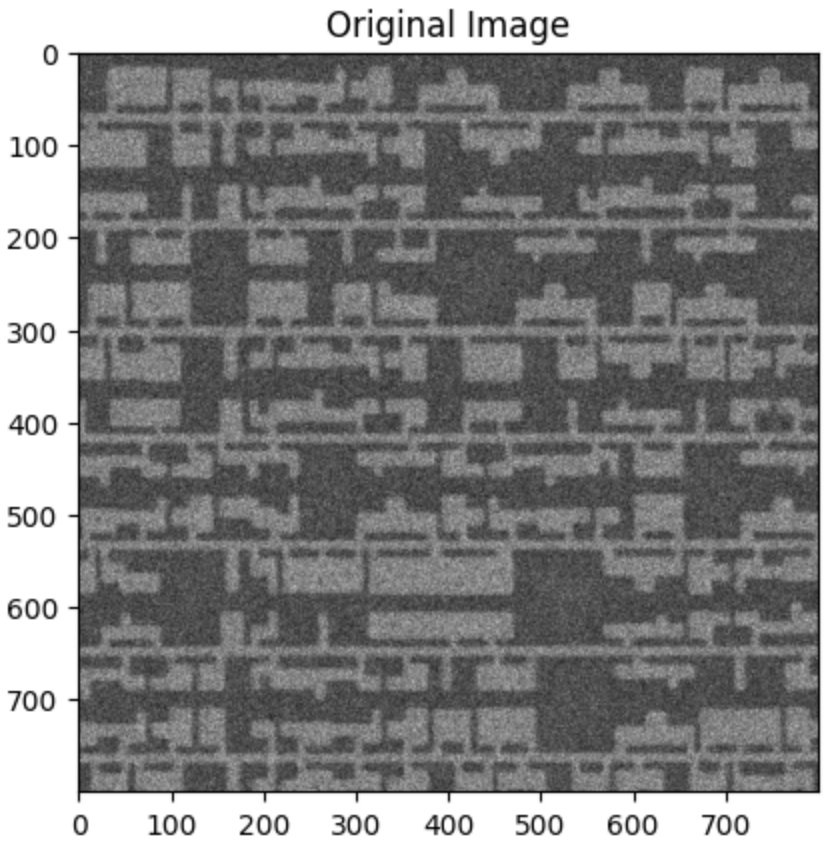
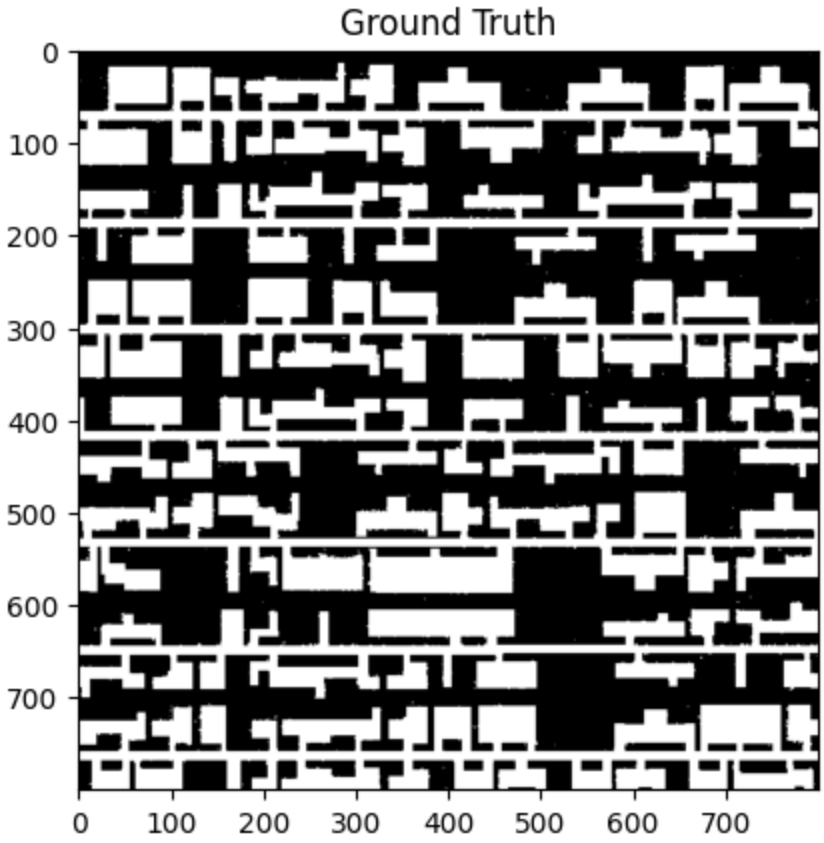
In [1]:
```
# import libraries
import numpy as np
import matplotlib.pyplot as plt
import skimage
import sklearn
```

In [2]:
```
# load and show original image

def imshow(img:np.ndarray, title:str):
    plt.figure()
    plt.imshow(img, cmap='gray')
    plt.title(title)
    plt.show()
    # print image statistics
    print('Type', type(img), img.dtype)
    print('Shape', img.shape)
    print('Range', np.min(img), '-', np.max(img))

original_image = skimage.io.imread('img_0.png')
imshow(original_image, 'Original Image')

# load and show ground truth
ground_truth = skimage.io.imread('gt_0.png').astype('bool')
imshow(ground_truth, 'Ground Truth')
```

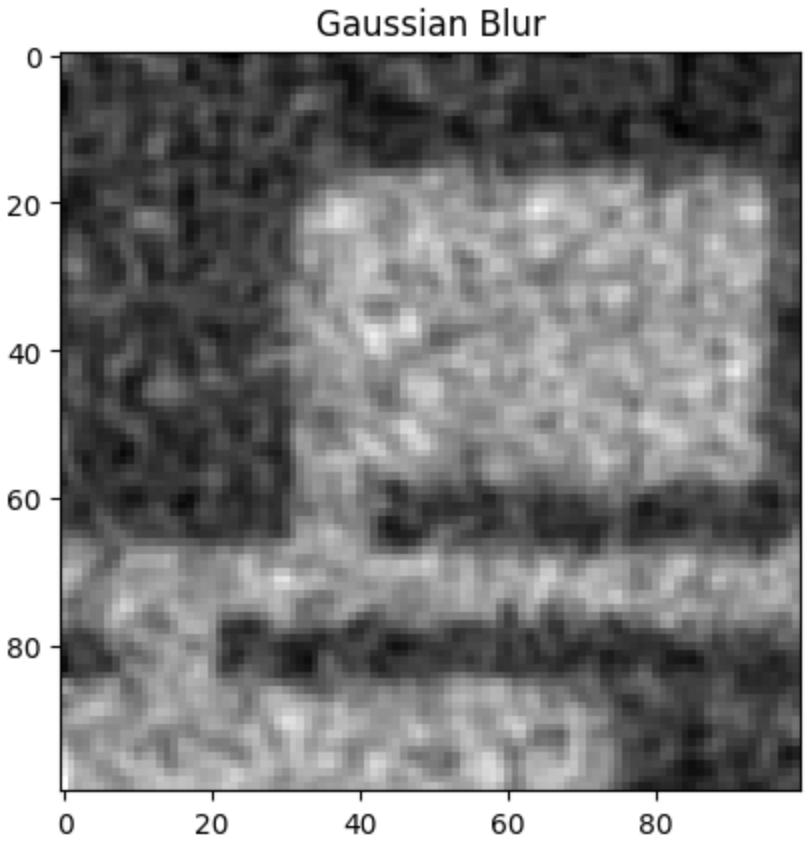



```
Type <class 'numpy.ndarray'> uint8
Shape (800, 800)
Range 0 - 255
```

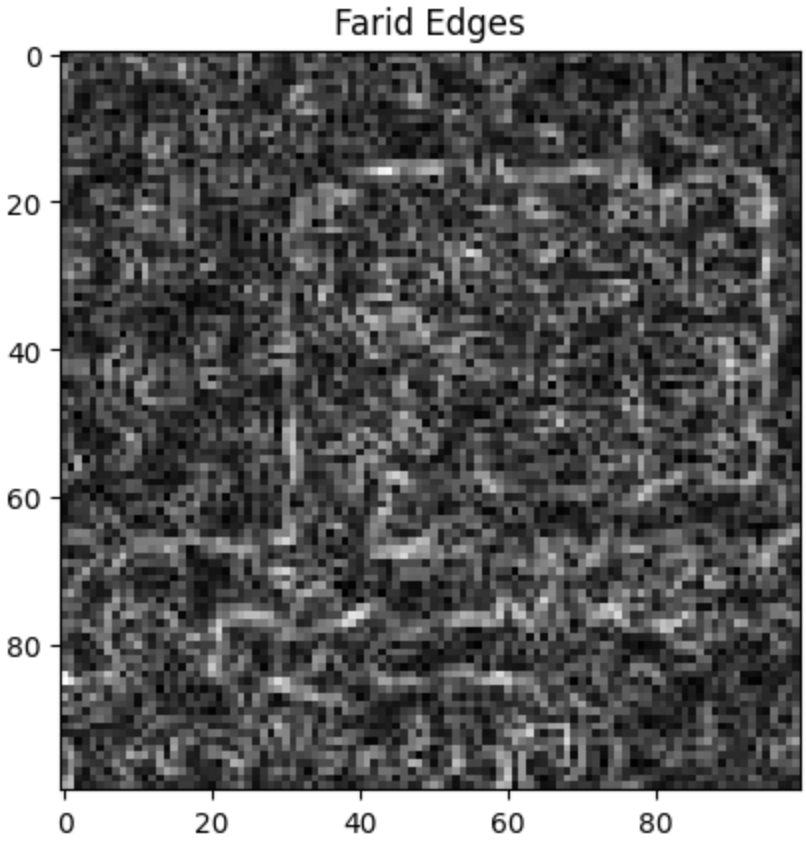



```
Type <class 'numpy.ndarray'> bool
Shape (800, 800)
Range False - True
```

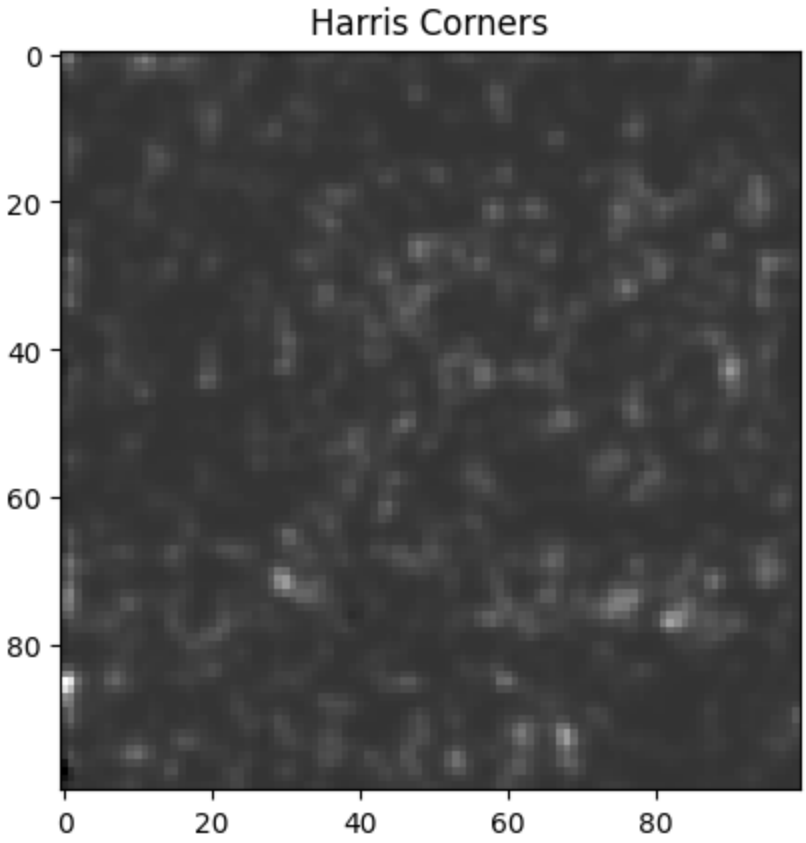
In [3]:
```
# feature extraction - intensity, edges, corners
intensity = skimage.filters.gaussian(original_image)
imshow(intensity[0:100, 0:100], 'Gaussian Blur')

edges = skimage.filters.farid(original_image)
imshow(edges[0:100, 0:100], 'Farid Edges')

corners = skimage.feature.corner_harris(original_image)
imshow(corners[0:100, 0:100], 'Harris Corners')

# format features
features = np.dstack((intensity, edges, corners))
features = features.reshape(800*800, -1)
print(features.shape)
```




```
Type <class 'numpy.ndarray'> float64
Shape (100, 100)
Range 0.18701638339792598 - 0.7106708581970249
```




```
Type <class 'numpy.ndarray'> float64
Shape (100, 100)
Range 0.0003618963572120814 - 0.11310742596480271
```



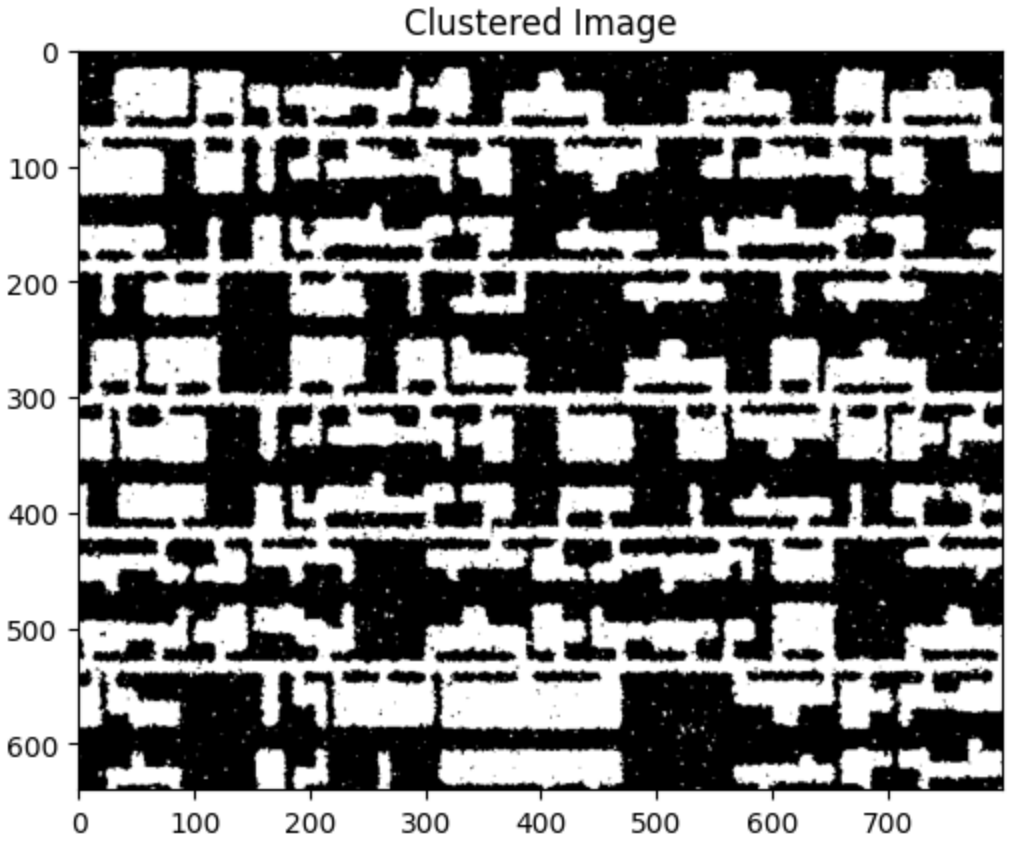
```
Type <class 'numpy.ndarray'> float64
Shape (100, 100)
Range -0.2331053845414695 - 0.9422119927366289
(640000, 3)
```

In [4]:
```
# train/test split
import sklearn.model_selection

test_percent = 0.20
n_test_rows = int(test_percent*800)
n_train_rows = 800-n_test_rows
print(n_test_rows, n_train_rows)

feats_train, feats_test, gt_train, gt_test = sklearn.model_selection.train_test
features, ground_truth.flatten().astype(int), test_size=test_percent, shuffle=
```

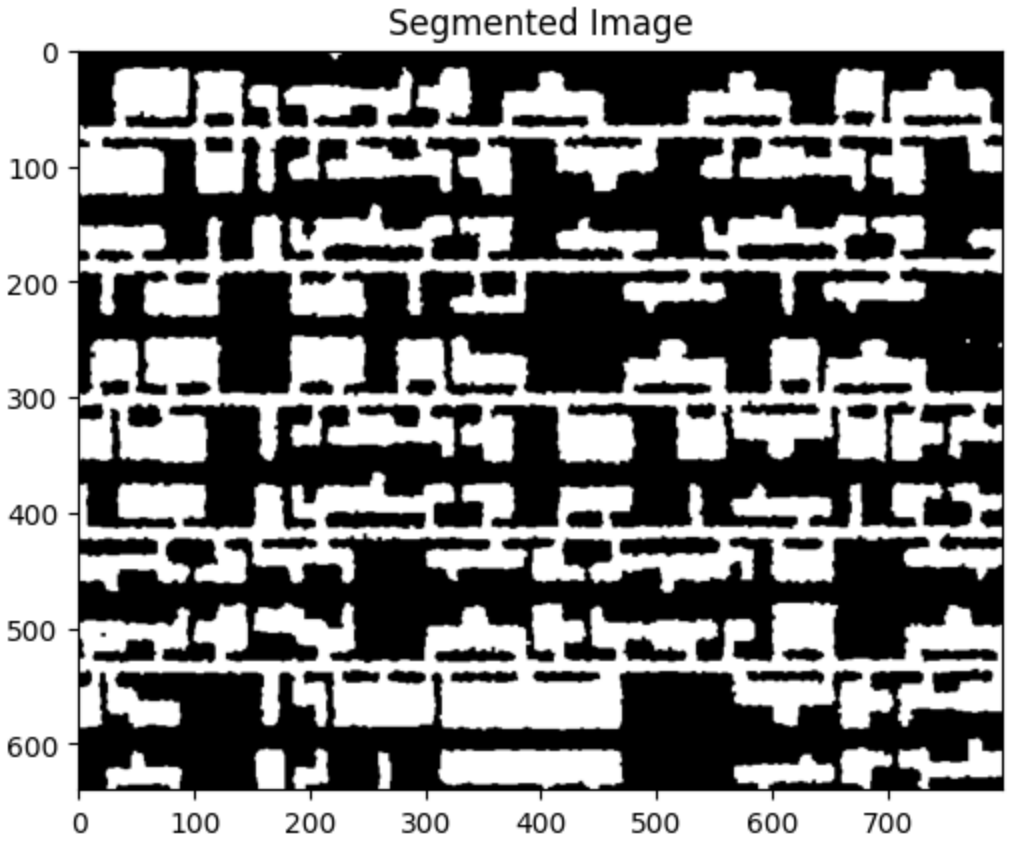
```
160 640
```

In [5]:
```
# TRAINING

# unsupervised ML - kmeans
import sklearn.cluster
model = sklearn.cluster.KMeans(n_clusters=2, random_state=12)
model.fit(feats_train)
clustered_image = model.predict(feats_train)
clustered_image = clustered_image.reshape(n_train_rows,800)
imshow(clustered_image, 'Clustered Image')

# postprocessing - morphological operations
# input: clustered_image, out: segmentation
structuring_element = np.ones([3,3])
print(structuring_element)
postprocess_image = skimage.morphology.binary_opening(clustered_image, structu
segmentation = skimage.morphology.binary_closing(postprocess_image, structurin
imshow(segmentation, 'Segmented Image')

# evaluation - IoU
import sklearn.metrics
iou = sklearn.metrics.jaccard_score(segmentation.flatten(),
                                    gt_train.flatten())
print('Iou', iou)

# visualize results
visualization = skimage.color.label2rgb(segmentation,
                                        original_image[0:n_train_rows, 0:800],
                                        ['yellow'])
imshow(visualization[0:100, 0:100], 'Result Overlay')
```

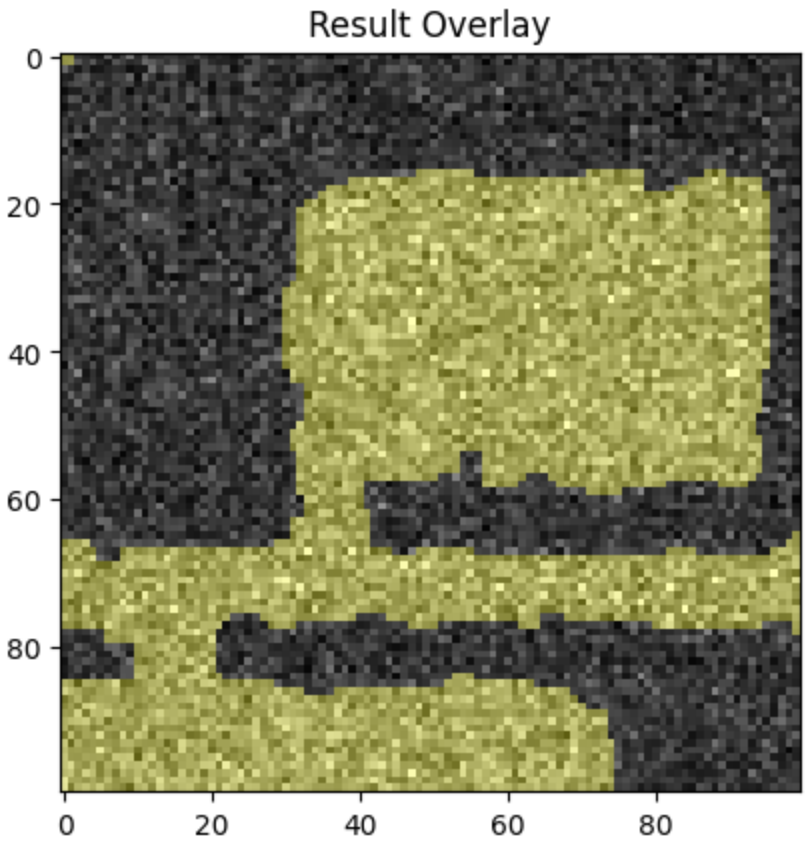
```
C:\Users\Olivia\Anaconda3\envs\ic_sem_re_tutorial\lib\site-packages\sklearn\c
luster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppres
s the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




```
Type <class 'numpy.ndarray'> int32
Shape (640, 800)
Range 0 - 1
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```




```
Type <class 'numpy.ndarray'> bool
Shape (640, 800)
Range False - True
Iou 0.8816728746810799
```

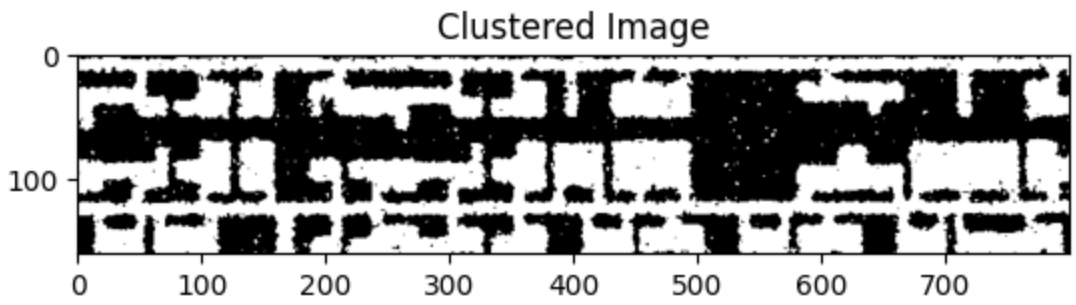


```
Type <class 'numpy.ndarray'> float64
Shape (100, 100, 3)
Range 0.002745098039215686 - 1.0
```

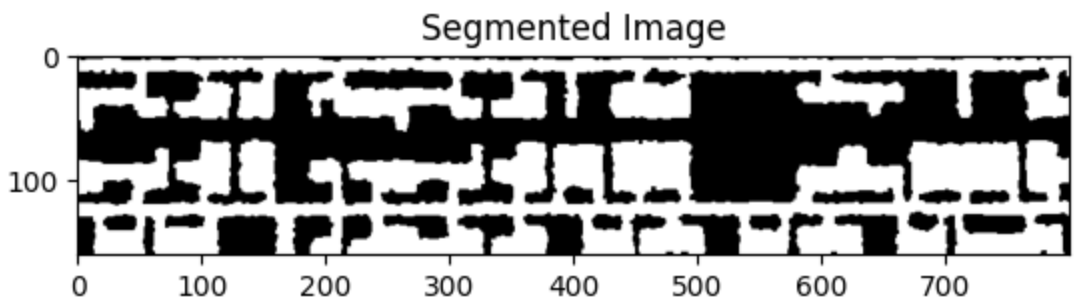
In [6]:
```
# model interpretation - cluster centers
for label, model.cluster_center in enumerate(model.cluster_centers_):
    print('label', label, '-', model.cluster_center)
# intensity, edges, corners
```

```
label 0 - [0.31454059 0.0279947  0.02885523]
label 1 - [0.50146887 0.03532551 0.06544869]
```

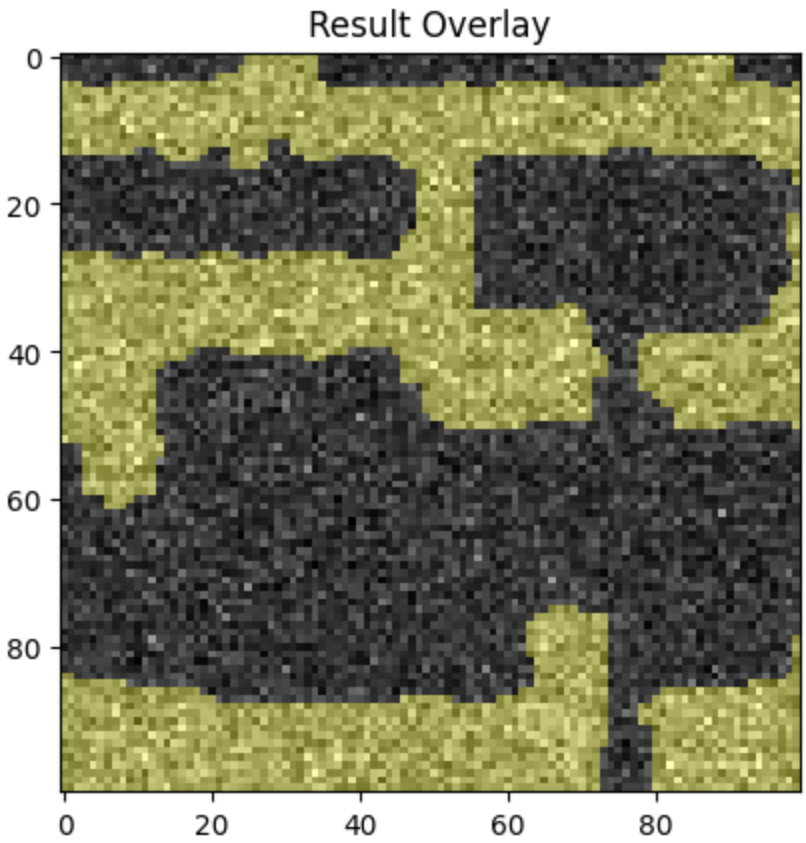
In [7]:
```
# TESTING

# unsupervised ML - kmeans
clustered_image = model.predict(feats_test)
clustered_image = clustered_image.reshape(n_test_rows,800)
imshow(clustered_image, 'Clustered Image')

# postprocessing - morphological operations
# input: clustered_image, out: segmentation
structuring_element = np.ones([3,3])
print(structuring_element)
postprocess_image = skimage.morphology.binary_opening(clustered_image, structu
segmentation = skimage.morphology.binary_closing(postprocess_image, structurin
imshow(segmentation, 'Segmented Image')

# evaluation - IoU
import sklearn.metrics
iou = sklearn.metrics.jaccard_score(segmentation.flatten(),
                                    gt_test.flatten())
print('Iou', iou)

# visualize results
visualization = skimage.color.label2rgb(segmentation,
                                        original_image[n_train_rows:800, 0:800]
                                        ['yellow'])
imshow(visualization[0:100, 0:100], 'Result Overlay')
```




```
Type <class 'numpy.ndarray'> int32
Shape (160, 800)
Range 0 - 1
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```



```
Type <class 'numpy.ndarray'> bool
Shape (160, 800)
Range False - True
Iou 0.8908481536278269
```




```
Type <class 'numpy.ndarray'> float64
Shape (100, 100, 3)
Range 0.005490196078431372 - 1.0
```