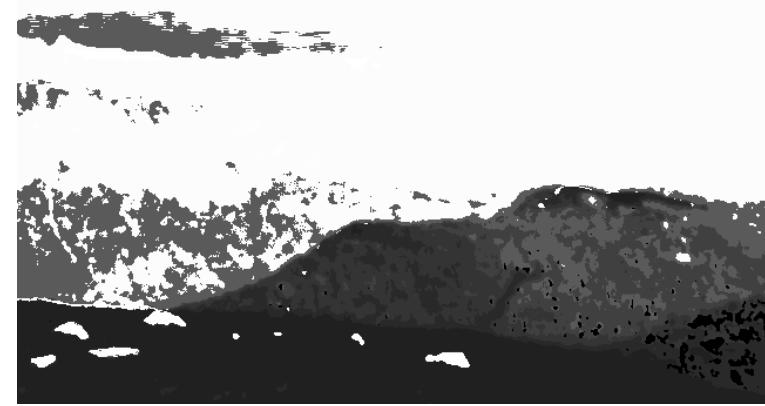
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);
```



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



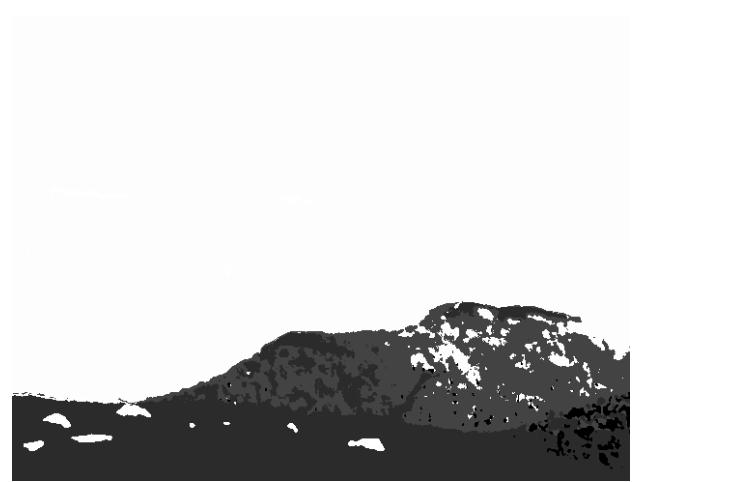
```
int kBands = 8; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gsImg1, kBands);
```



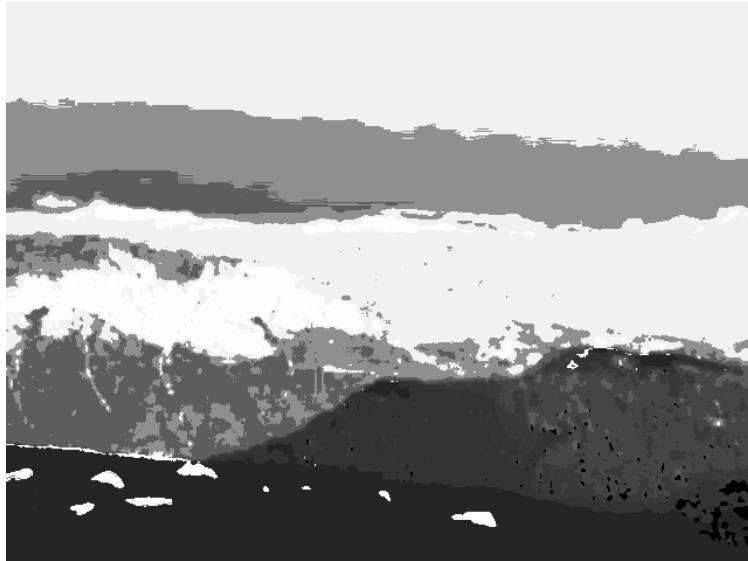
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gsImg1,  
kBands);
```



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



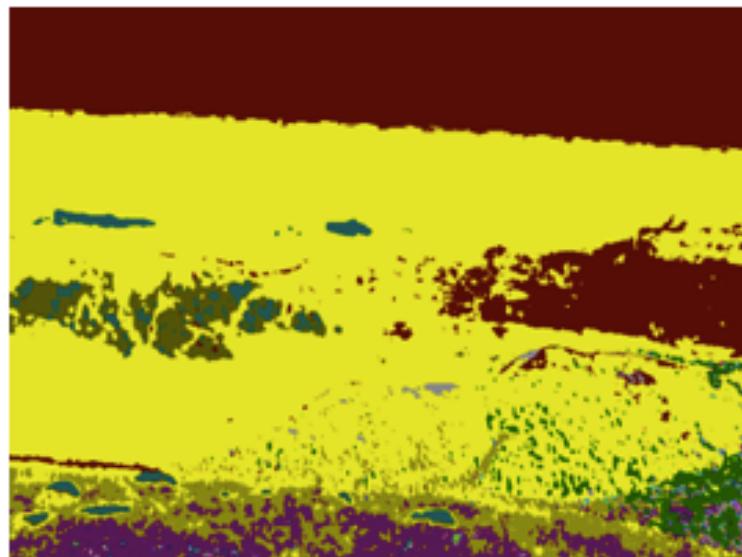
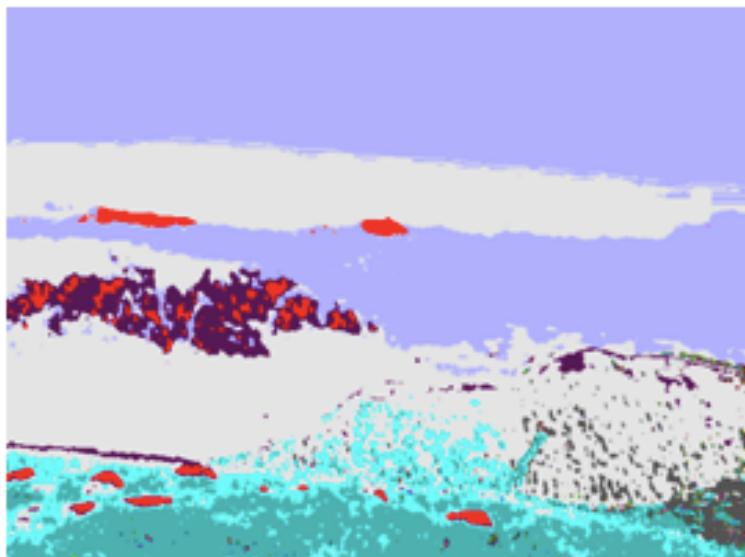
```
int kBands =8; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq
```



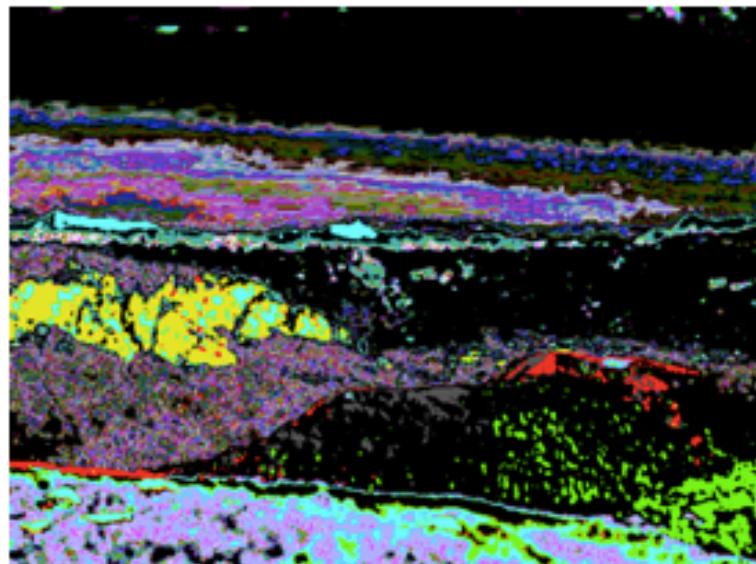
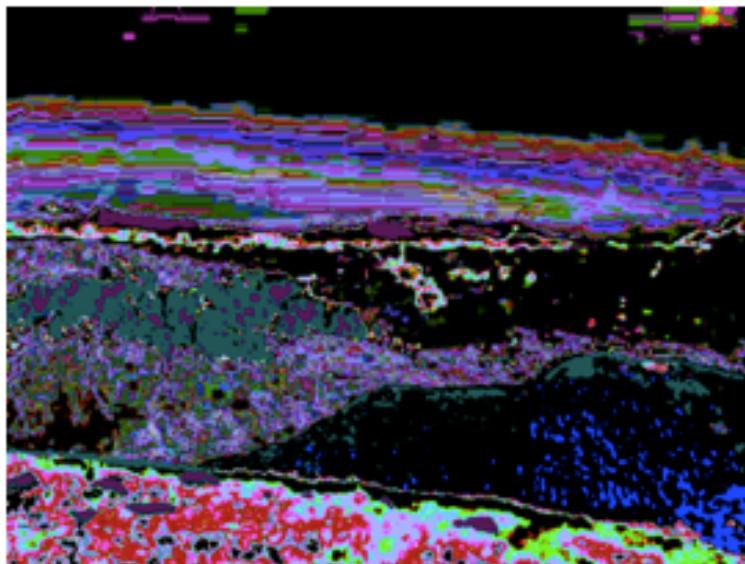
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistogram(gsImg1, kBands);
```



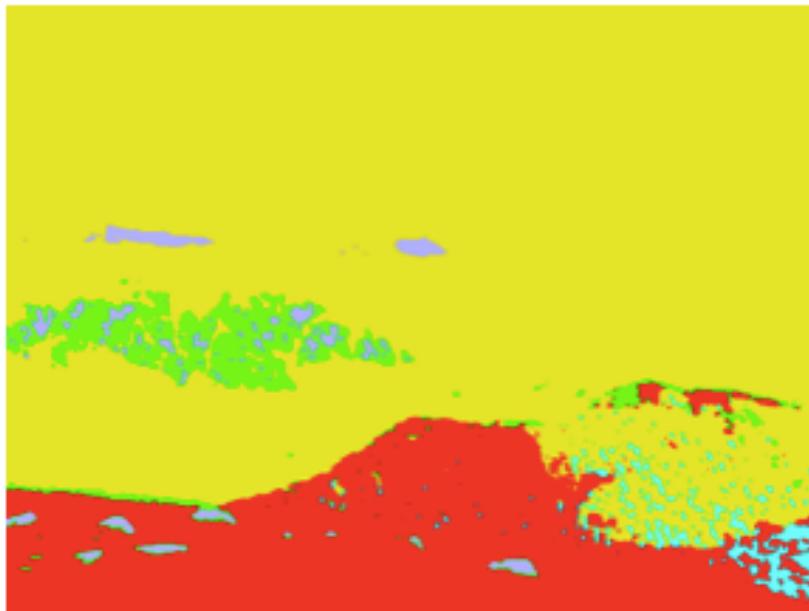
```
imageSegmentation.calculateUsingCIEXYAndClustering(gslImg1, true);
```



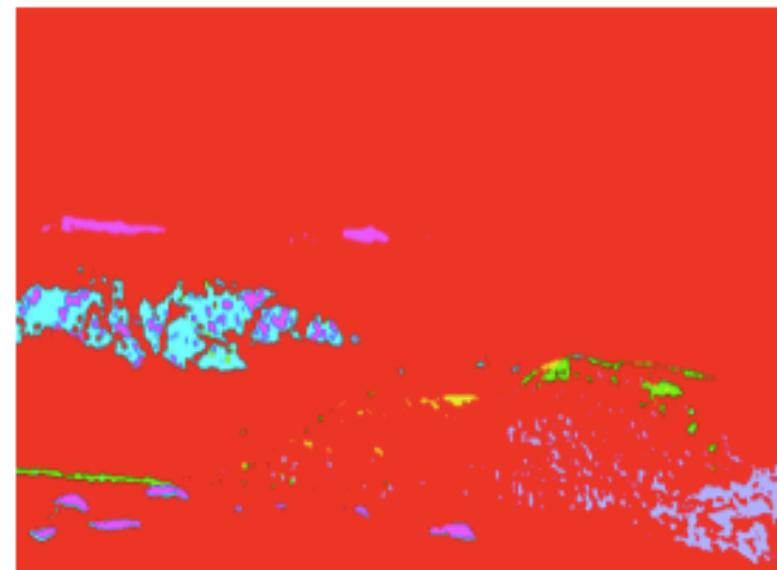
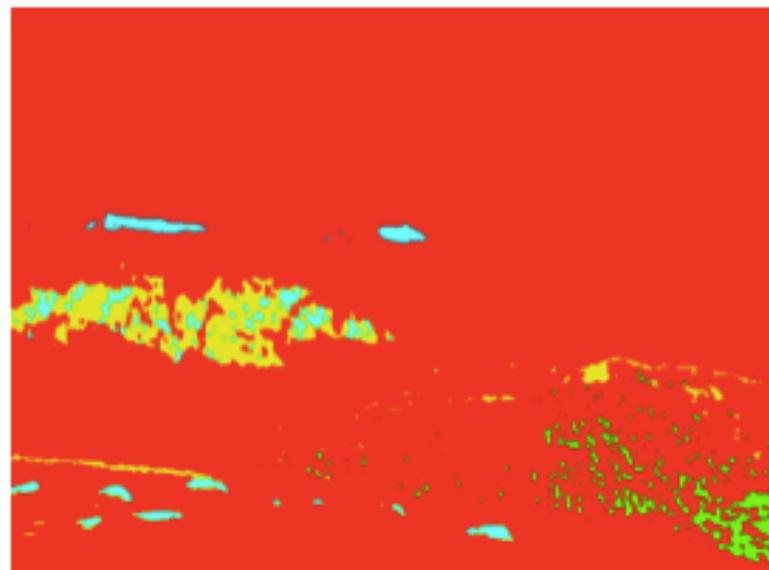
```
imageSegmentation.calculateUsingPolarCIEXYAndClustering(gslImg1, true);
```



```
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gslImg1, true);
```



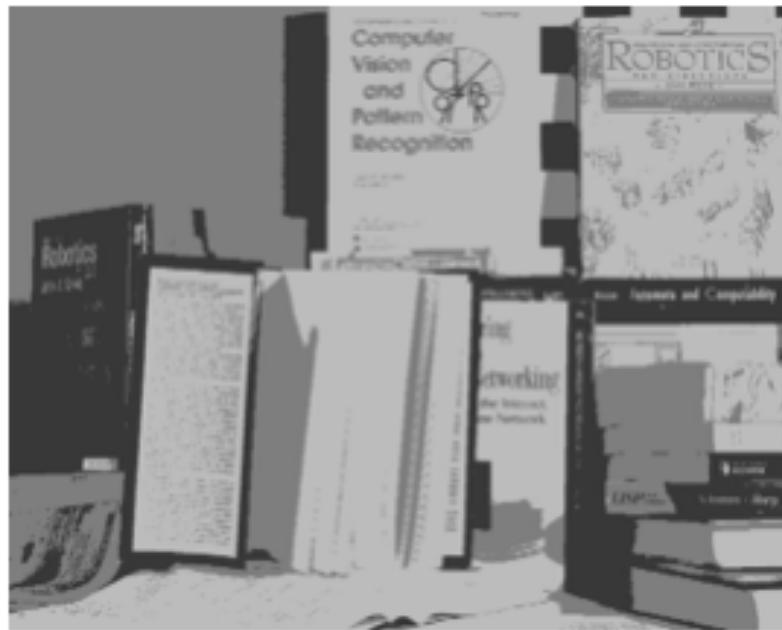
```
imageSegmentation.calculateUsingPolarCIEXYAndFrequency(gslImg1, 0.2f, true);
```



```
int kBands = 2; imageSegmentation.applyUsingKMPP(gslImg1, kBands);
```



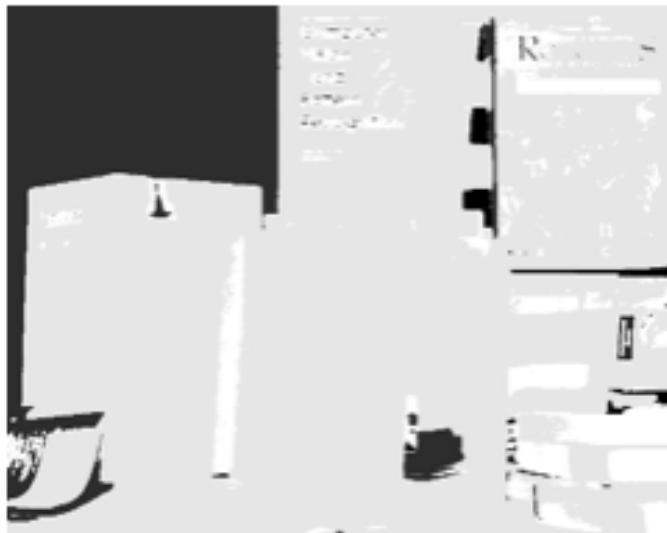
```
int kBands = 3; imageSegmentation.applyUsingKMPP(gslImg1, kBands);
```



```
int kBands = 8; imageSegmentation.applyUsingKMPP(gslmg1, kBands);
```



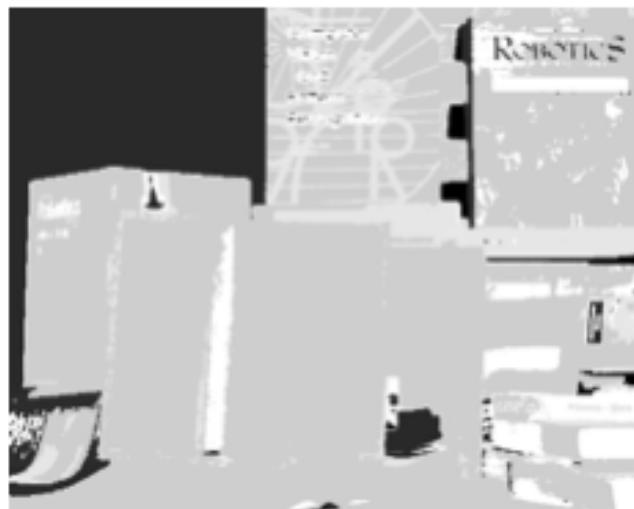
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(gslmg1, kBands);
```



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



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



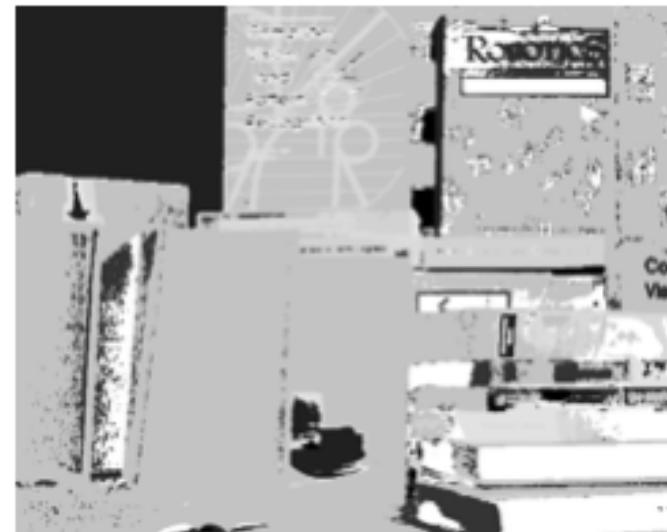
```
int kBands = 2; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gslImg1,  
kBands);
```



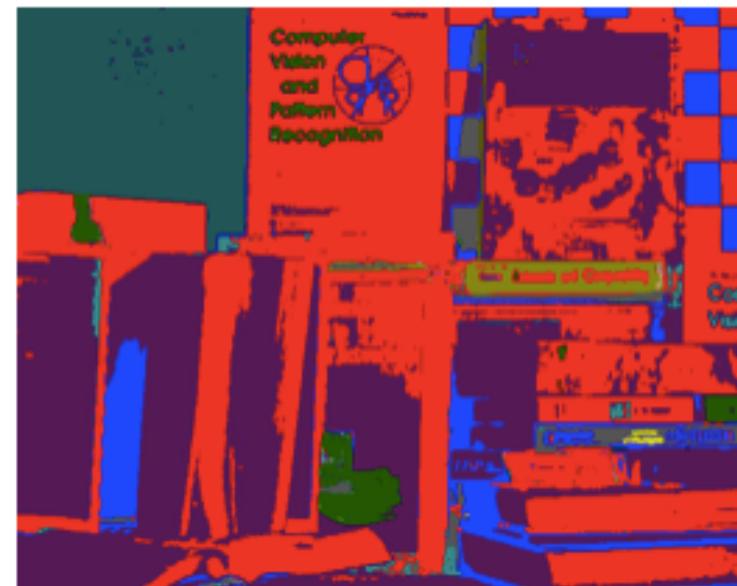
```
int kBands = 3; imageSegmentation.applyUsingCIEXYPolarThetaThenKMPPThenHistEq(gslImg1,  
kBands);
```



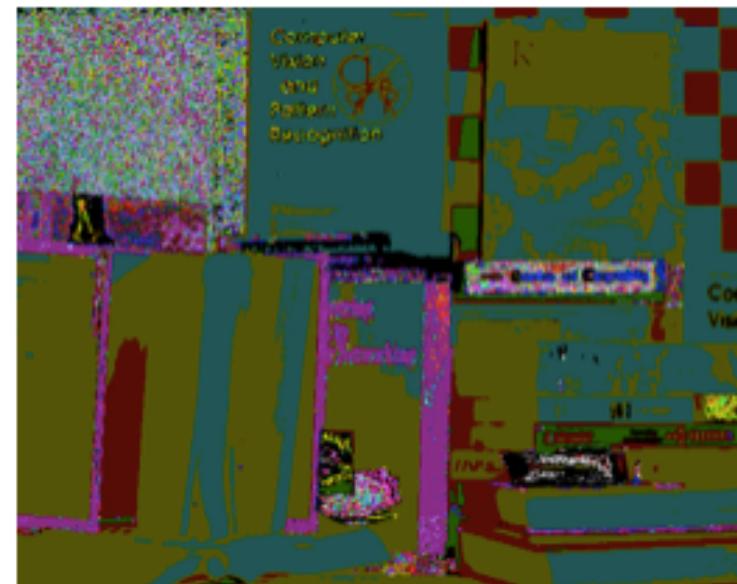
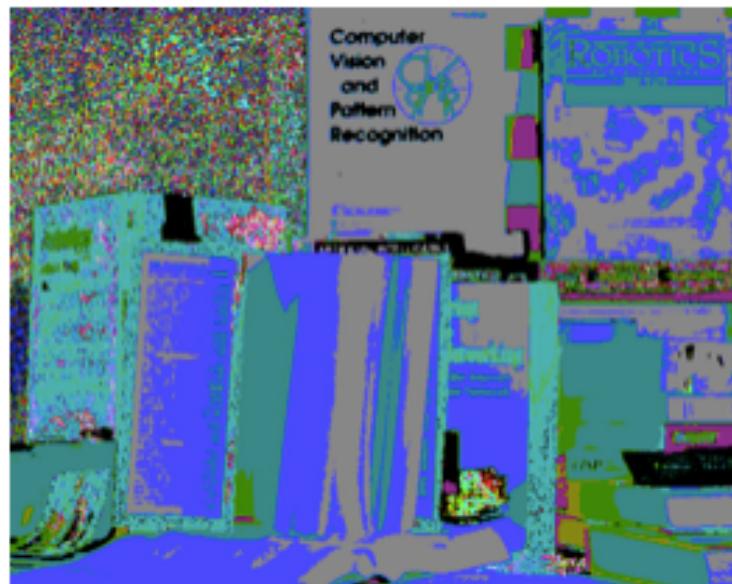
```
int kBands = 8; imageSegmentation.applyUsingCIEXYZPolarThetaThenKMPPThenHistEq(gsImg1,  
kBands);
```



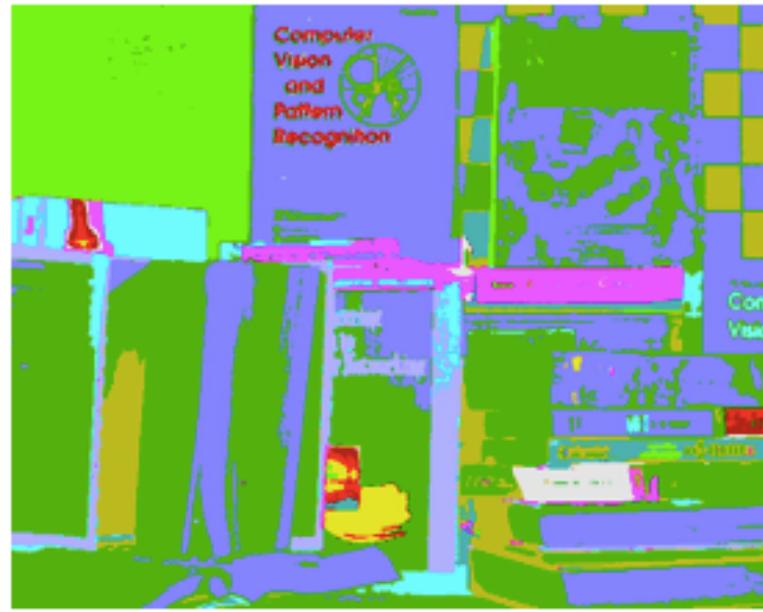
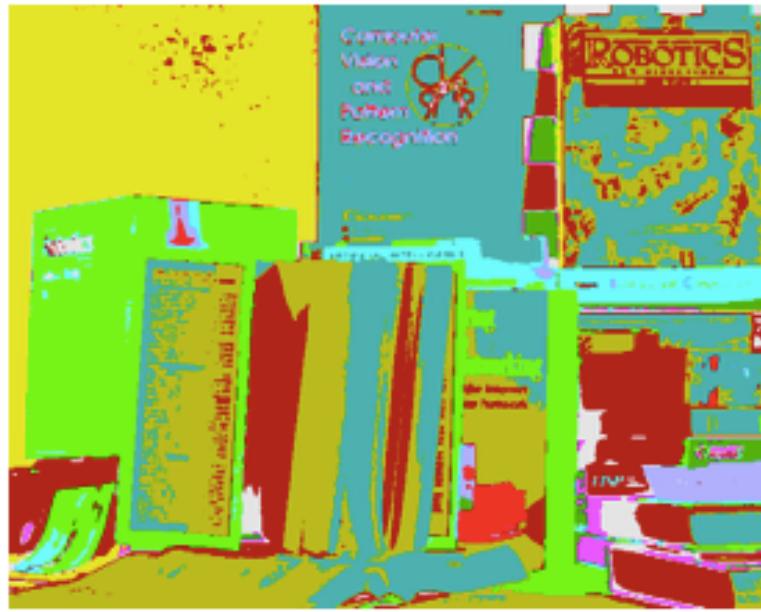
```
imageSegmentation.calculateUsingCIEXYAndClustering(gsImg1, true);
```



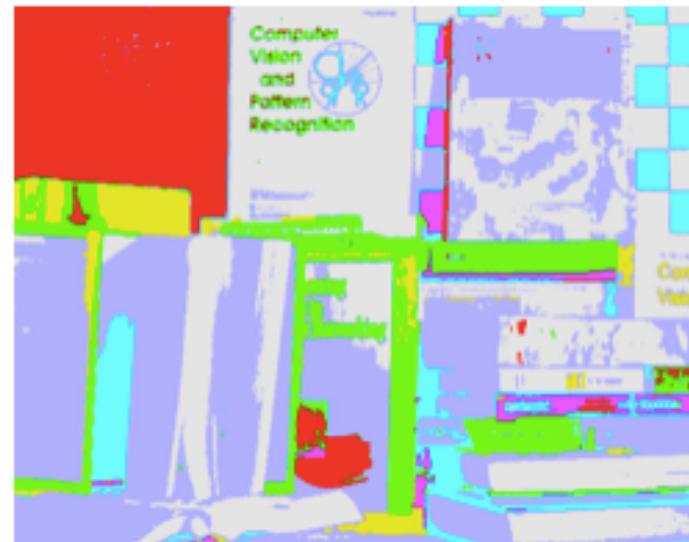
```
imageSegmentation.calculateUsingPolarCIEXYAndClustering(gsImg1, true);
```



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



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



```
int kBands = 2; imageSegmentation.applyUsingKMPP(gslImg1, kBands);  
zoom in
```

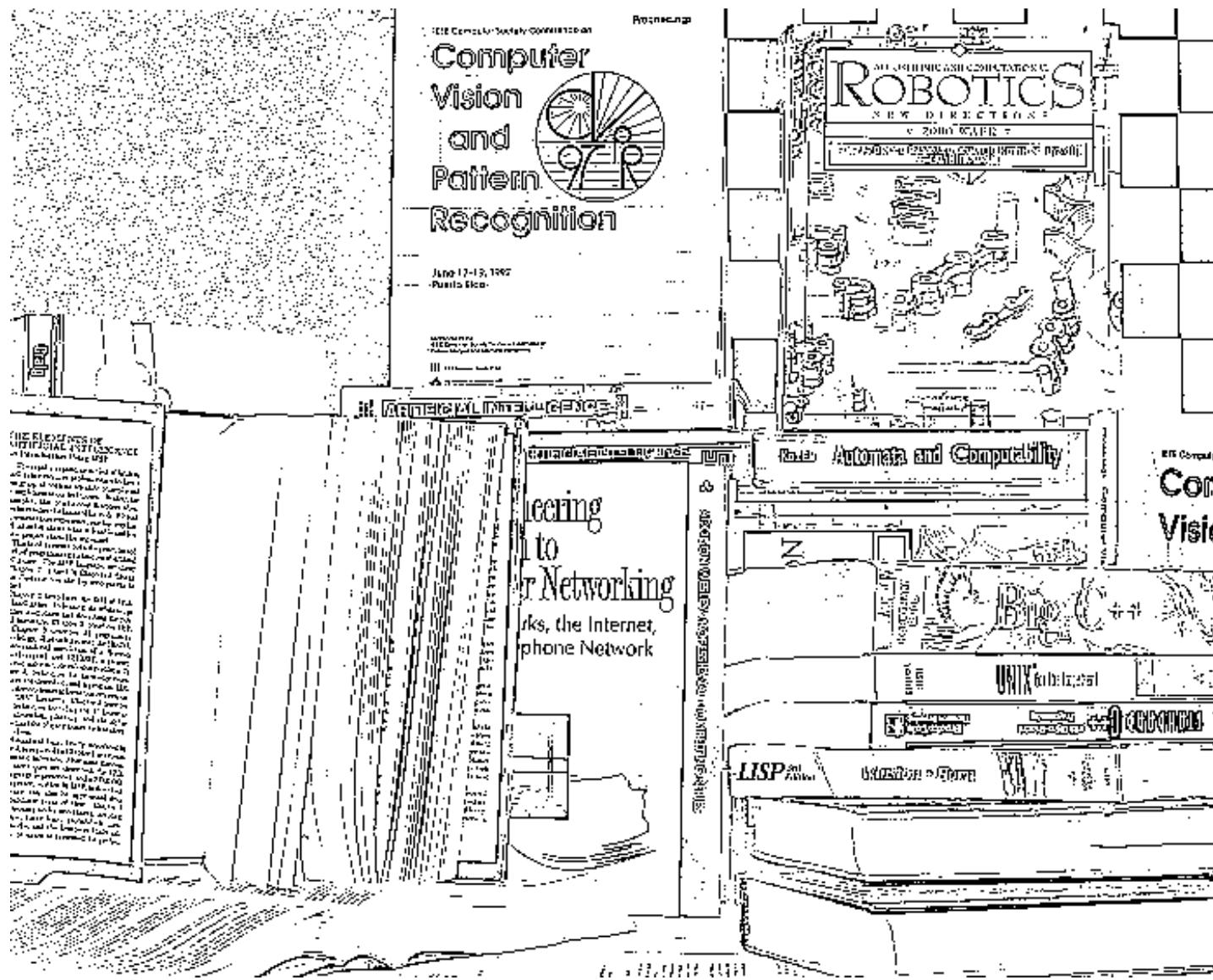
Computer
Vision
and
Pattern
Recognition



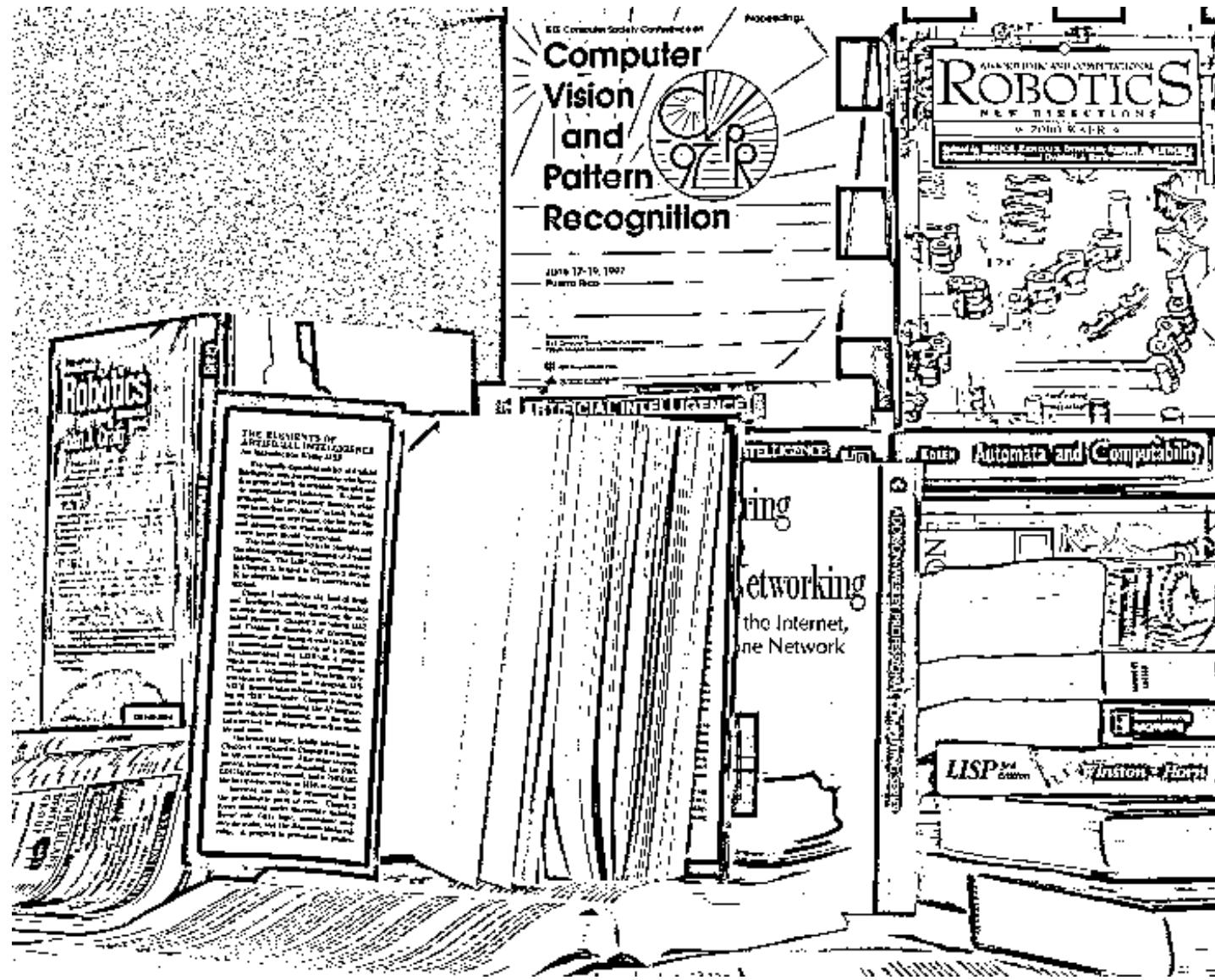
Computer
Vision
and
Pattern
Recognition



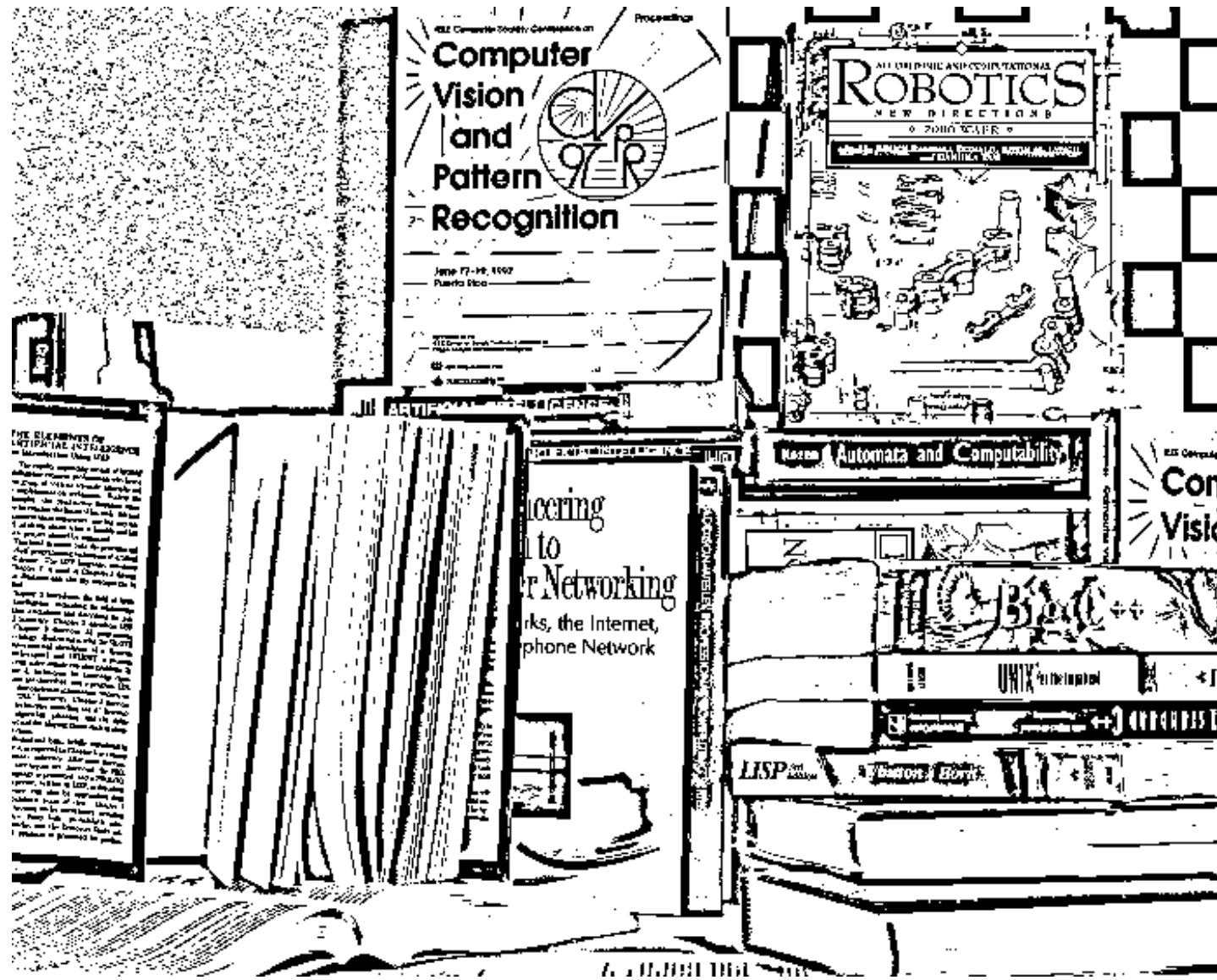
adaptive means, h=1



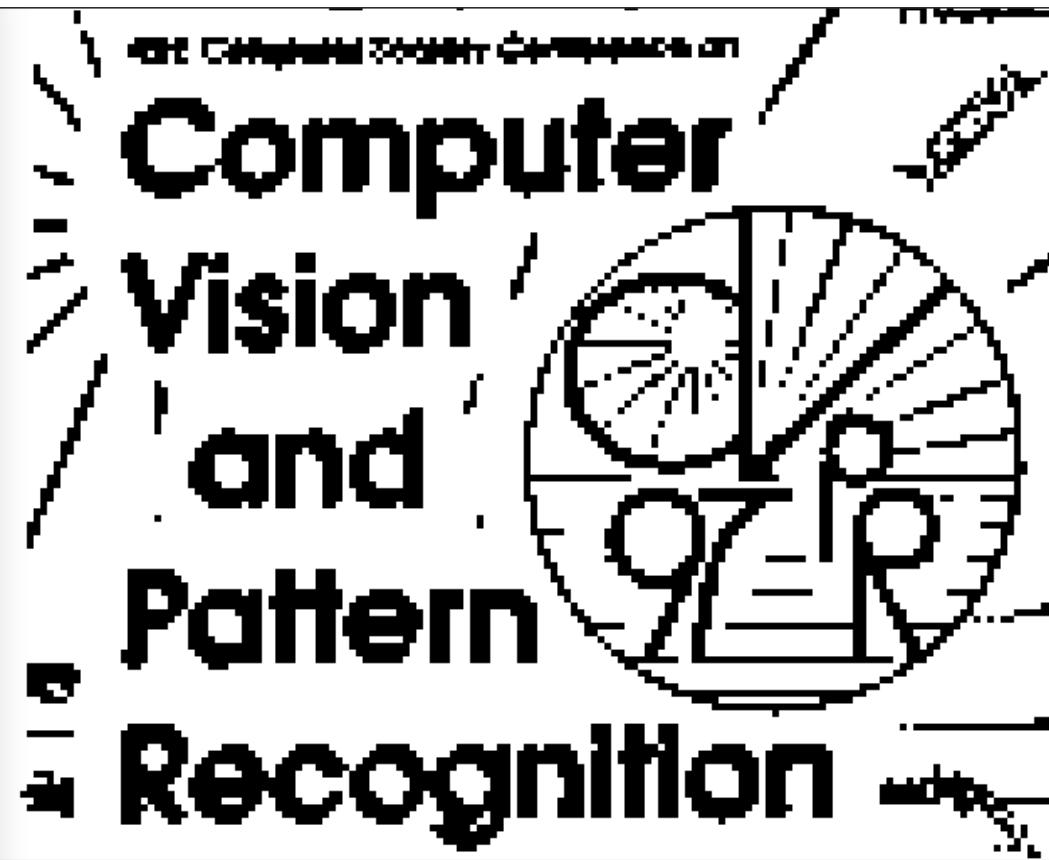
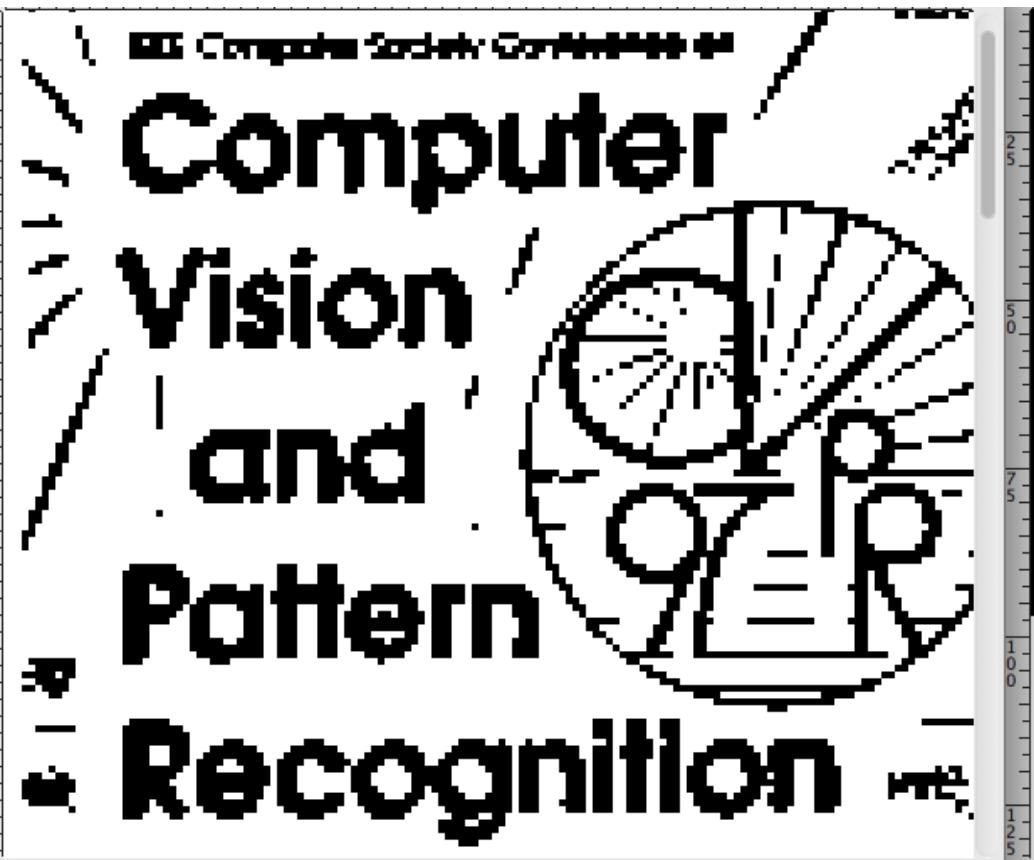
adaptive means, h=3



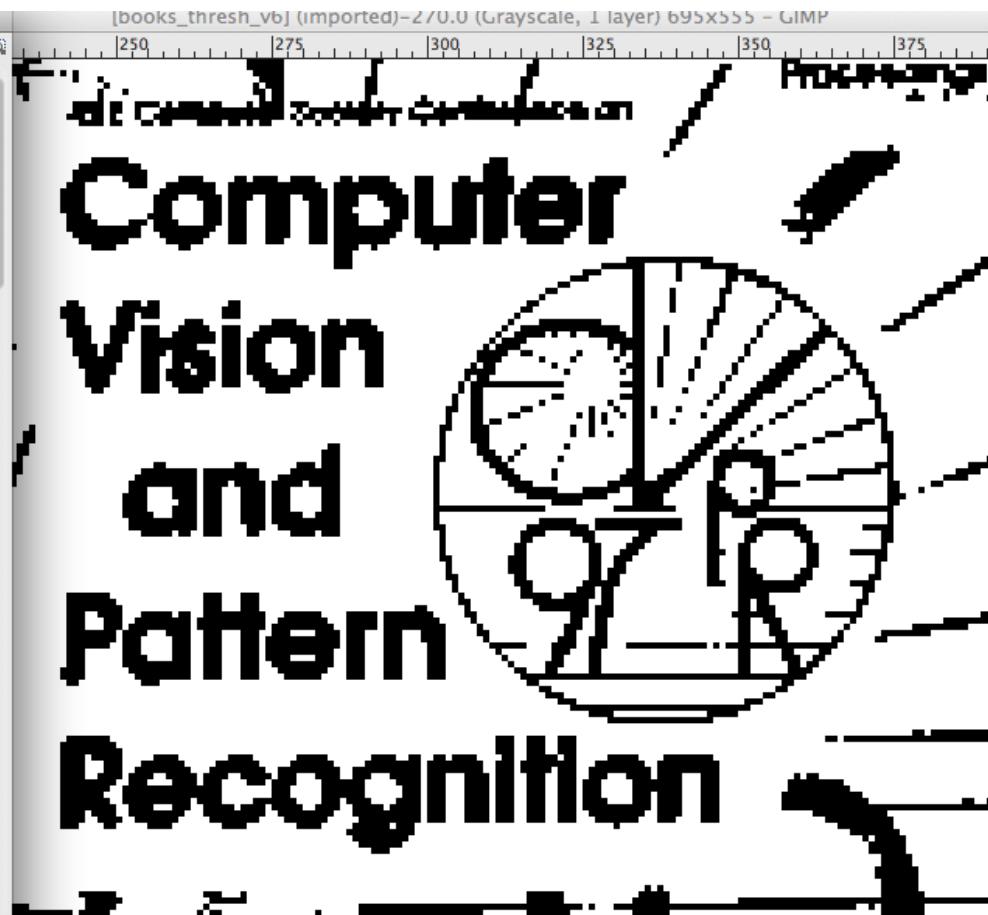
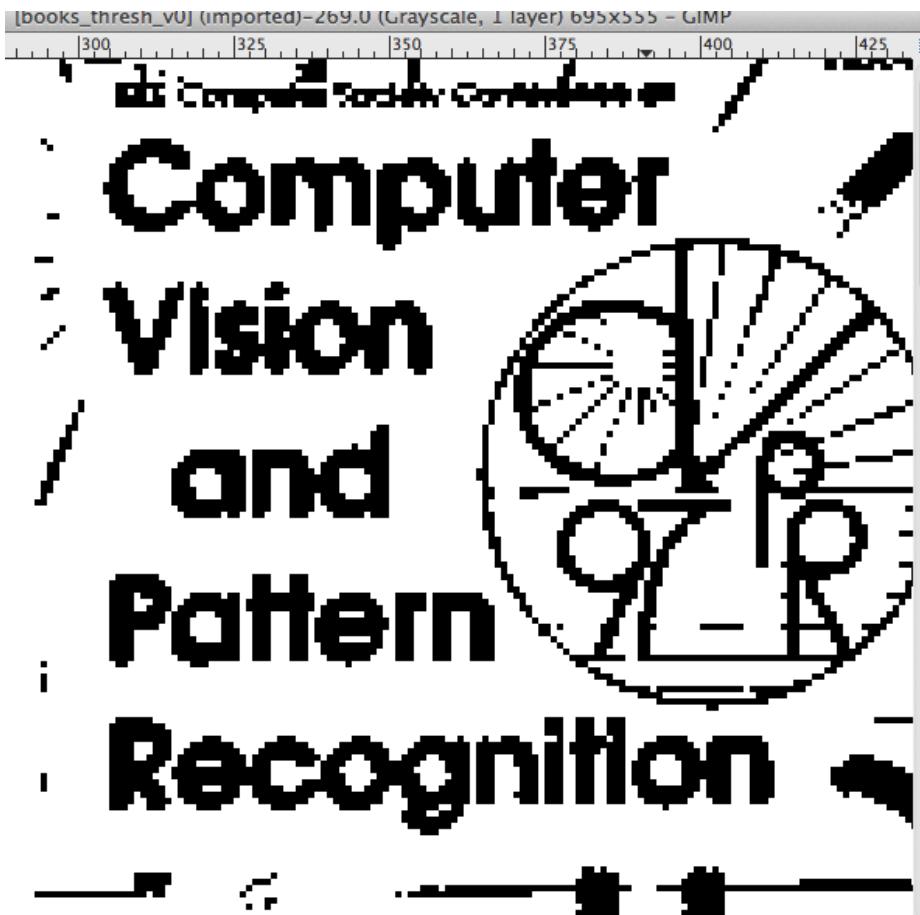
adaptive means, h=5



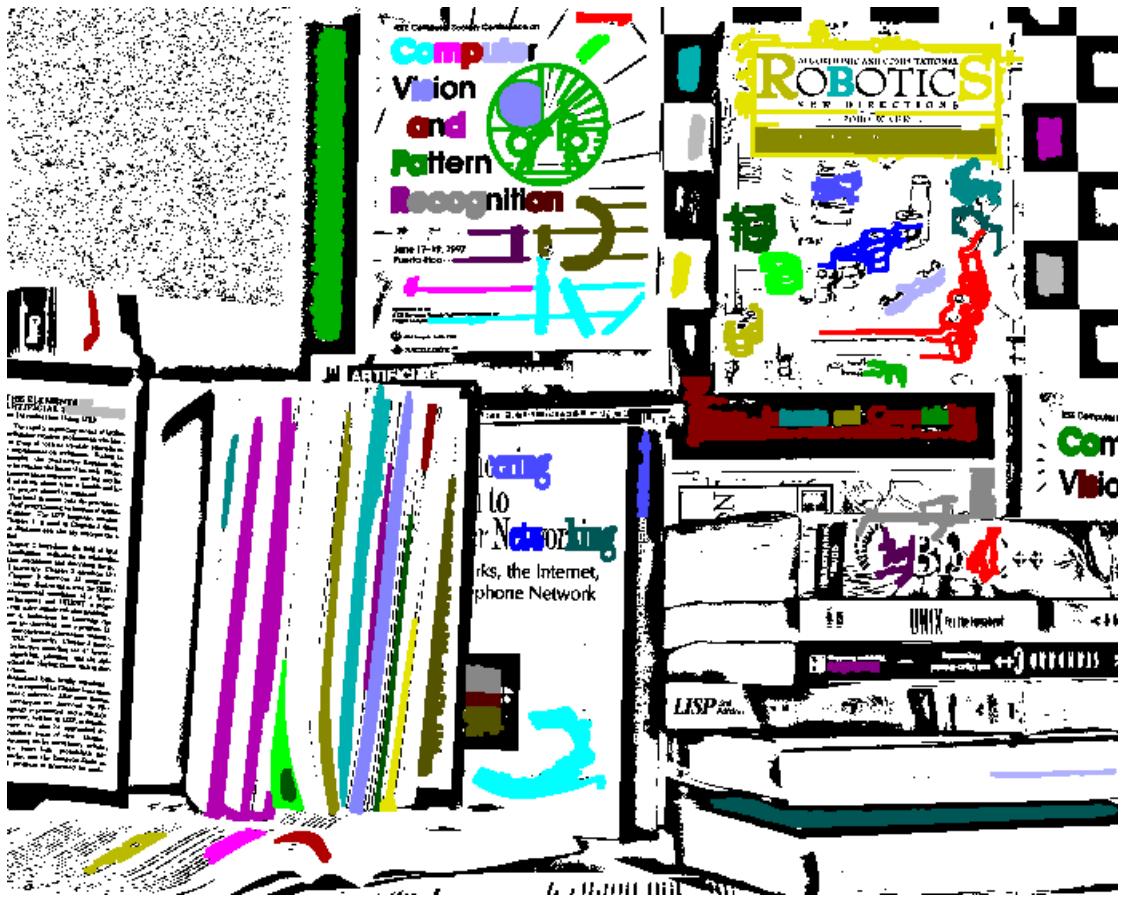
adaptive means, $h=7$



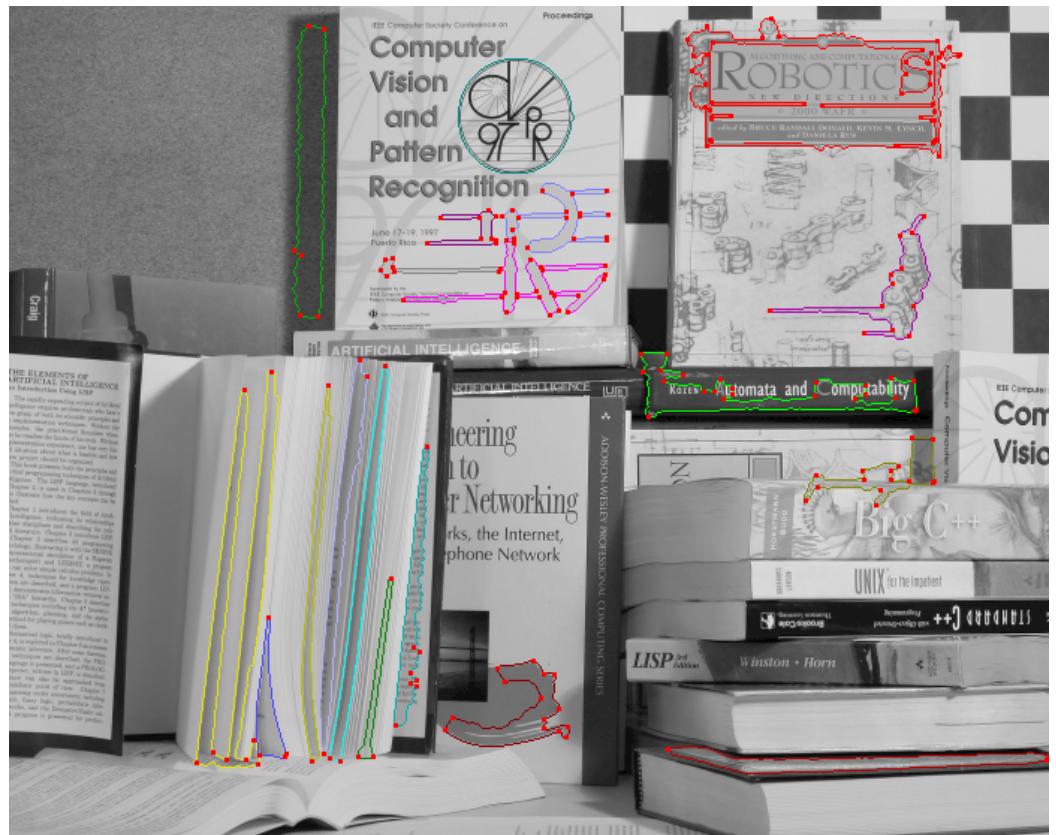
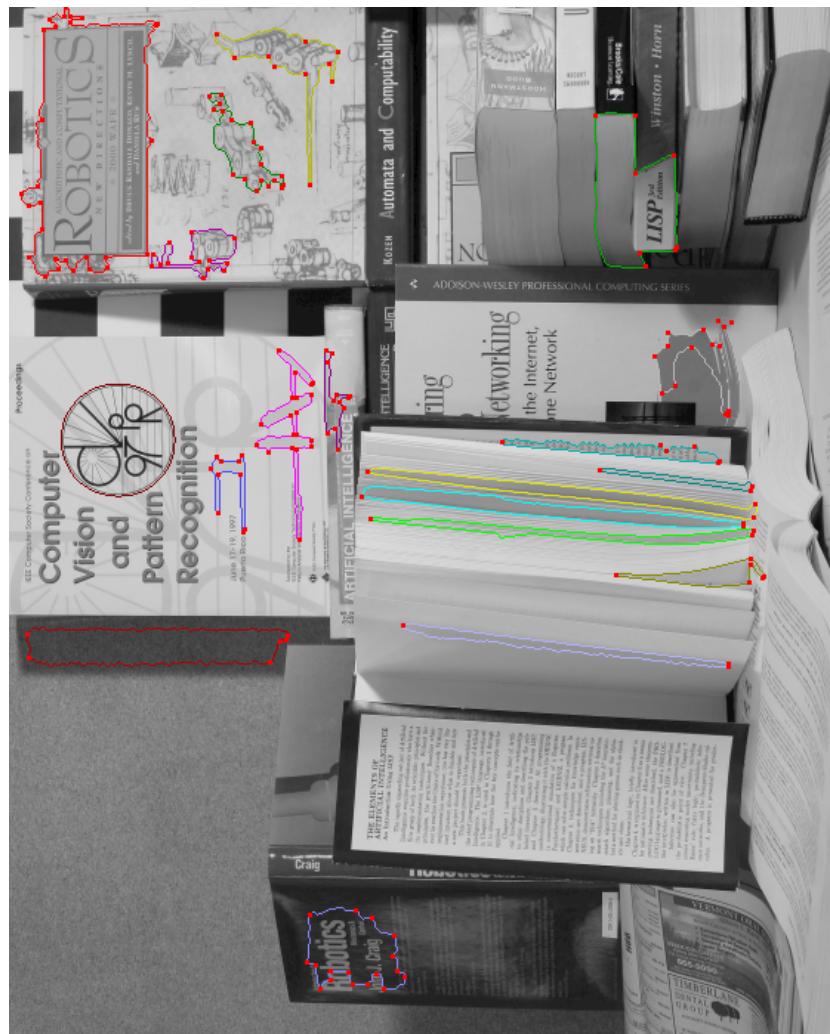
adaptive means, h=11 for adaptive to join close letters



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

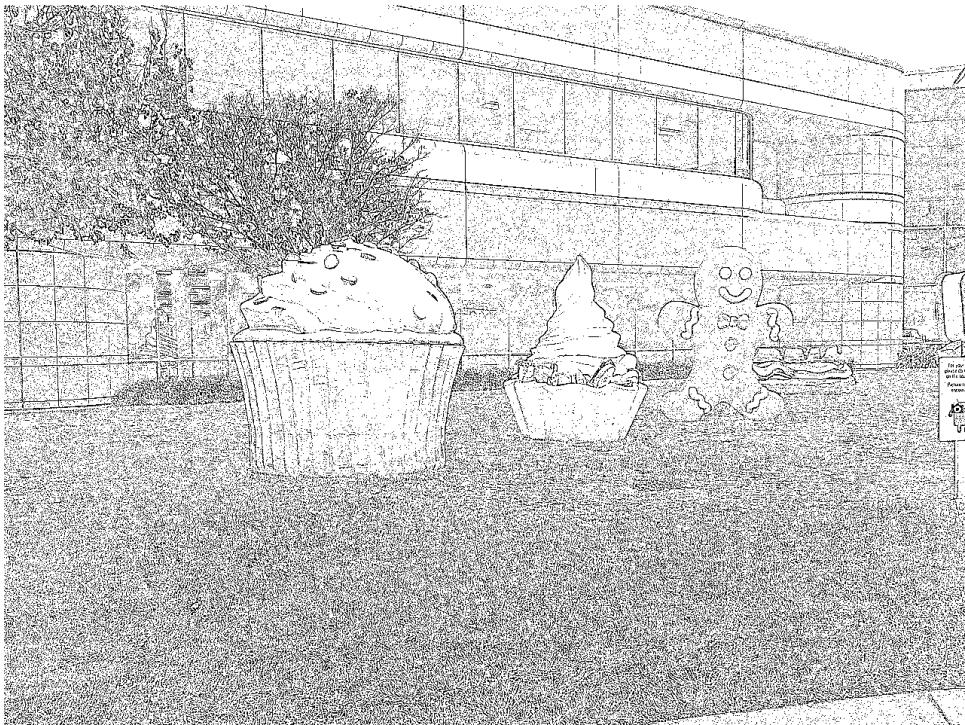


adaptive means, h=11 blob corners

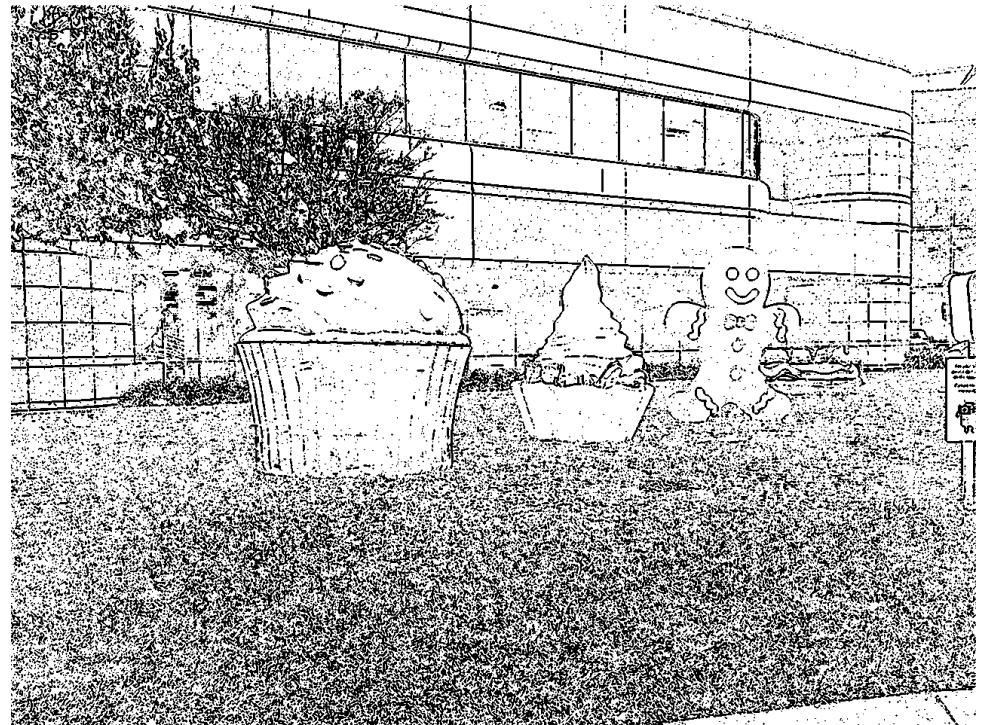


the letters are filtered out due to large nPerimeter compared to nInternal
adaptive means, h=3 was better for letters as blobs

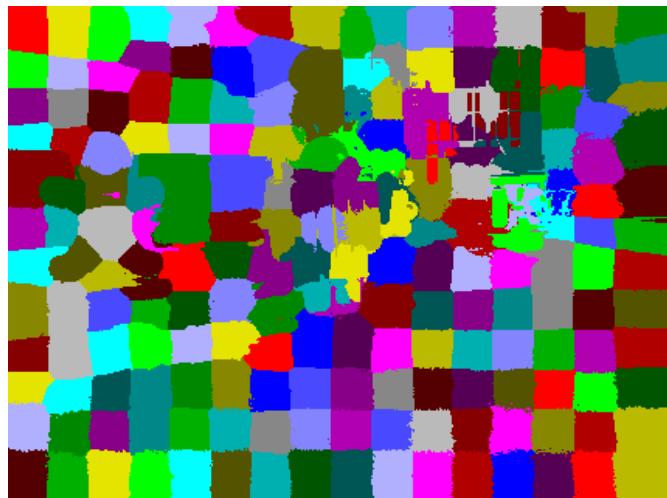
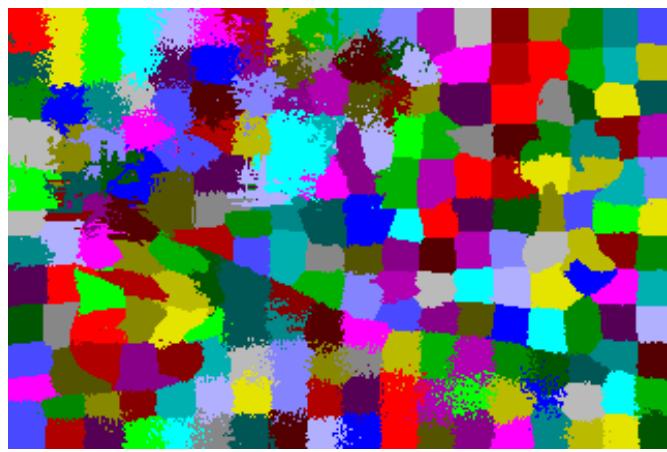
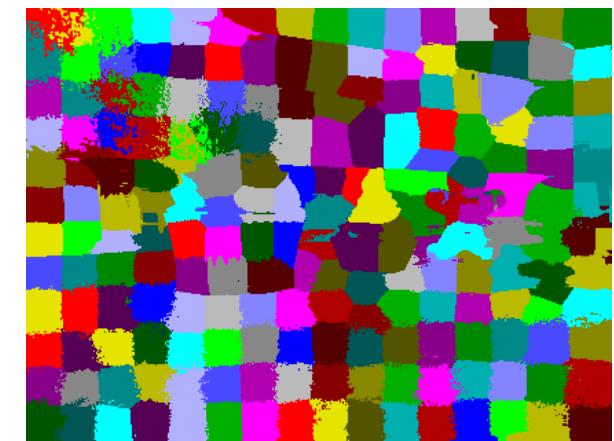
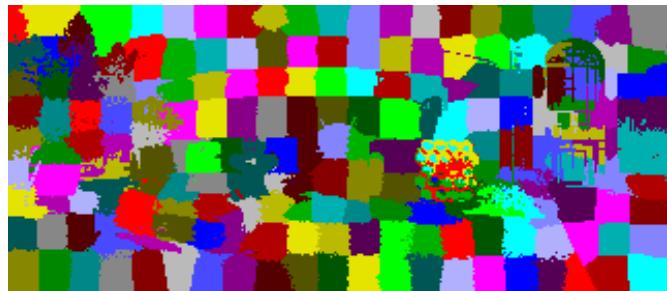
adaptive means, h=1



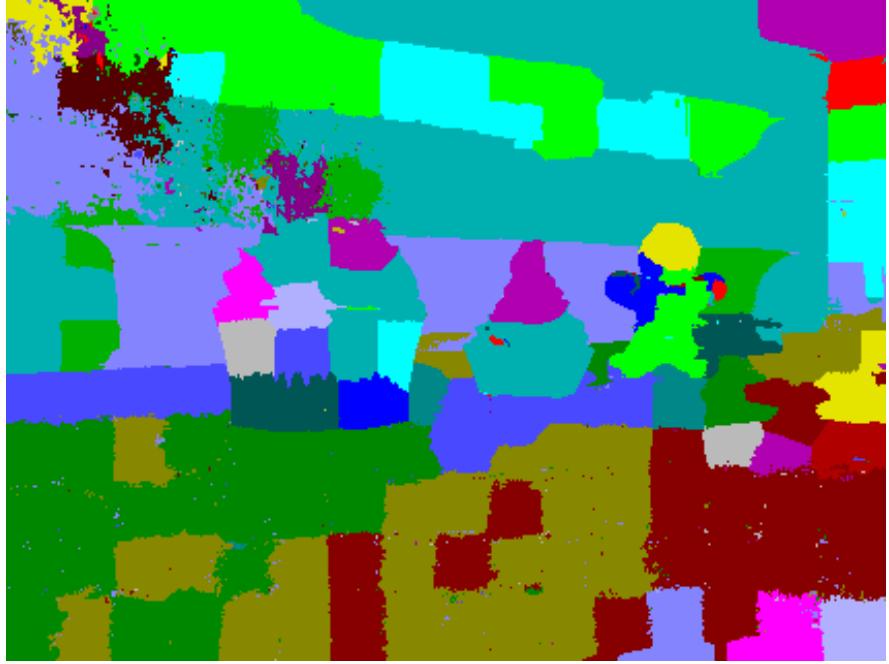
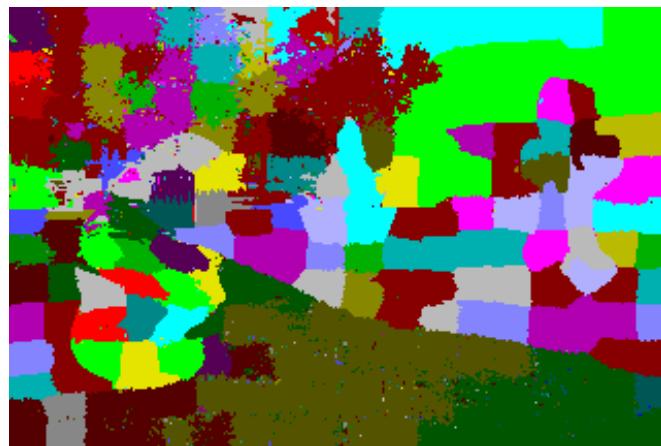
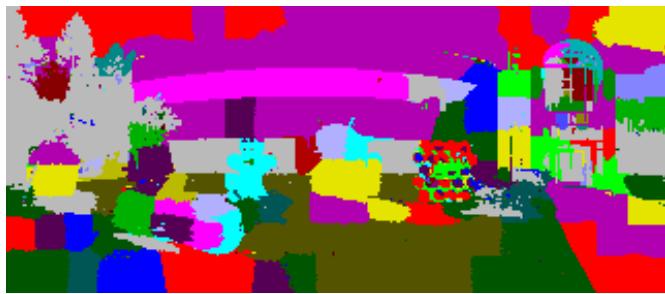
two median smooth with h=2



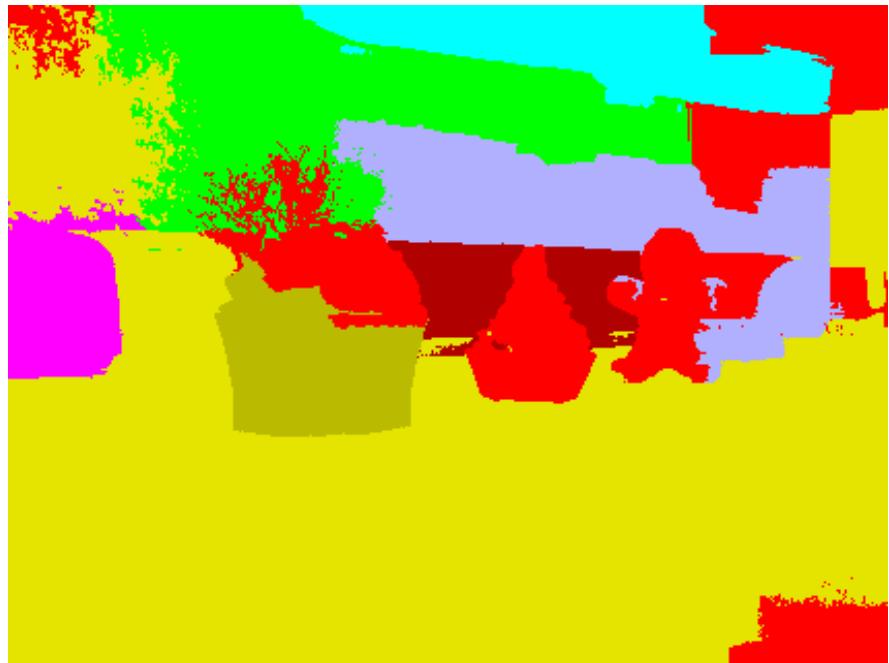
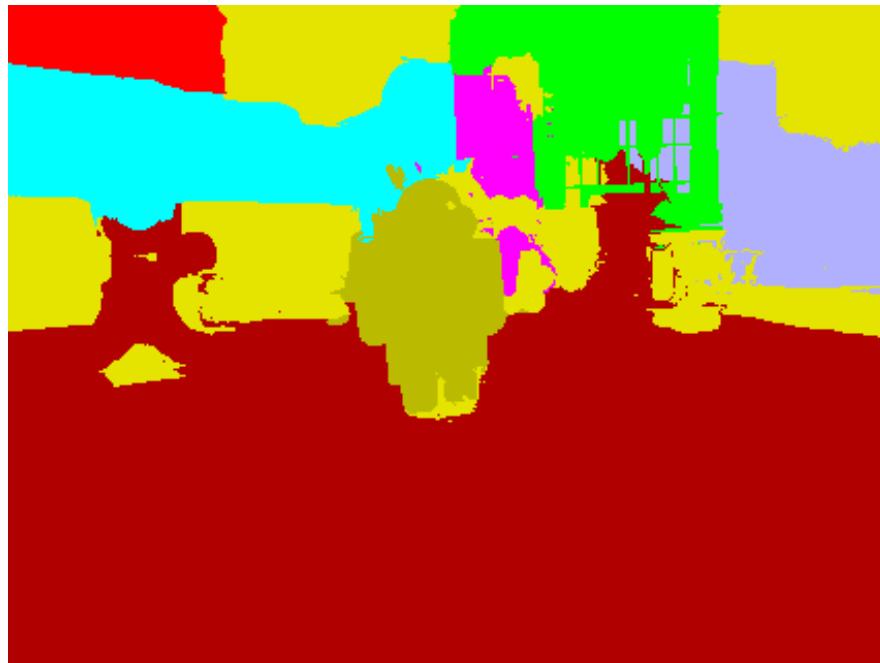
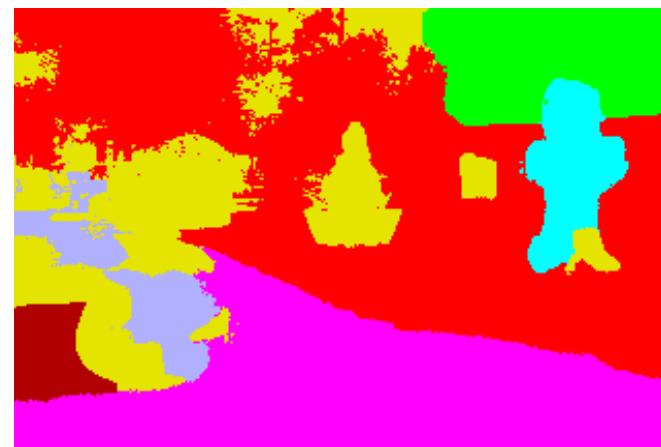
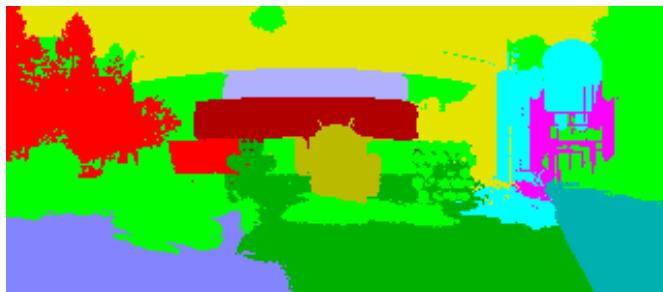
SLIC super pixels algorithm



CIE XY Theta Clustering of the super pixels



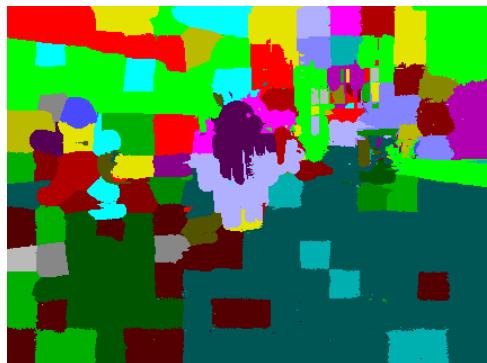
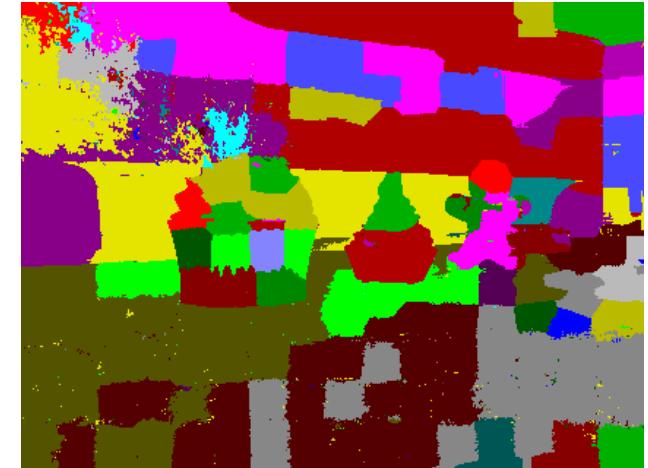
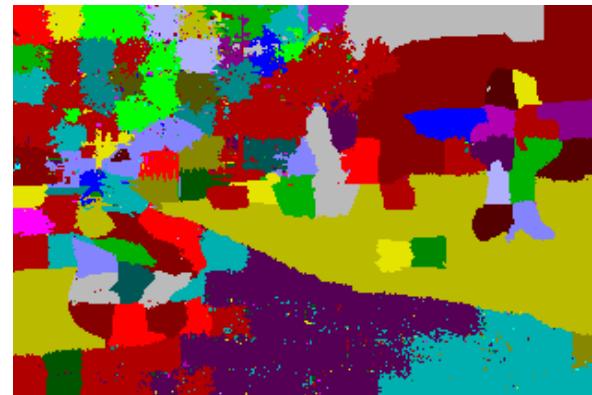
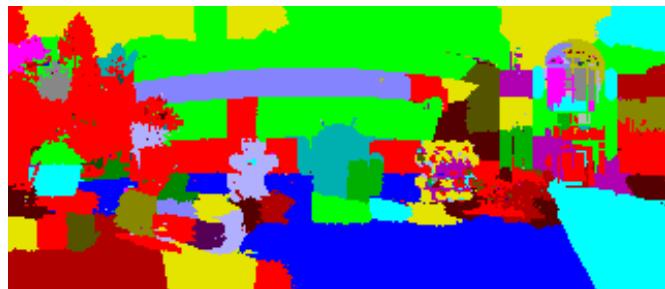
Normalized Cuts w/ HSV cs on the super pixels



combination of algorithms:

- super pixels
- normalized cuts using color space hsv
- restoration of super pixels where canny edge boundaries are lost
- feedback into normalized cuts (this time, having larger cells of background and foreground segments as input)
- Next: (*not yet implemented*) labels3 should always be merging of labels2, so needs edits to restore any labels2 that hold more than one label from labels3

Result is that foreground and background are larger segments.



results before last unfinished step

Partial Shape Matching of the gingerbread man

extract ordered borders of shapes, blur them by one or two sigma, then ordered bipartite matching algorithm using difference of chords as a descriptor.

