IMAGE SEGMENTATION

PATRICK KEVORKIAN

TOPICS

- Bi-level thresholding
- Multi-level thresholding
- Histogram Based Segmentation
- Local Adaptive thresholding
- Niblack’s Locally Adaptive Thresholding
- Sauvola’s Locally Adaptive Thresholding

BI-LEVEL

- One Threshold
- Two regions (black & white)
- New pixel value determined on if its greater or less than the threshold
- Too high, white will obscure image
- Too low, black will obscure image
- Has issues with uneven illumination

EXAMPLE 1

THRESH = 55

Binary Image, obtained by thresholding

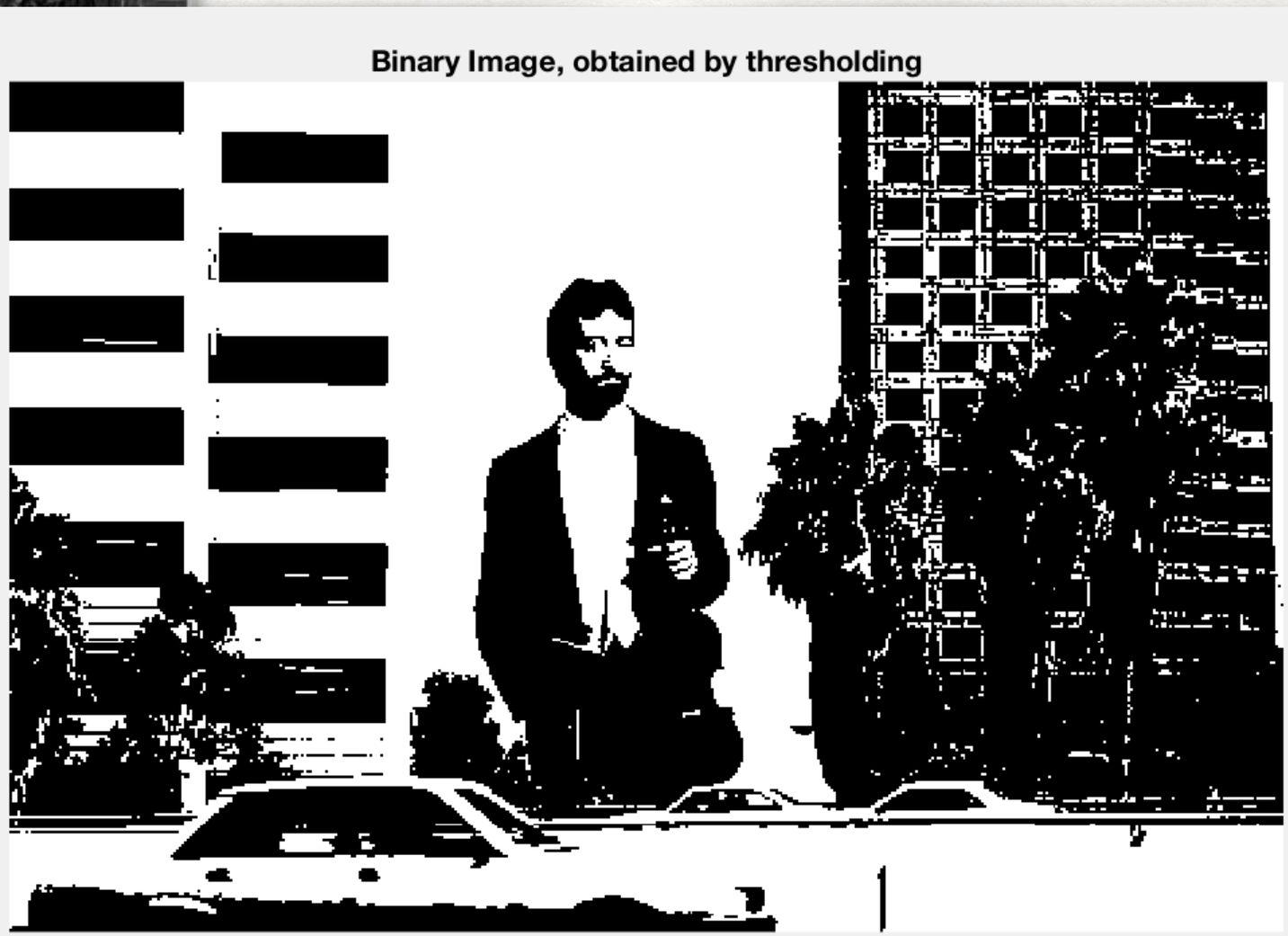
EXAMPLE 2

THRESH = 75

Binary Image, obtained by thresholding

EXAMPLE 3

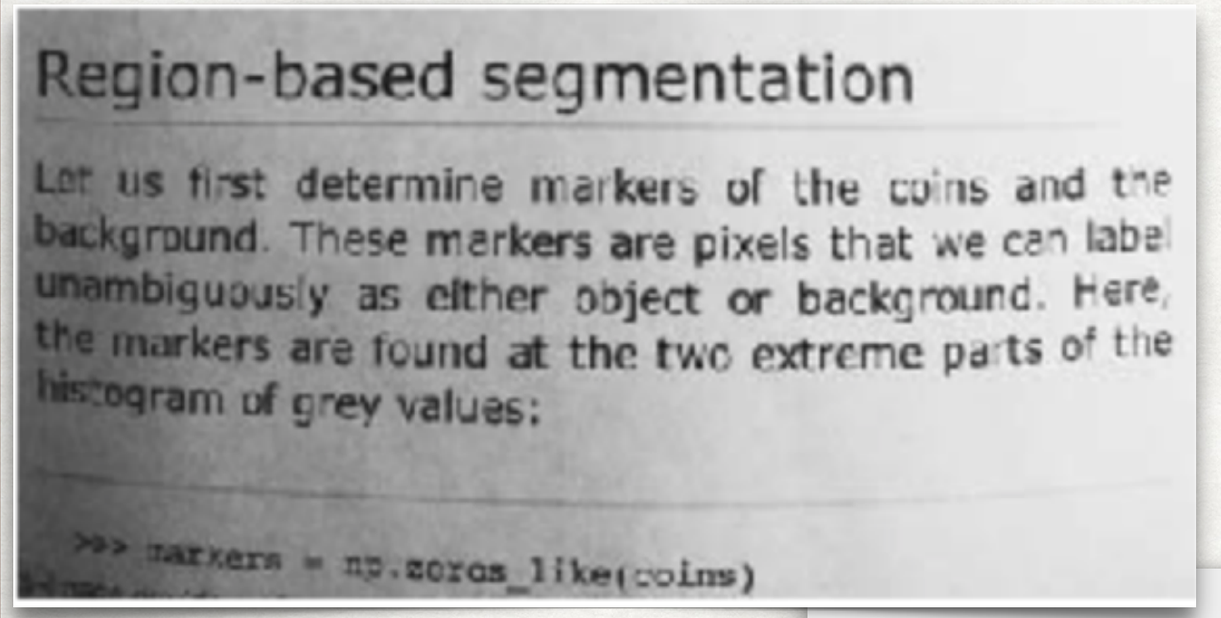
THRESH = 100



EXAMPLE 3 BERKLEY RESULTS

EXAMPLE 4

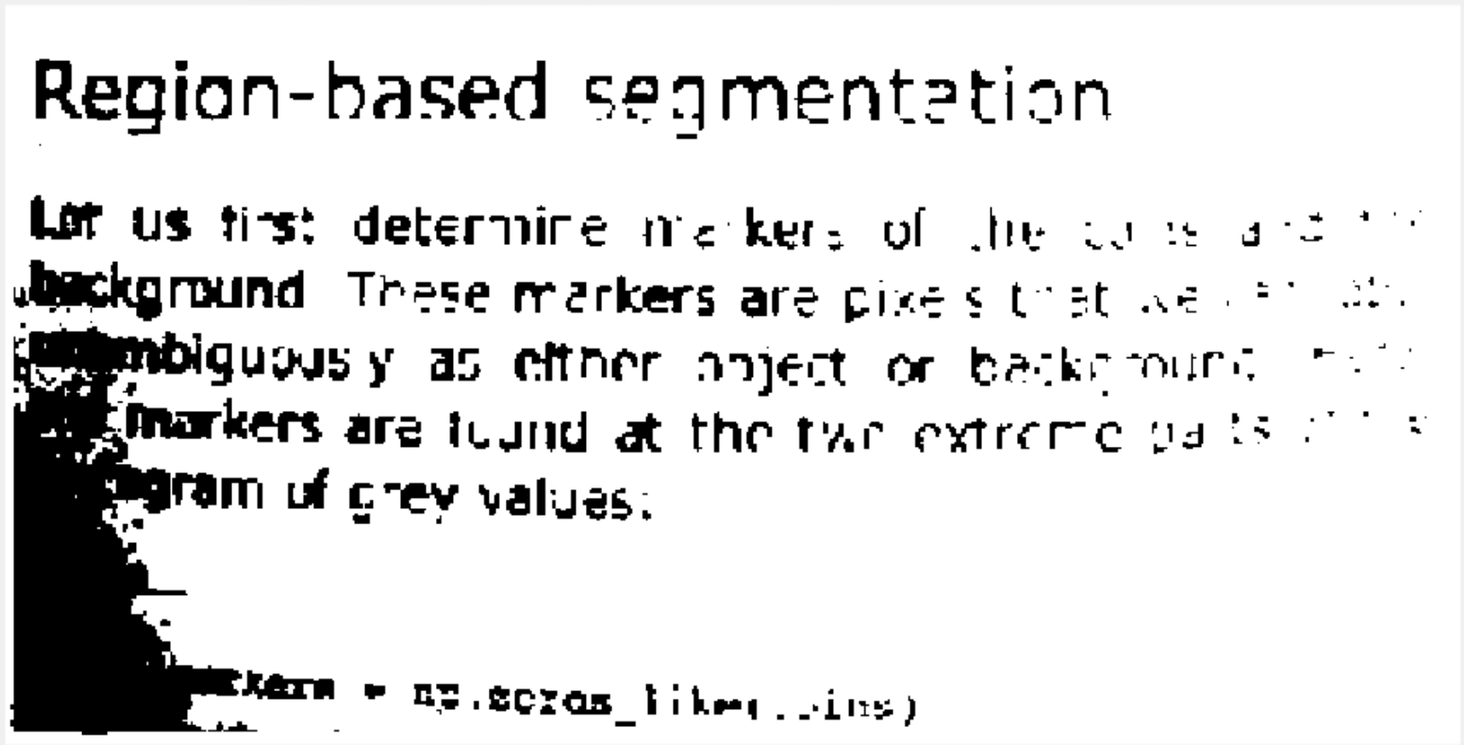
THRESH =100


Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

Binary Image, obtained by thresholding


Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

MULTI-LEVEL

- Multiple thresholds
- Segmentation done with levels of gray
- New pixel value determined by where it sits in between, below, and above thresholds
- Examples using 2 thresholds for total of 3 levels
- Does better with detail and levels of illumination

HISTOGRAM

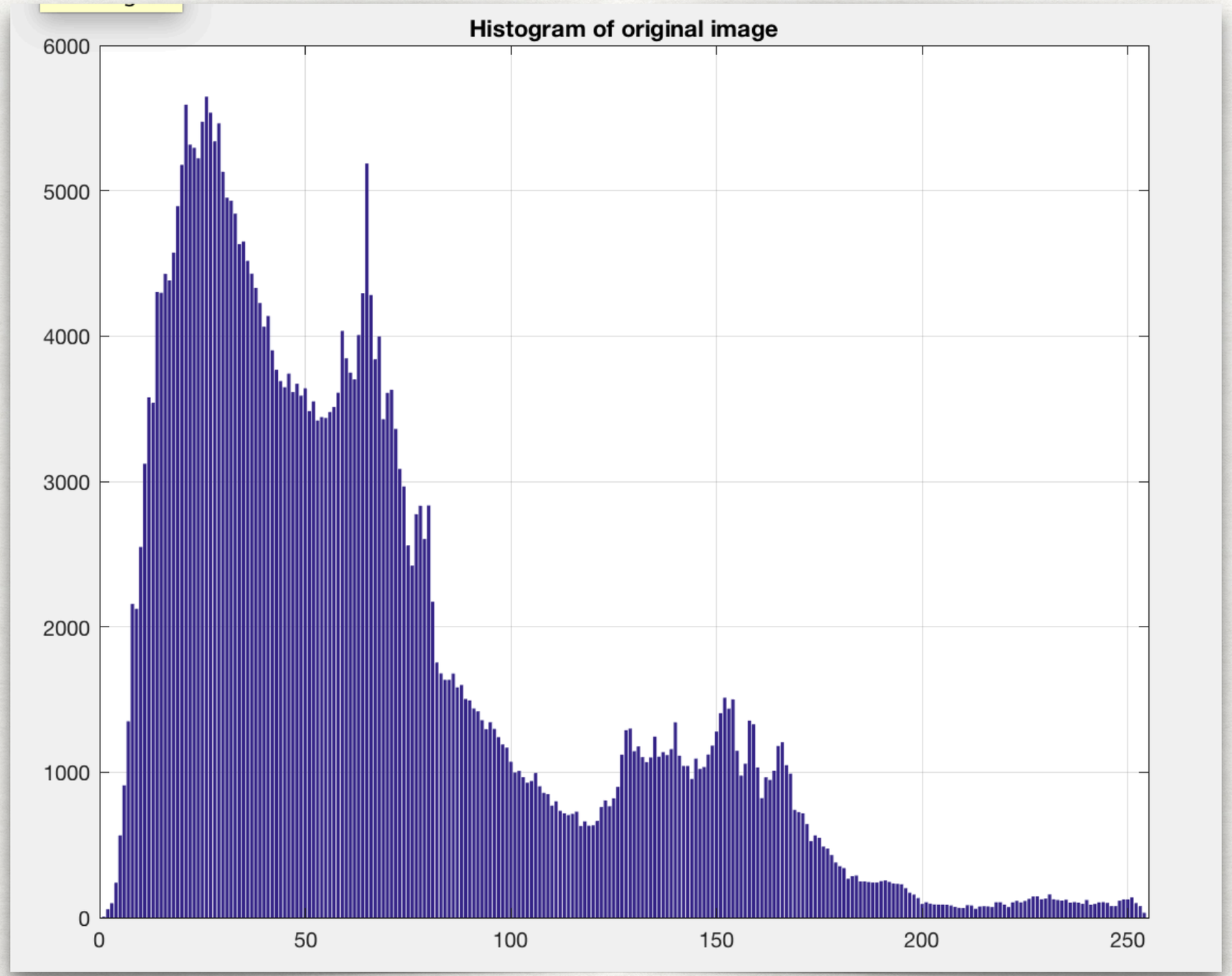
- Use histogram peaks of original image to determine thresholds
- Makes segmentation a bit easier
- But it is difficult to determine number of three olds automatically
- Once again for our examples we are using 2
- Following Multi level segmentation was done with Histogram

EXAMPLE 1

THRESH = 50,130

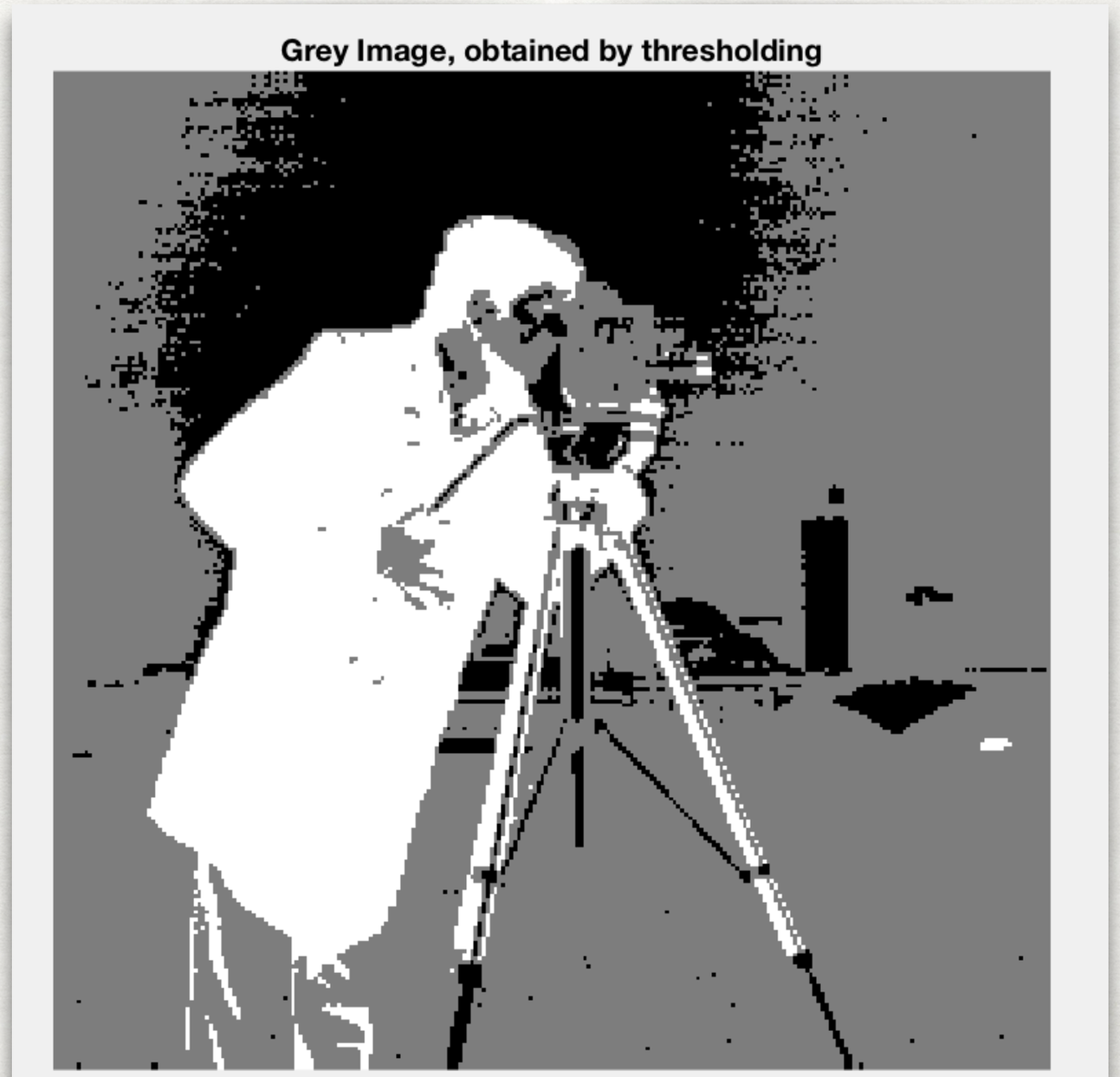
Grey Image, obtained by thresholding

EX 1 HISTOGRAM

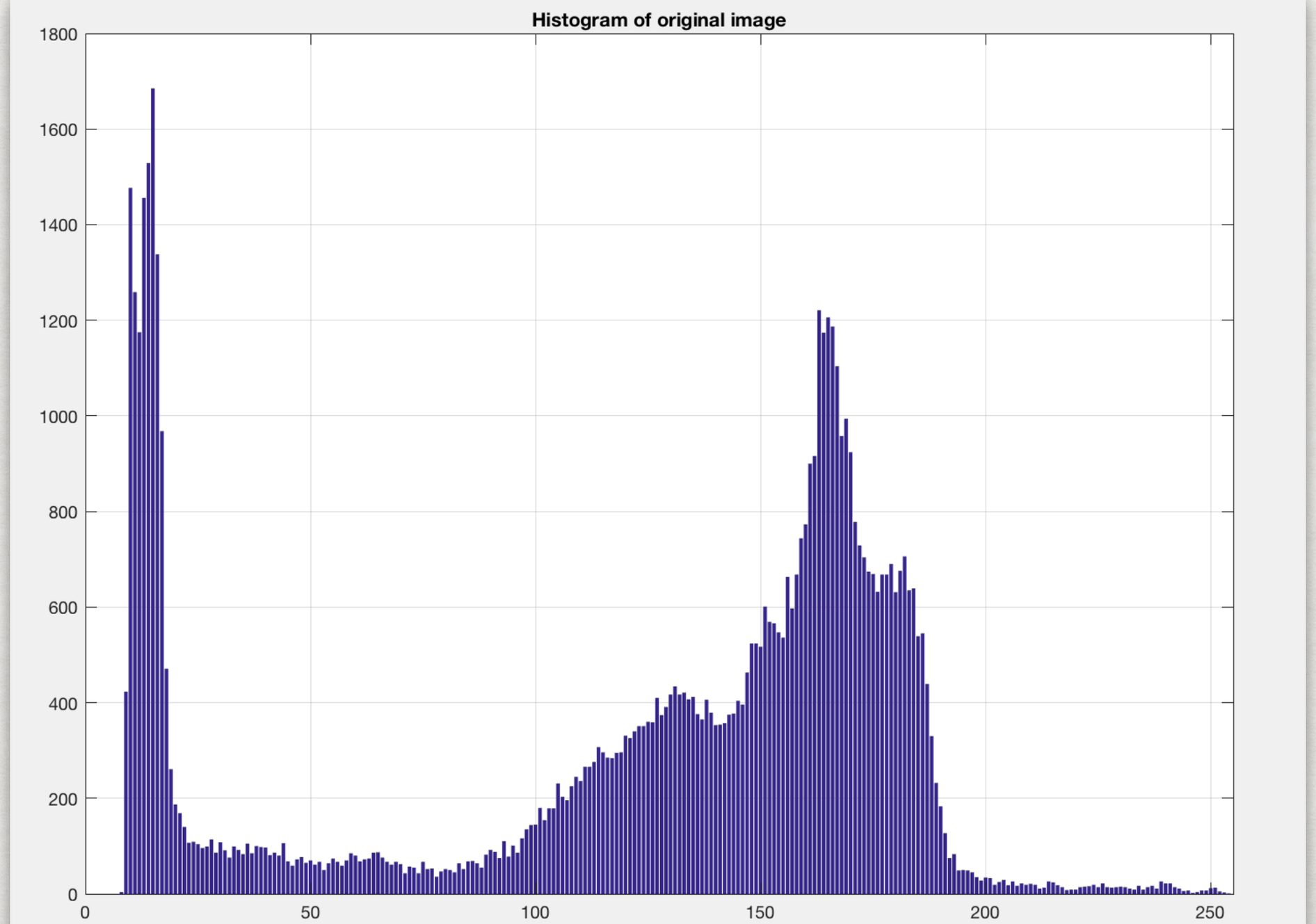


EXAMPLE 2

THRESH = 20, 170



EX 2 HISTOGRAM

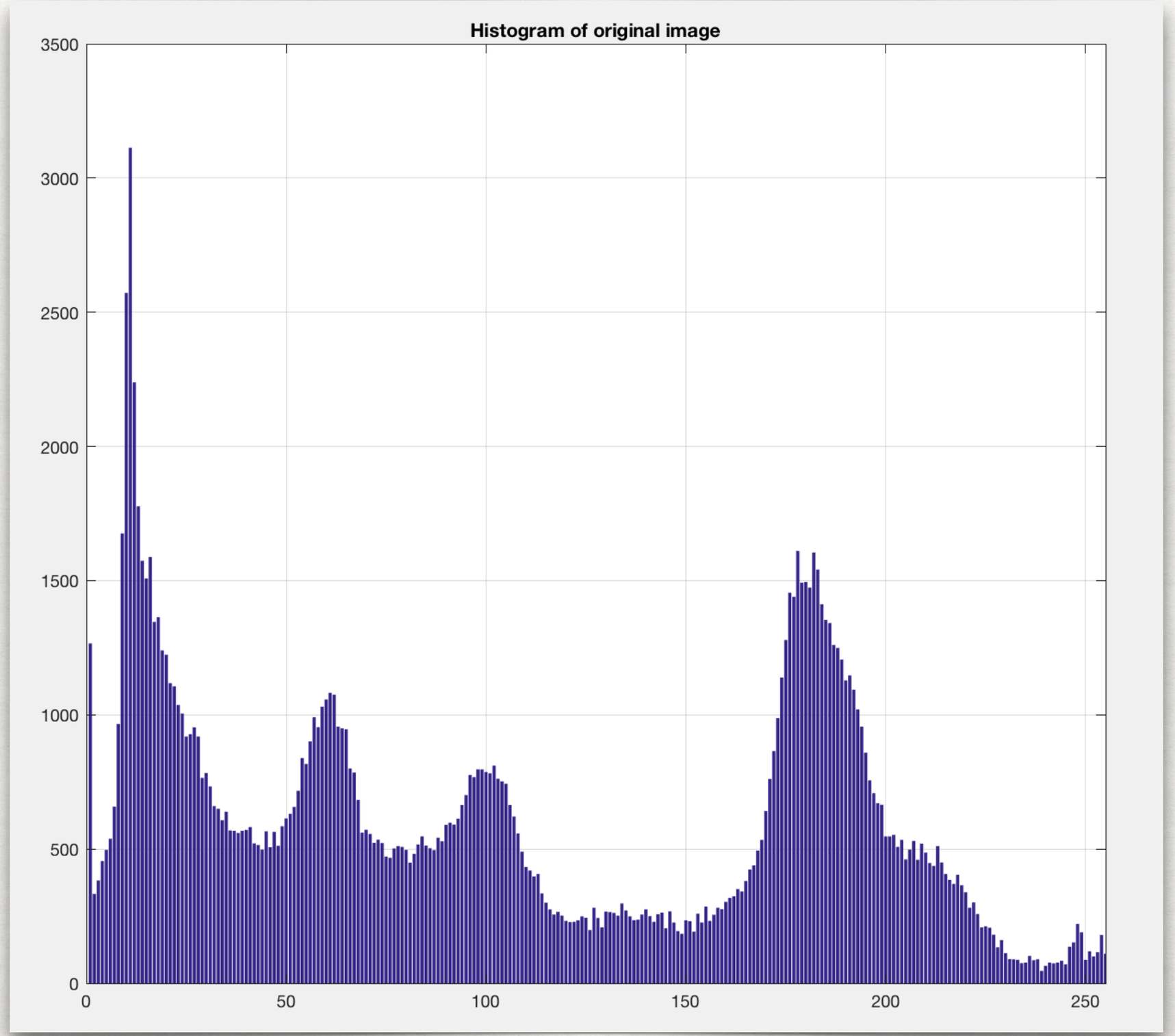


EXAMPLE 3

THRESH = 20,175

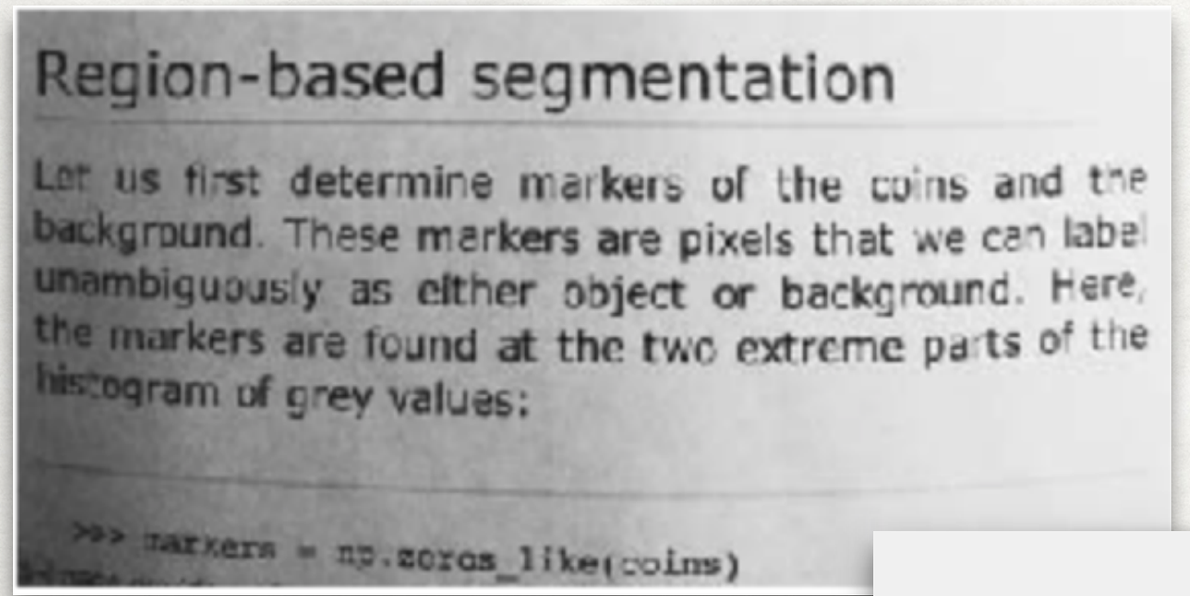
Grey Image, obtained by thresholding

EX 3 HISTOGRAM



EXAMPLE 4

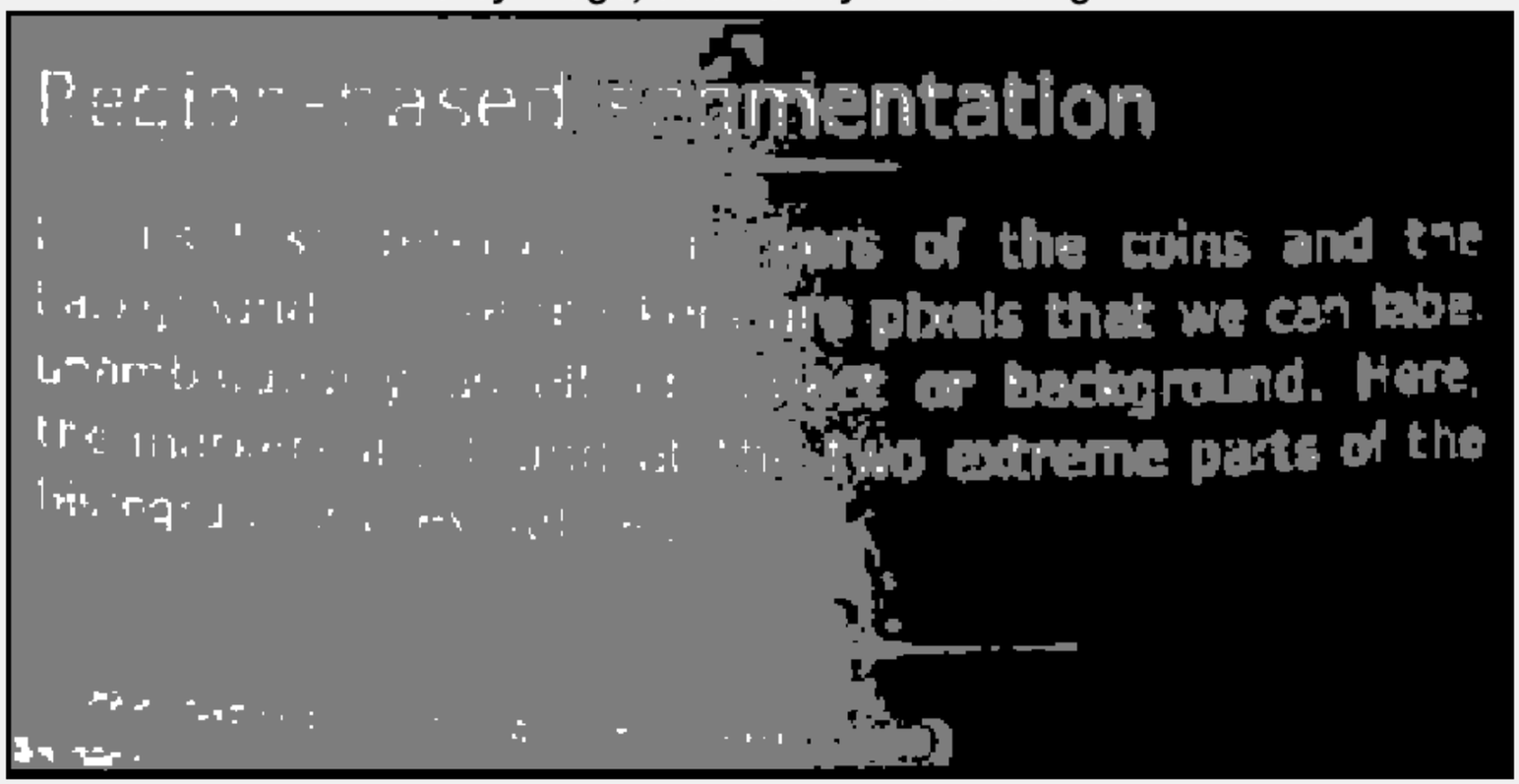
THRESH = 50, 200


Region-based segmentation

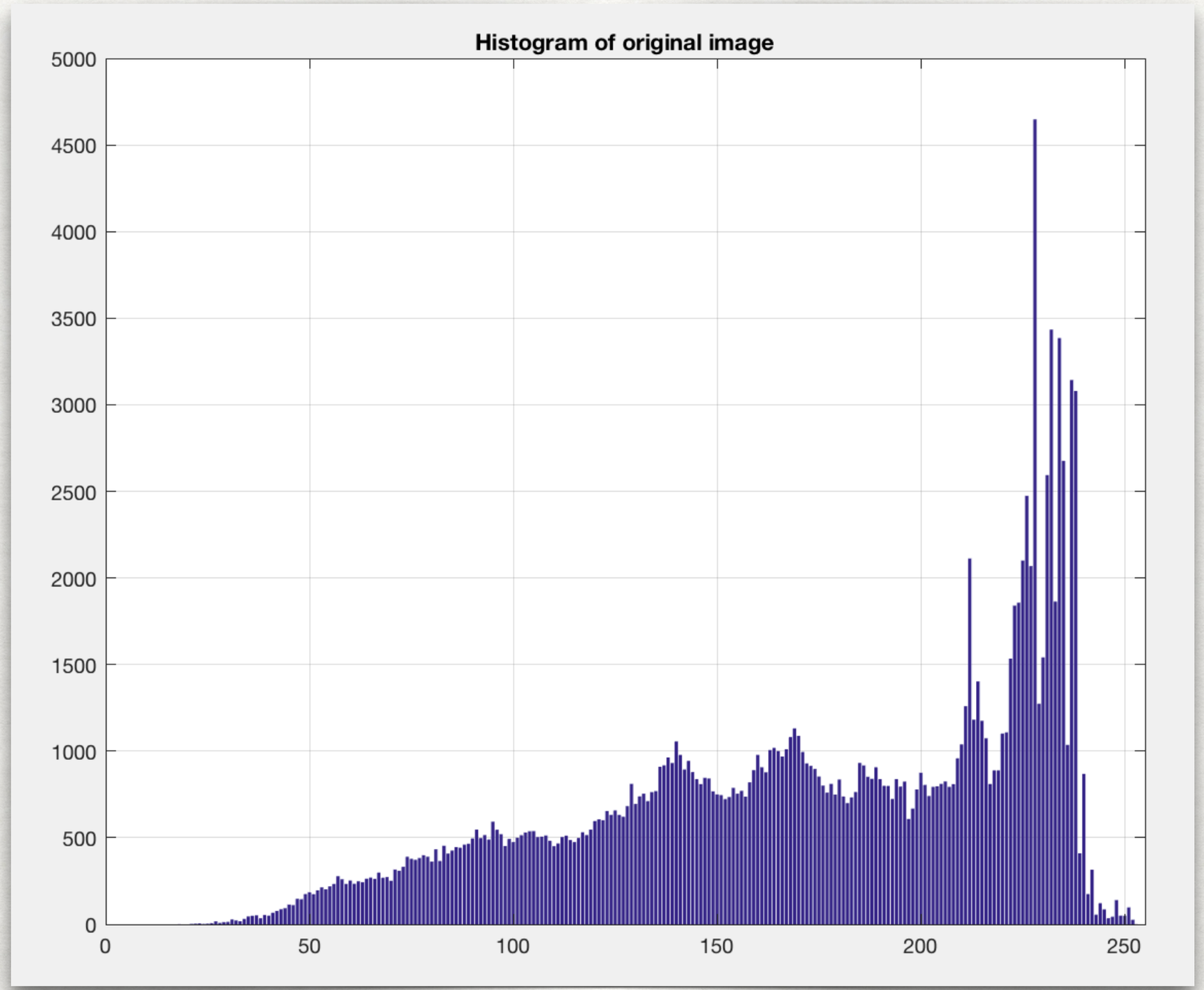
Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

Grey Image, obtained by thresholding



EX 4 HISTOGRAM



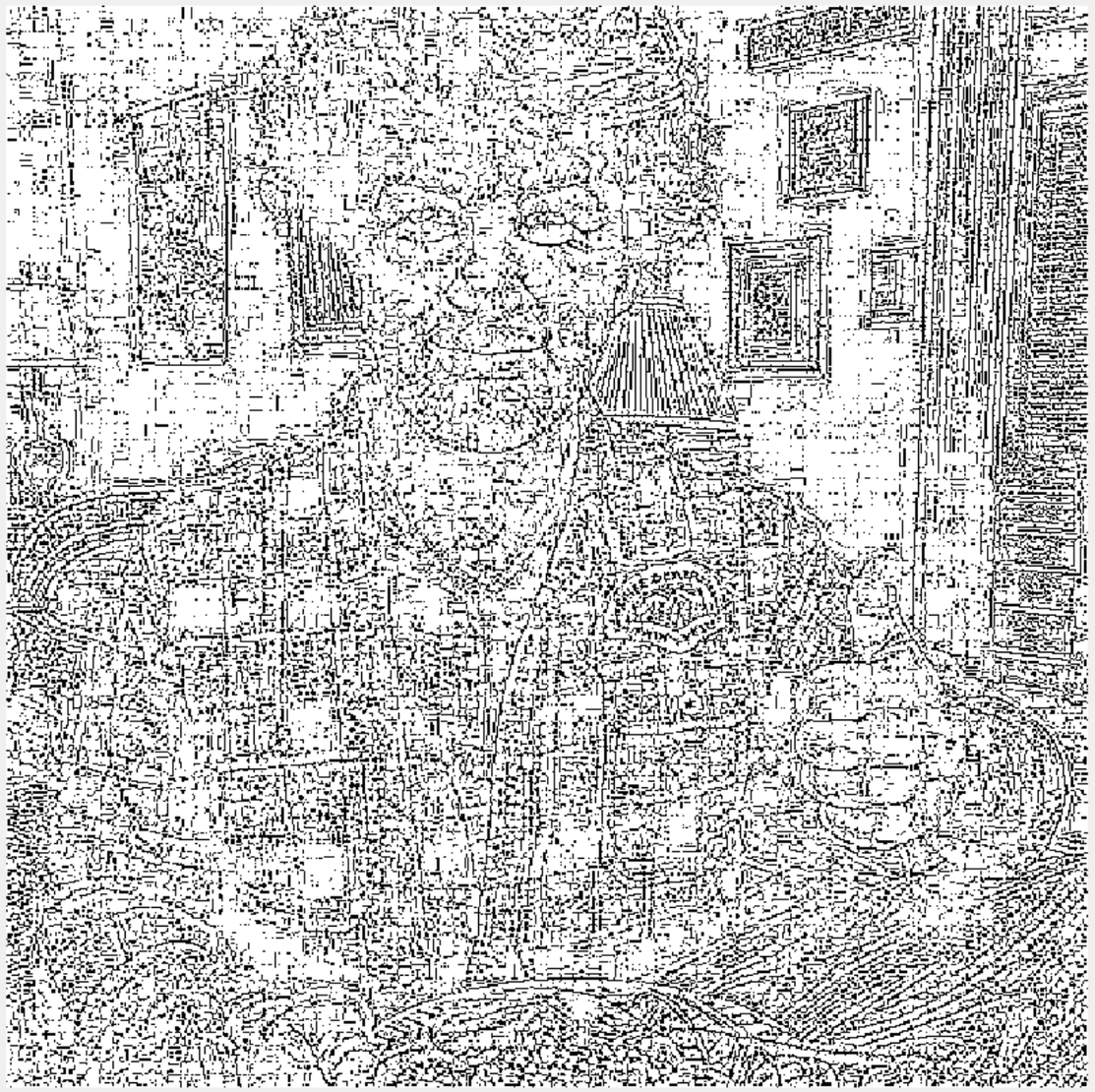
LOCAL ADAPTIVE THRESHOLDING

- Sliding window approach
- Uniform areas are classified as foreground or background
- Non-uniform areas we apply a thresholding method
- Is able to better overcome illumination variances
- But success is not always guaranteed

EXAMPLE 1

DEFAULT: 3X3, 0

DEFAULT MEAN



DEFAULT MEDIAN

EXAMPLE 1

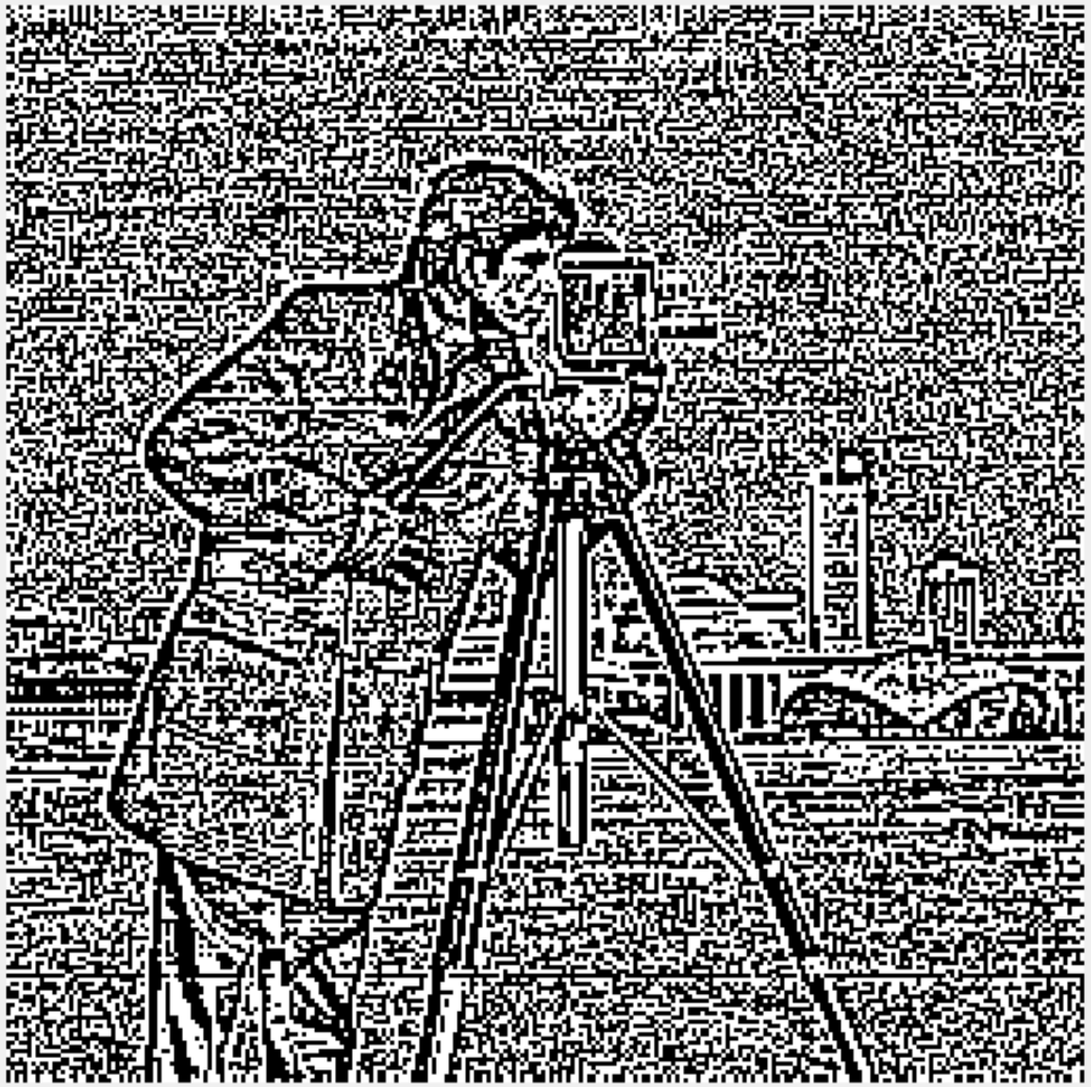
"PLAYING" FOR BETTER RESULTS:

MEAN: 9X9, 0.11

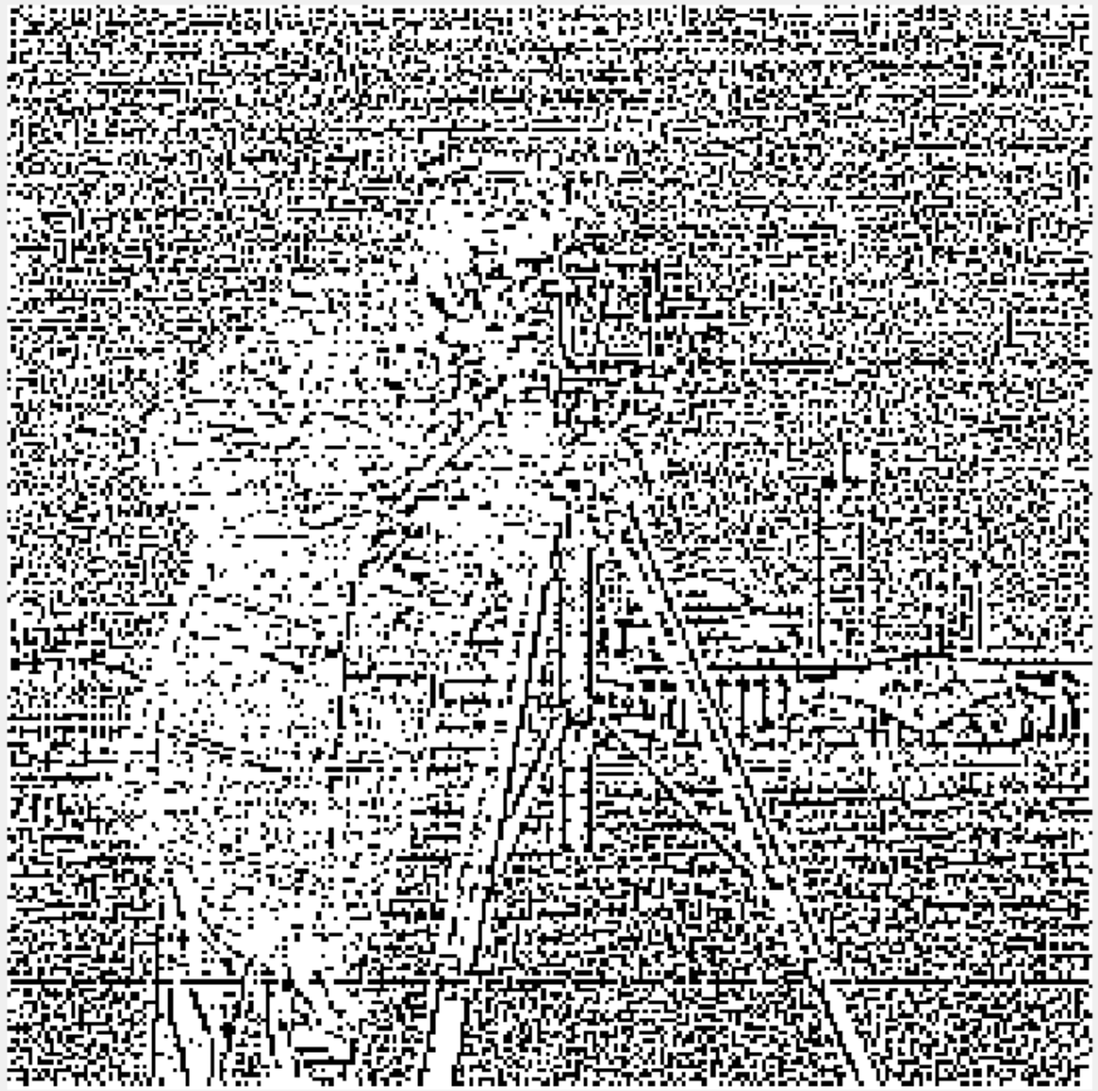
MEDIAN: 9X9, 0.30

EXAMPLE 2

DEFAULT: 3X3, 0



DEFAULT MEAN



DEFAULT MEDIAN

EXAMPLE 2

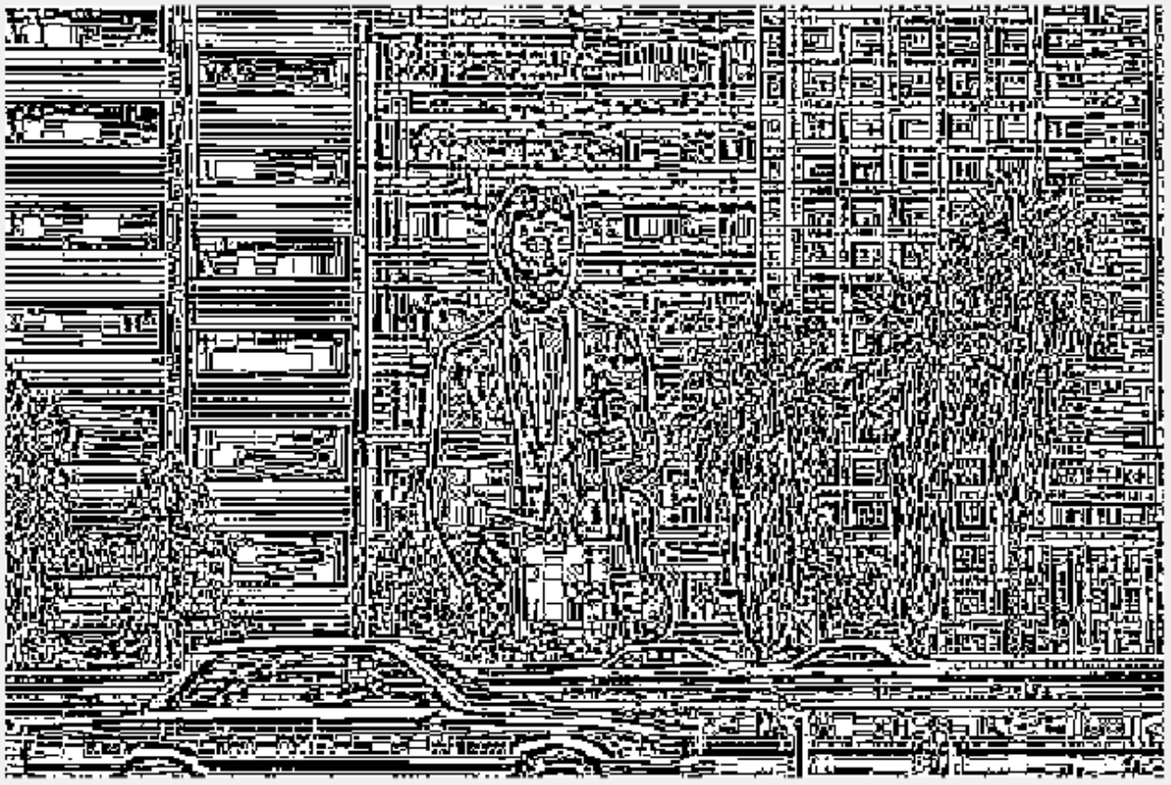
"PLAYING" FOR BETTER RESULTS:

MEAN: 9X9, 0.32

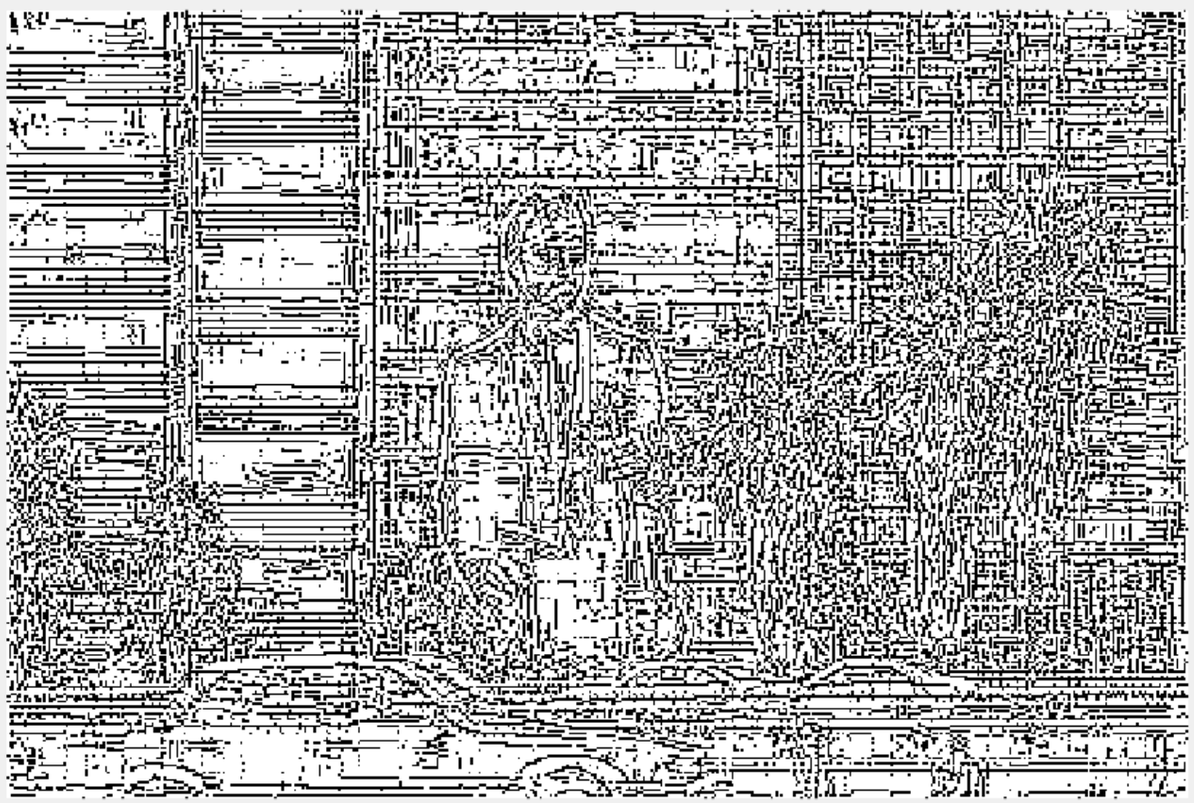
MEDIAN: 9X9, 0.20

EXAMPLE 3

DEFAULT: 3X3, 0



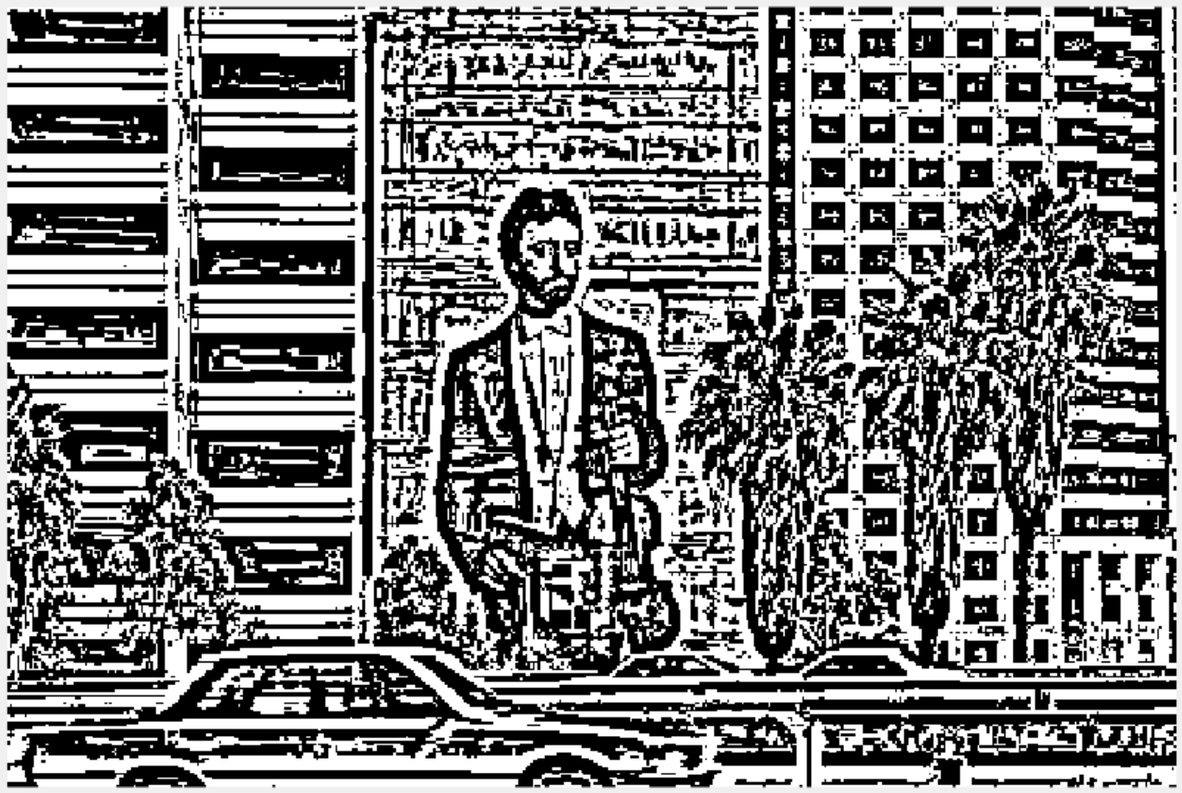
DEFAULT MEAN

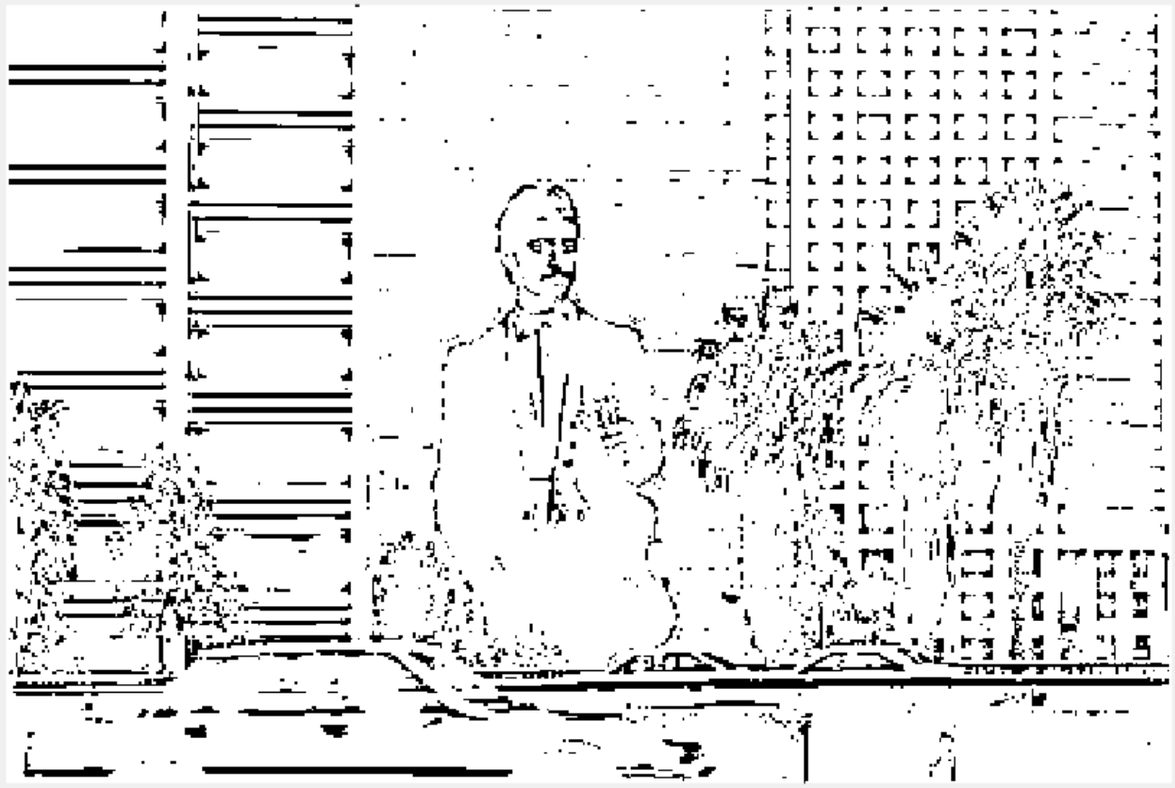


DEFAULT MEDIAN

EXAMPLE 3

"PLAYING" FOR BETTER RESULTS:



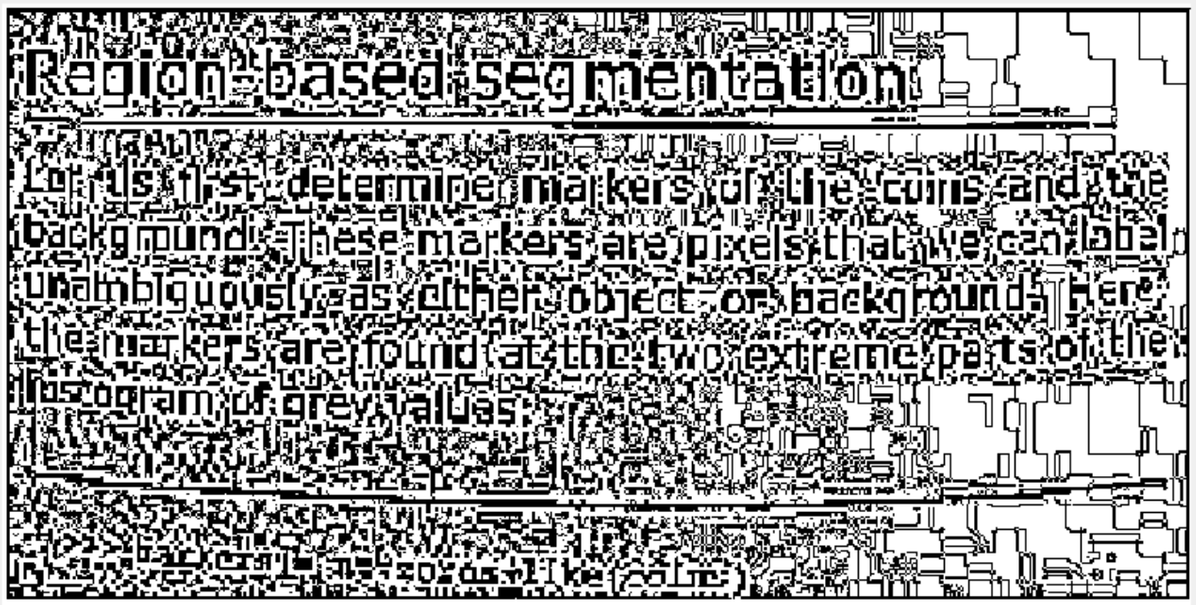


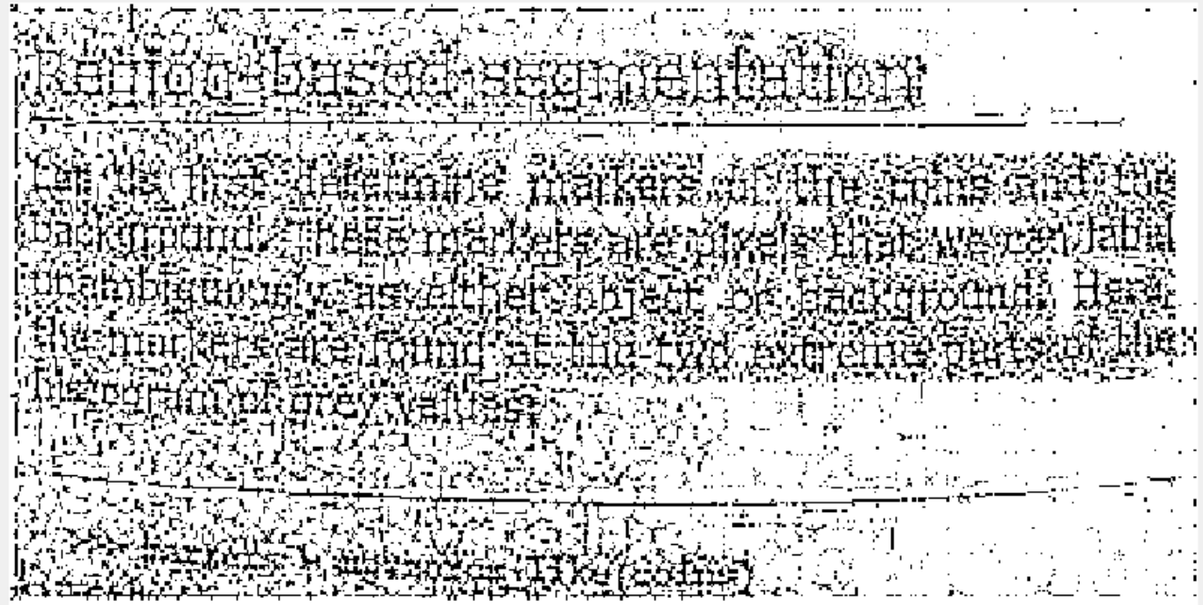
MEAN: 9X9, 0.50

MEDIAN: 9X9, 0.7

EXAMPLE 4

DEFAULT: 3X3, 0



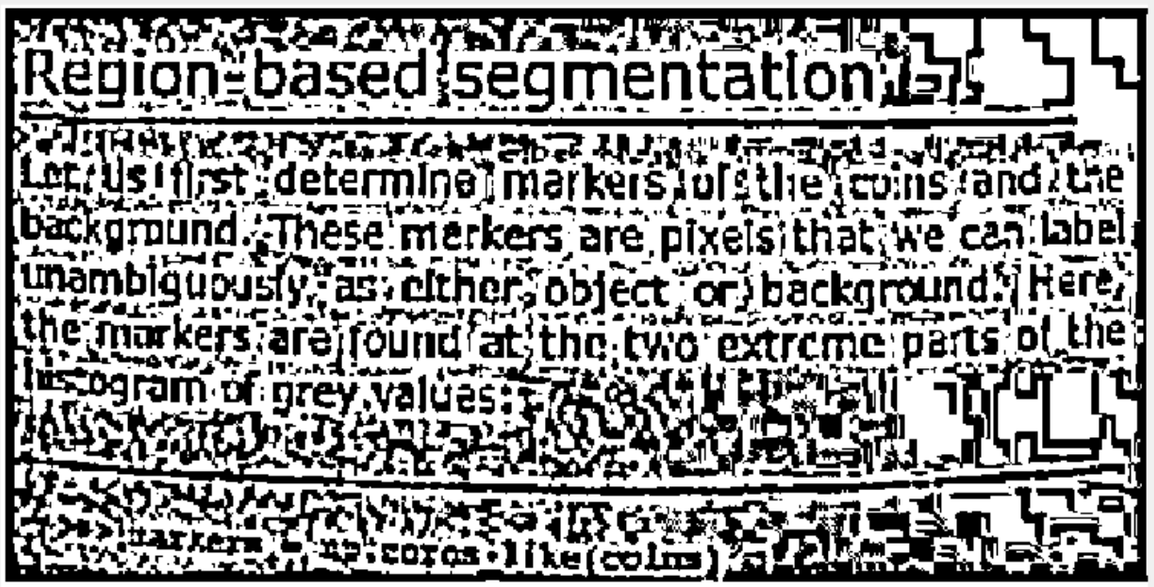


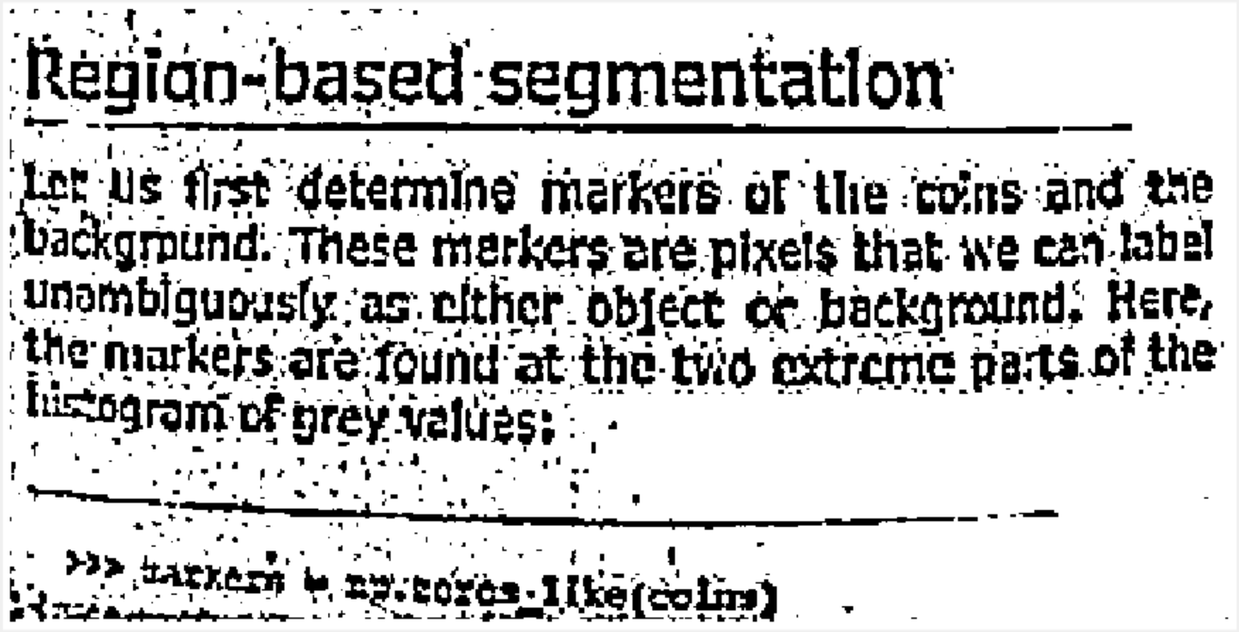
DEFAULT MEAN

DEFAULT MEDIAN

EXAMPLE

"PLAYING" FOR BETTER RESULTS:





MEAN: 9X9, 0.14

MEDIAN: 9X9, 0.2

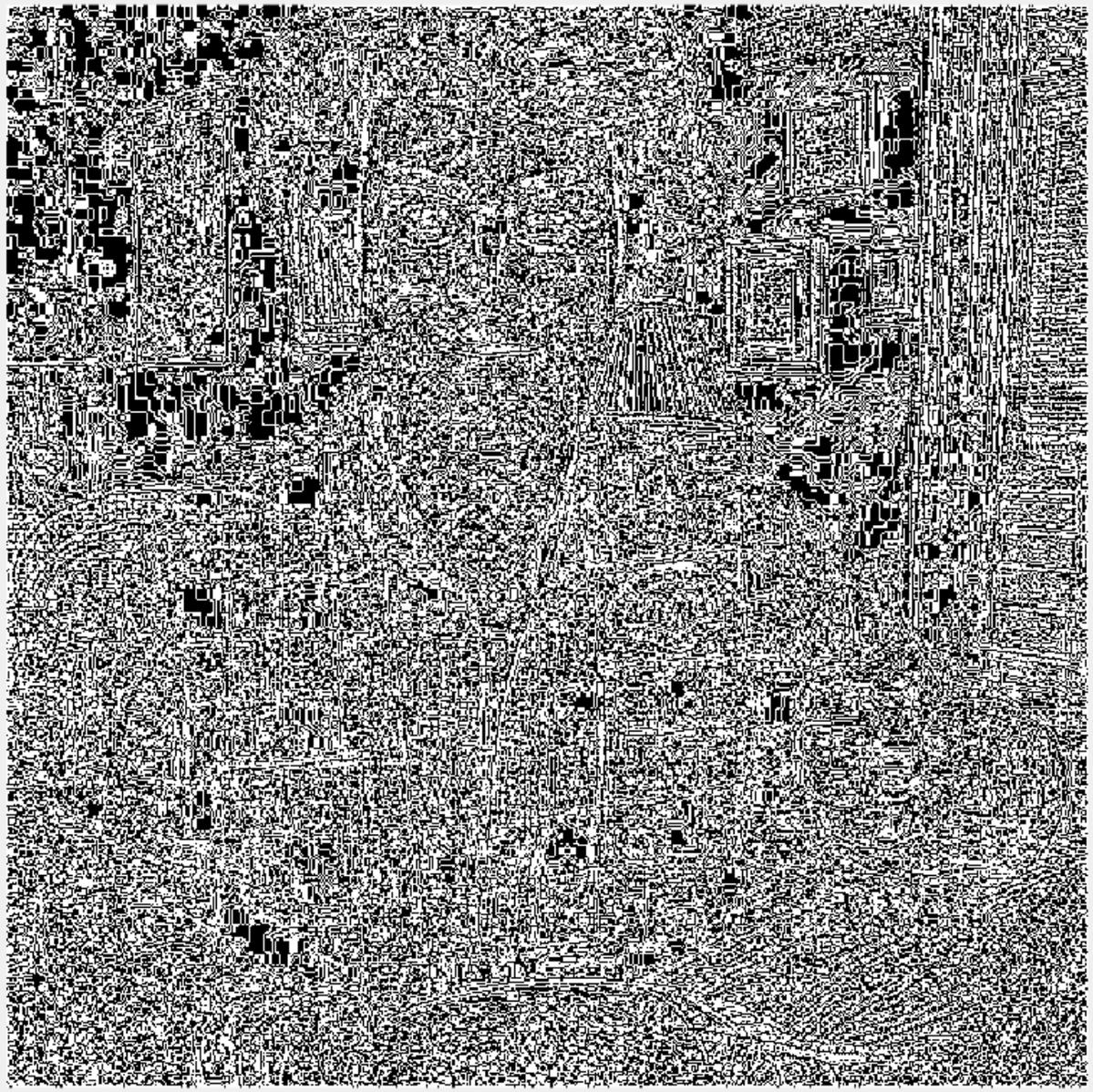
NIBLACK'S LOCALLY ADAPTIVE THRESHOLDING

- Window size, to split image into local windows
- Threshold of local threshold based on local mean and standard deviation
- Default constant "k" is suggested as -0.2
- Does not do well with large variance in illumination
- Optimum "k" hard to achieve
- Following images done with MEAN and MEDIAN

EXAMPLE 1

DEFAULT: 3X3, -0.2

DEFAULT MEAN



DEFAULT MEDIAN

EXAMPLE 1

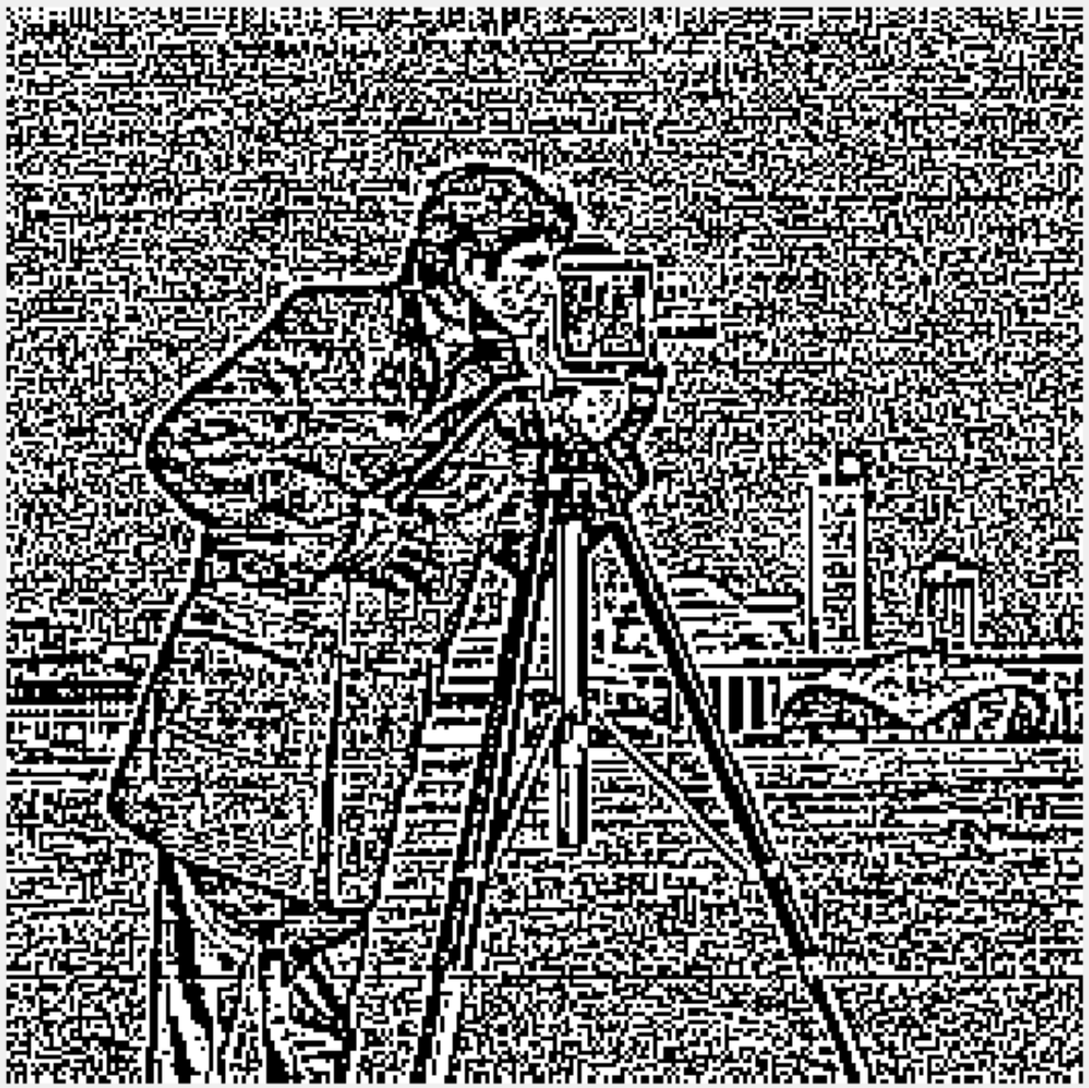
"PLAYING" FOR BETTER RESULTS:

MEAN: 9X9, -0.2

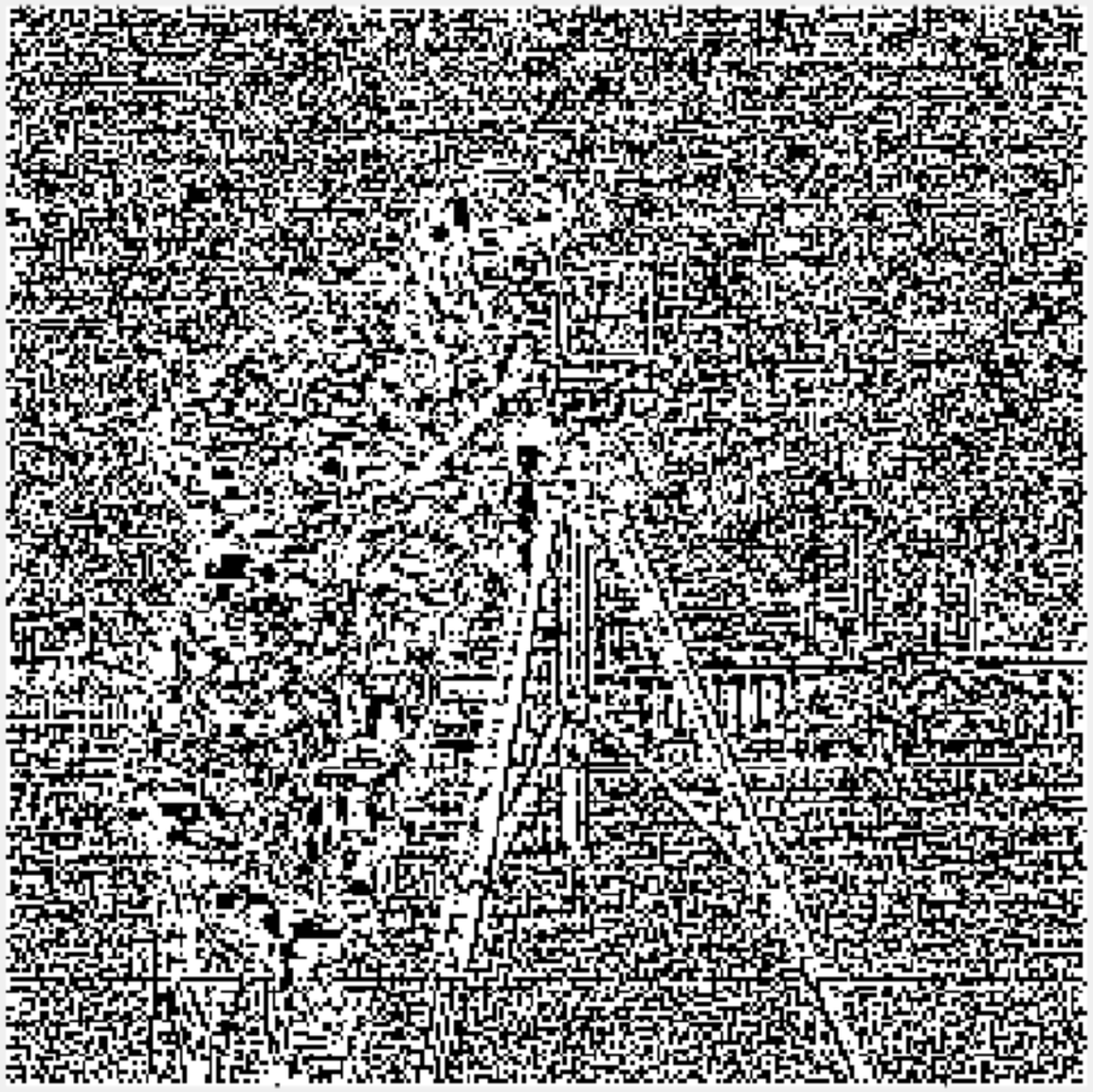
MEDIAN: 9X9, -0.99

EXAMPLE 2

DEFAULT: 3X3, -0.2



DEFAULT MEAN

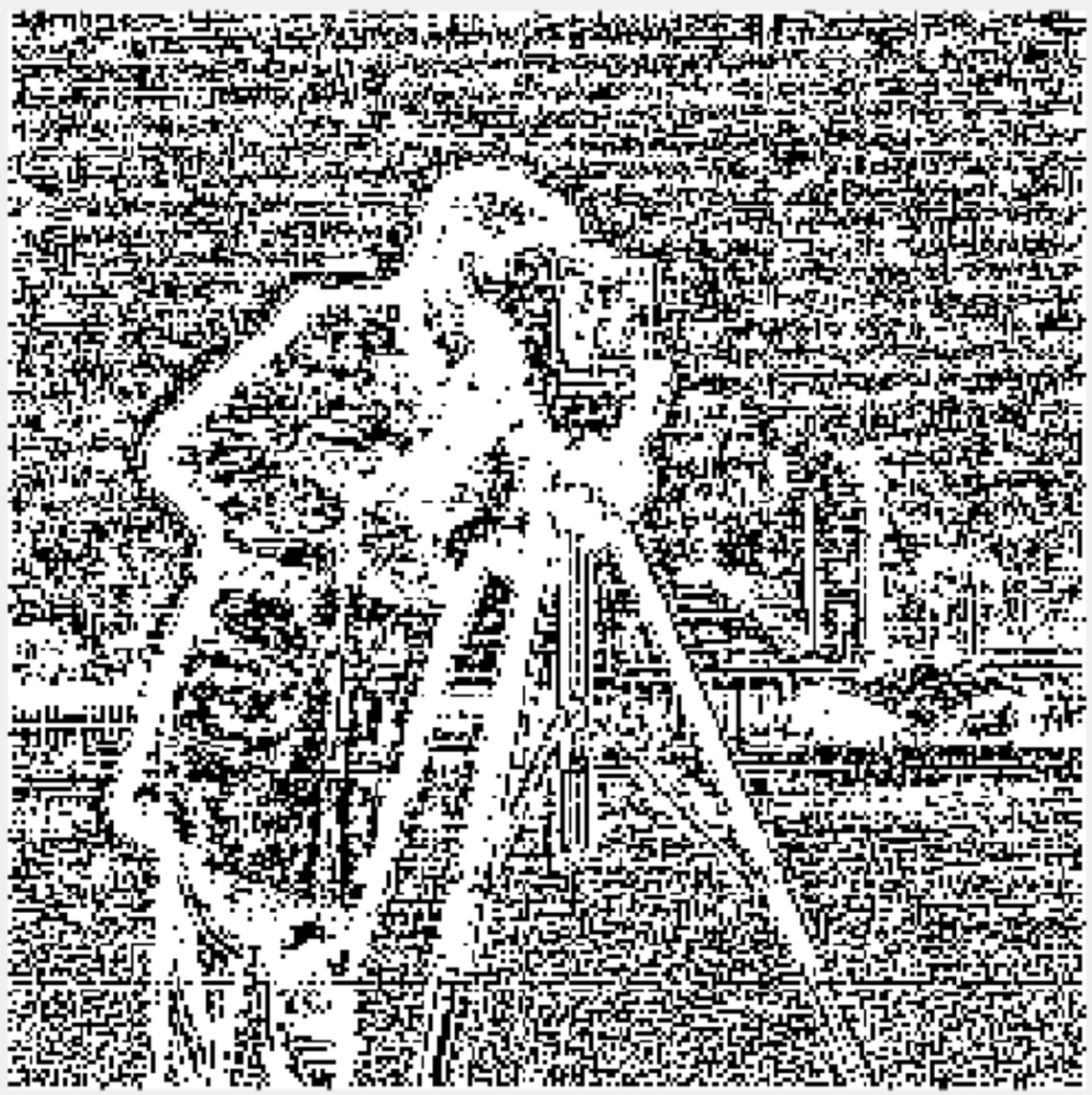


DEFAULT MEDIAN

EXAMPLE 2

"PLAYING" FOR BETTER RESULTS:

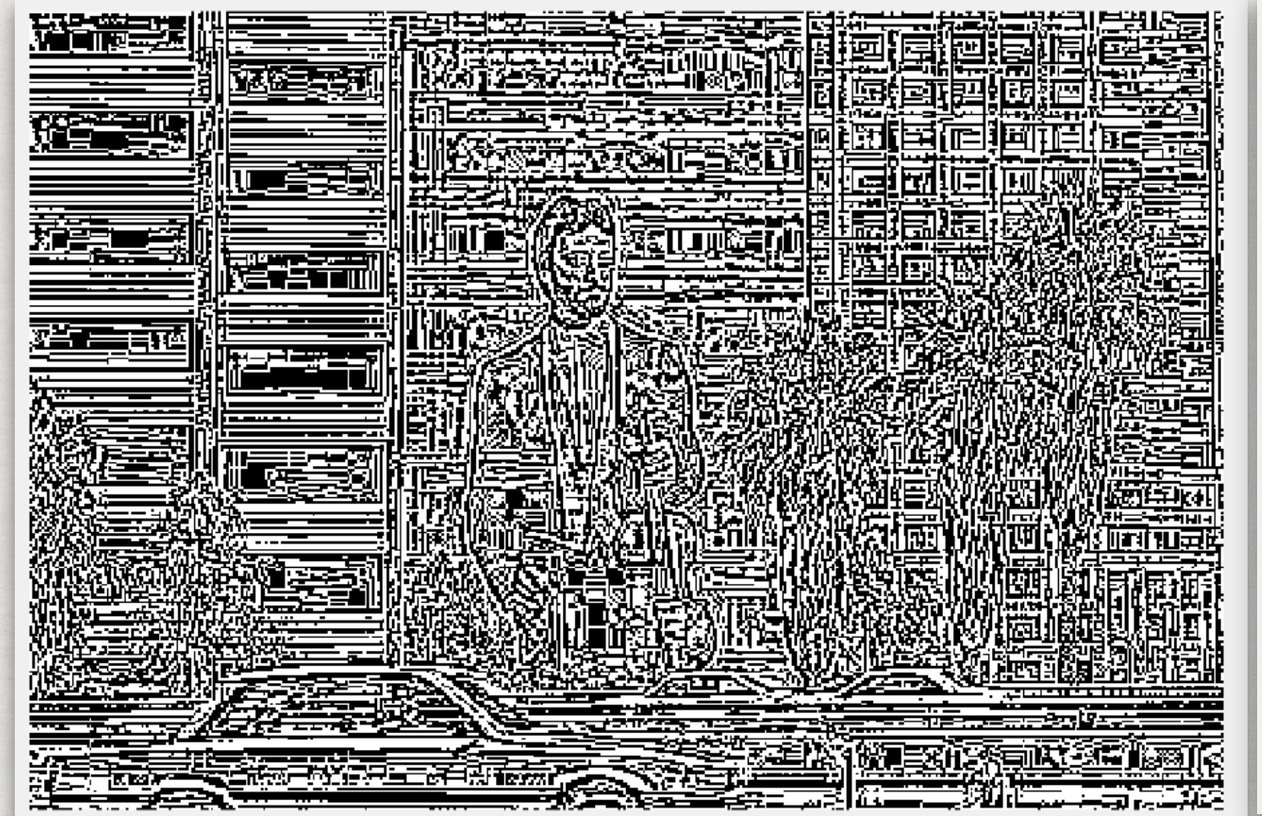
MEAN: 9X9, -0.2



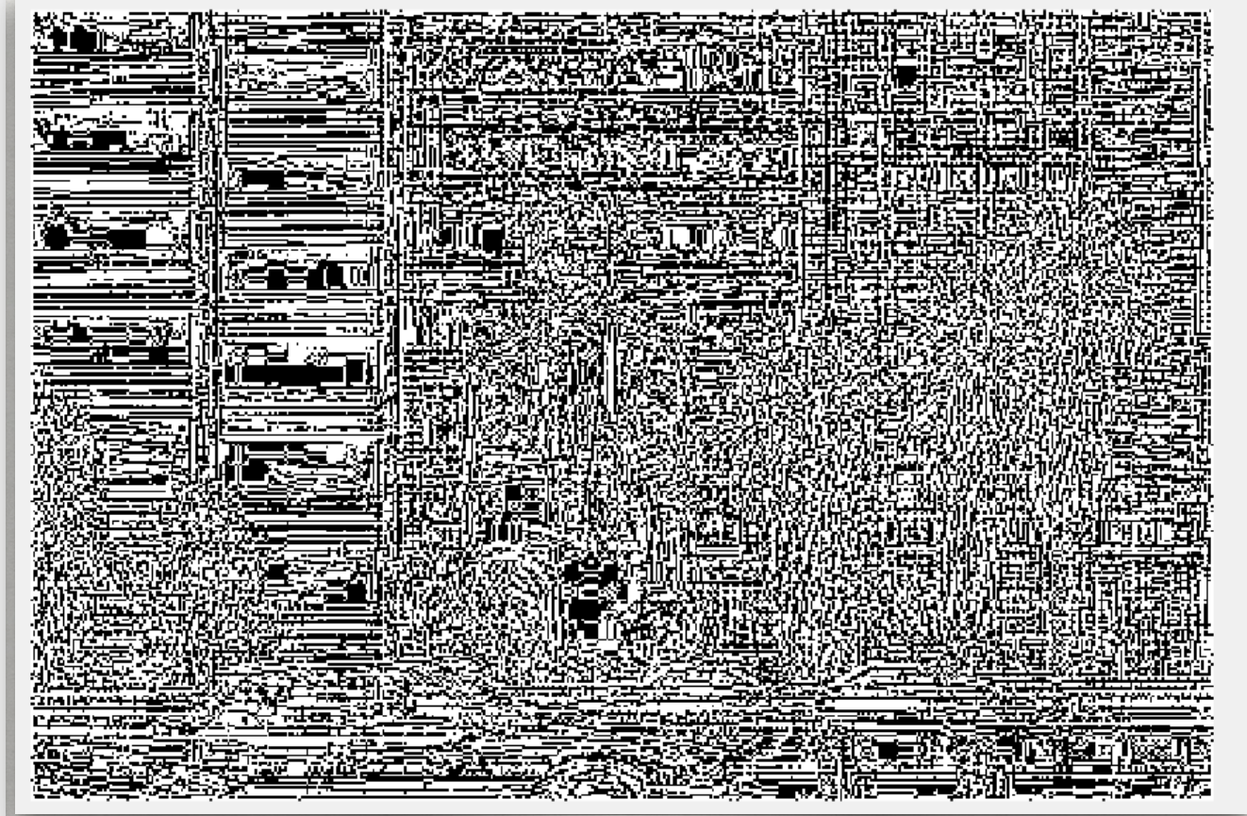
MEDIAN: 9X9, -0.9

EXAMPLE 3

DEFAULT: 3X3, -0.2



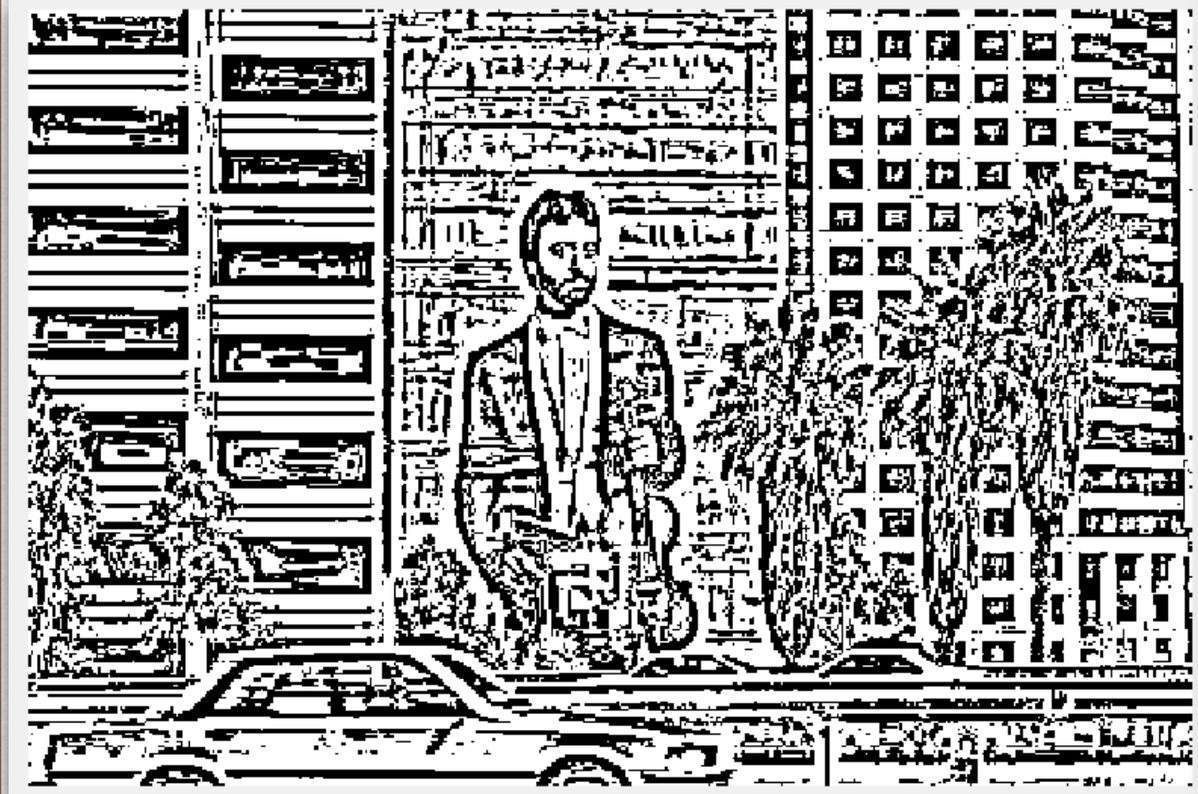
DEFAULT MEAN



DEFAULT MEDIAN

EXAMPLE 3

"PLAYING" FOR BETTER RESULTS:

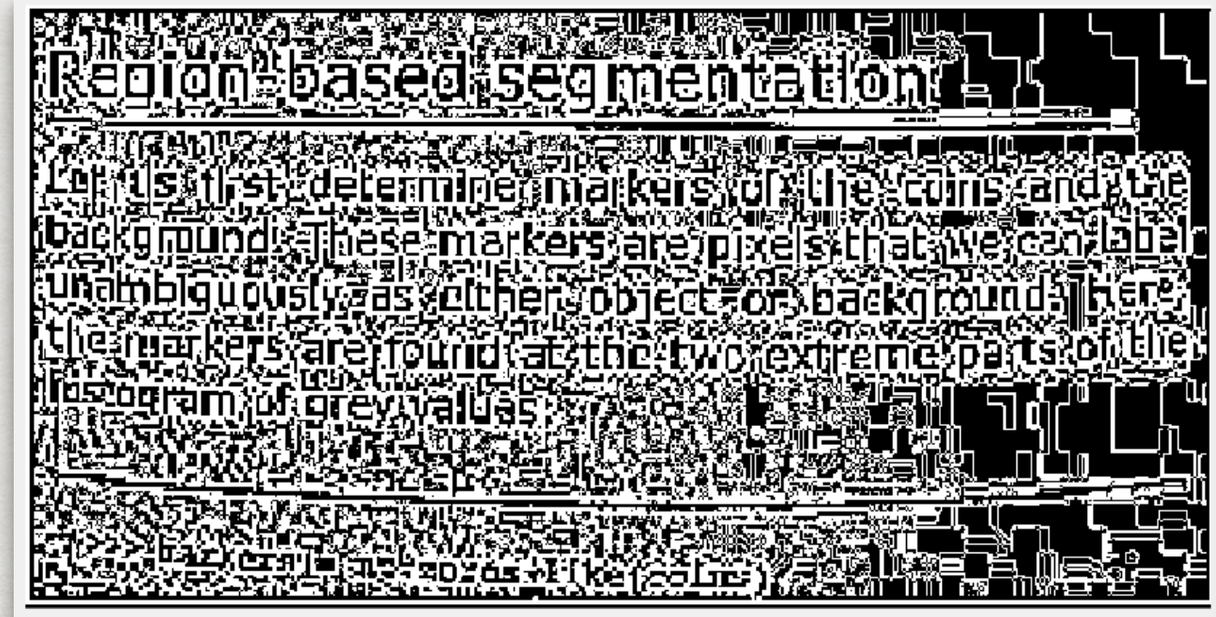


MEAN: 9X9, -0.30

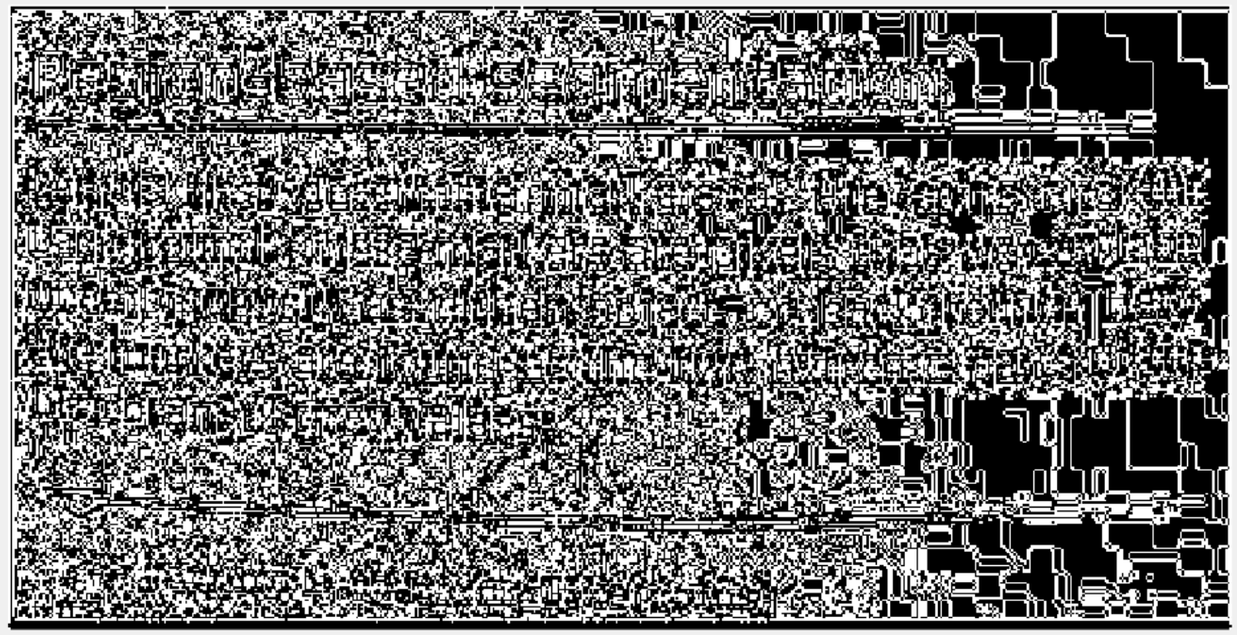
MEDIAN: 9X9, -0.90

EXAMPLE 4

DEFAULT: 3X3, -0.2



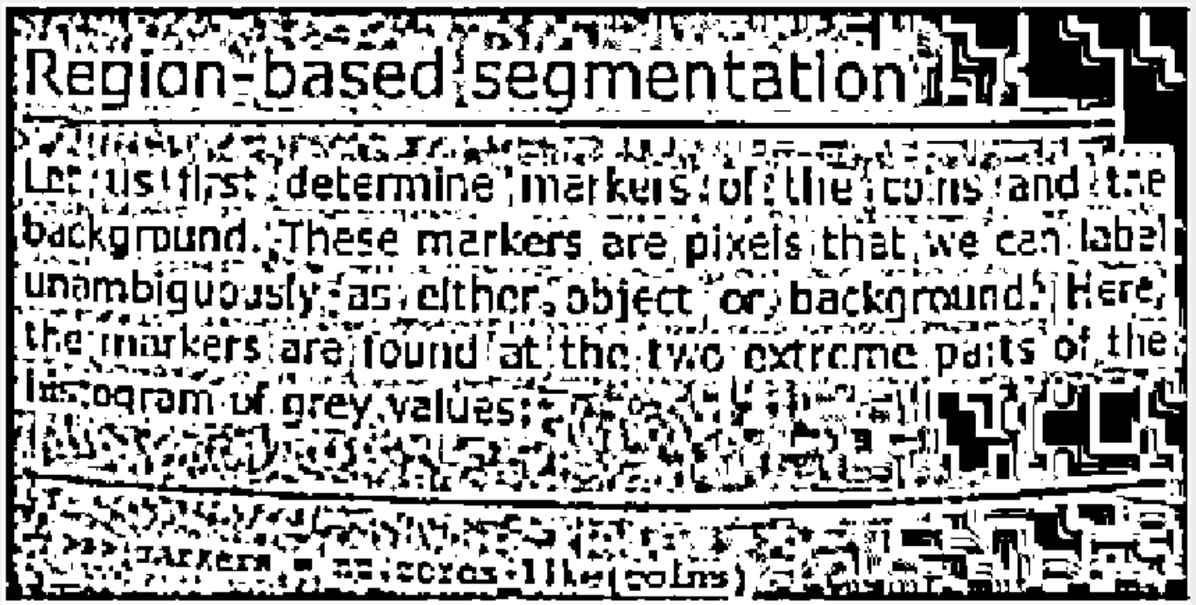
DEFAULT MEAN



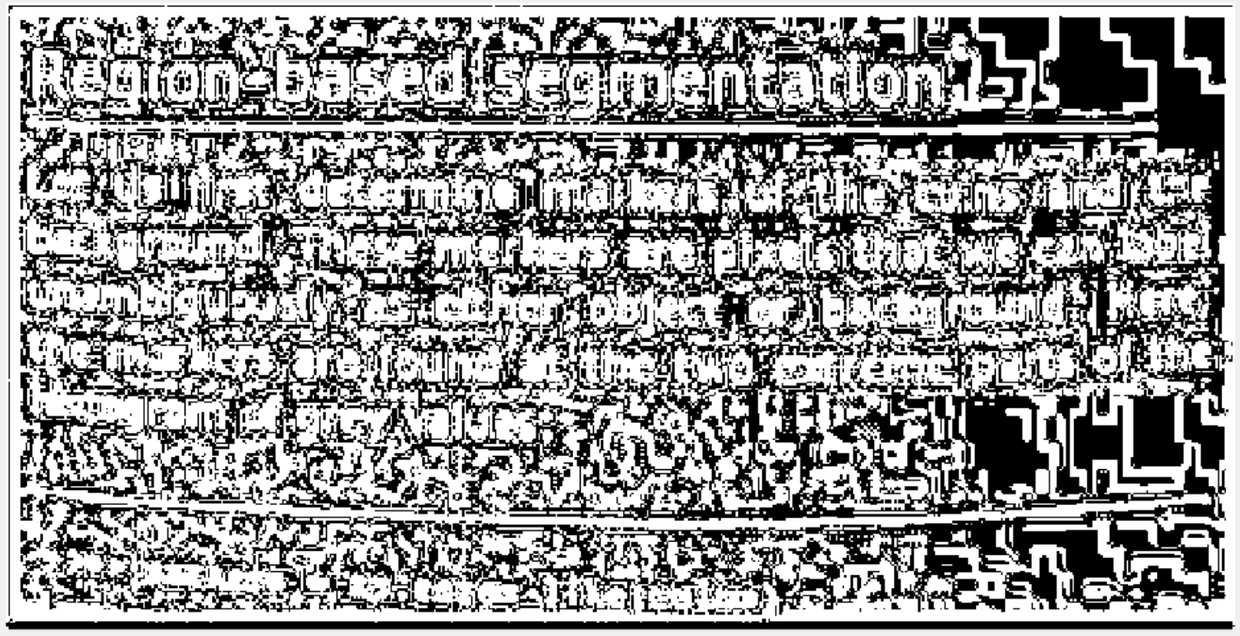
DEFAULT MEDIAN

EXAMPLE 4

"PLAYING" FOR BETTER RESULTS:



MEAN: 9X9, -0.35



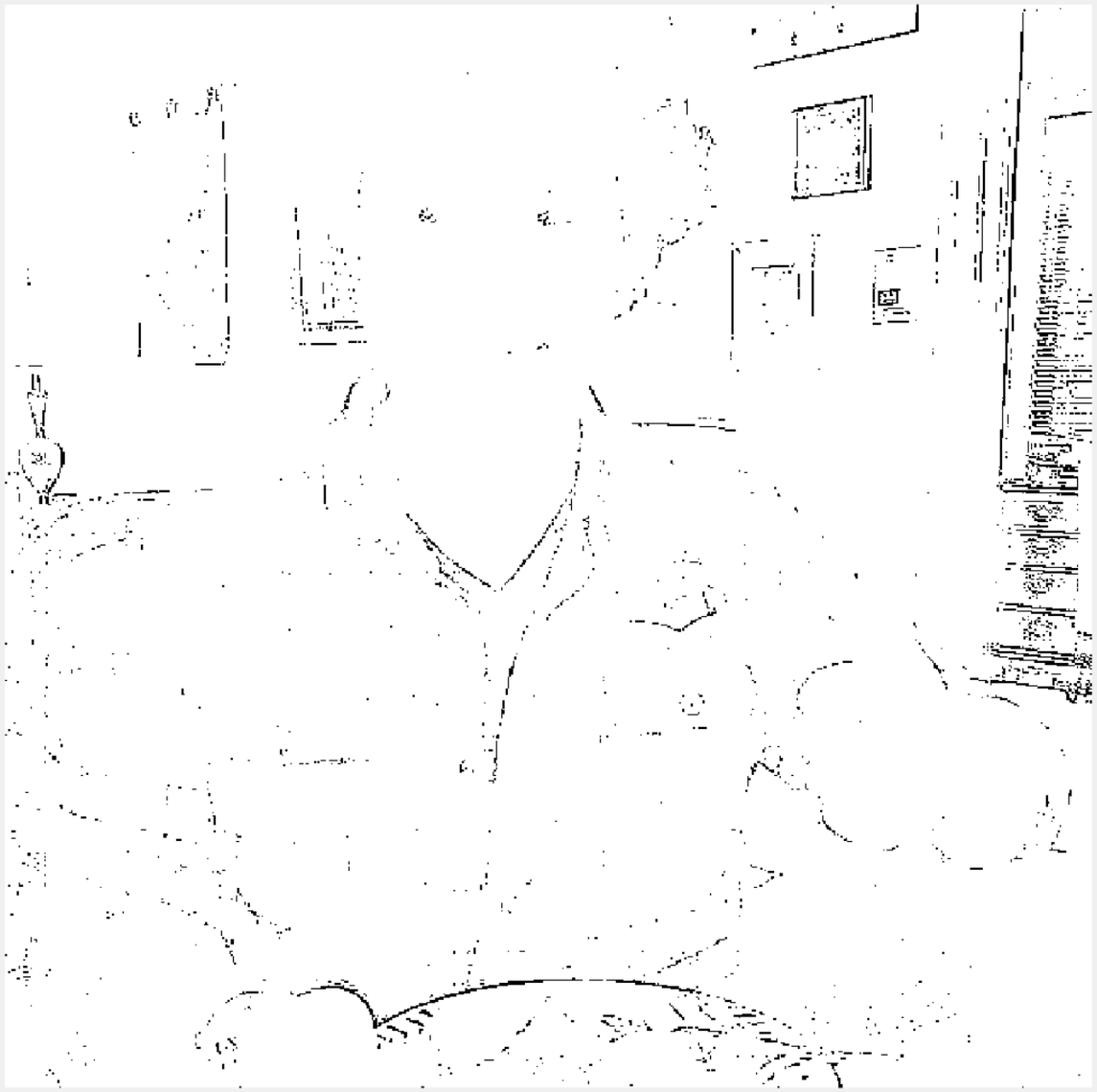
MEDIAN: 9X9, -0.80

SAUVOLA'S LOCALLY ADAPTIVE THRESHOLDING

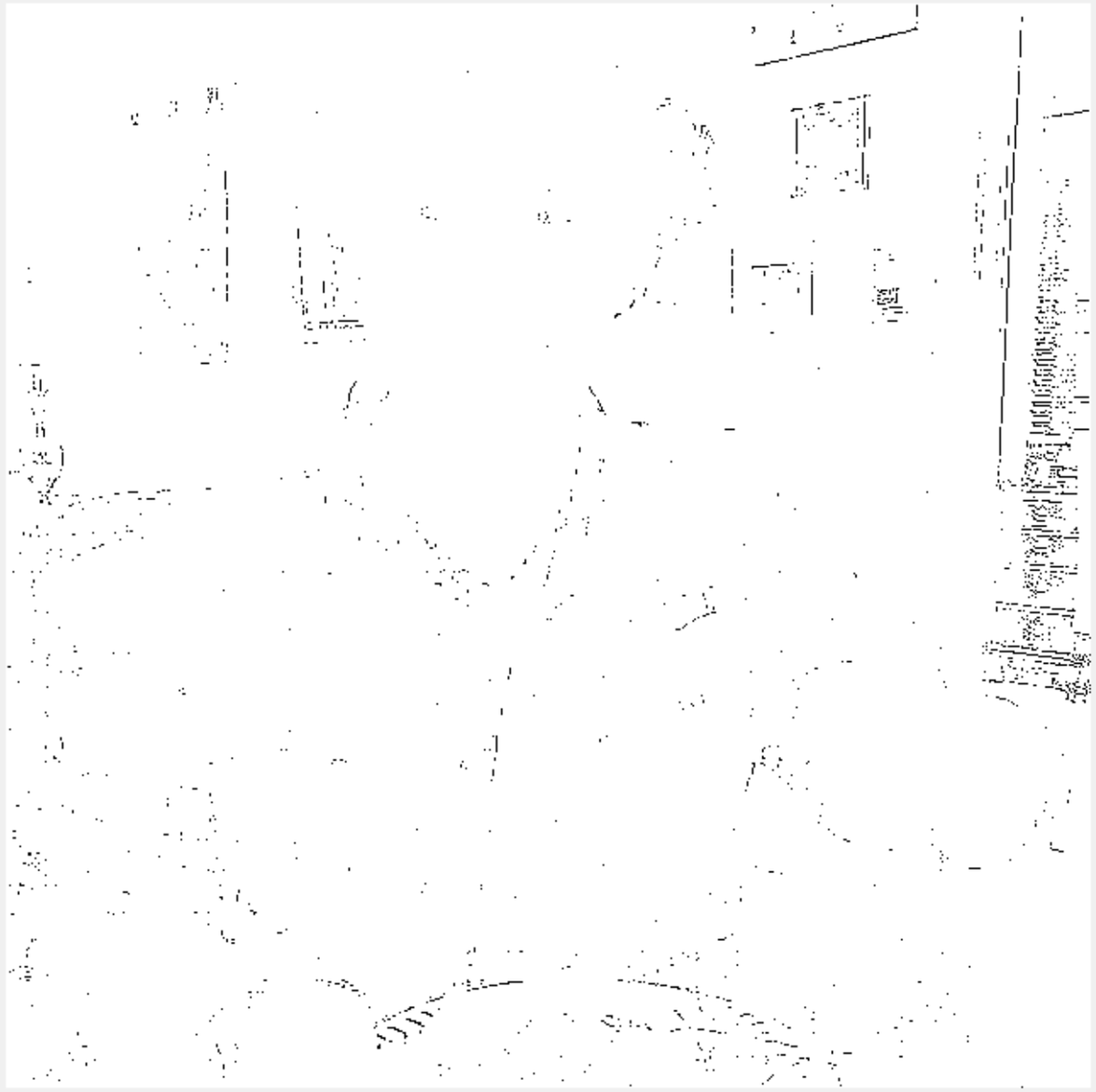
- Window size, to split image into local windows
- Threshold of local threshold based on local mean and standard deviation
- Default constant "k" is suggested between 0.2-0.5
- Does does better with variance in illumination
- "k" used to control the effect of standard deviation
- Following images done with MEAN and MEDIAN

EXAMPLE 1

DEFAULT: 3X3, 0.34



DEFAULT MEAN

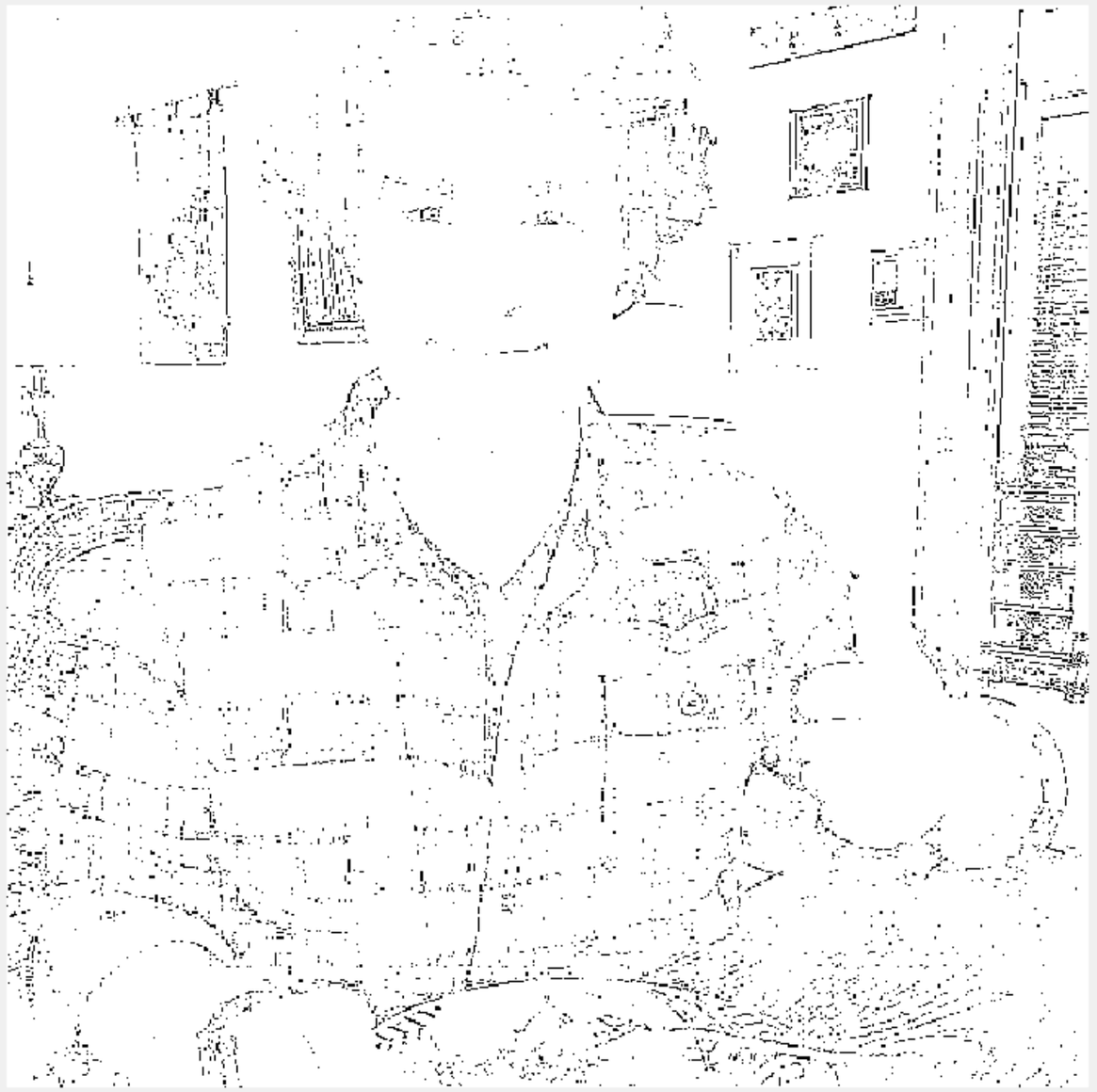


DEFAULT MEDIAN

EXAMPLE 1

"PLAYING" FOR BETTER RESULTS:

MEAN: 10X10, 0.9

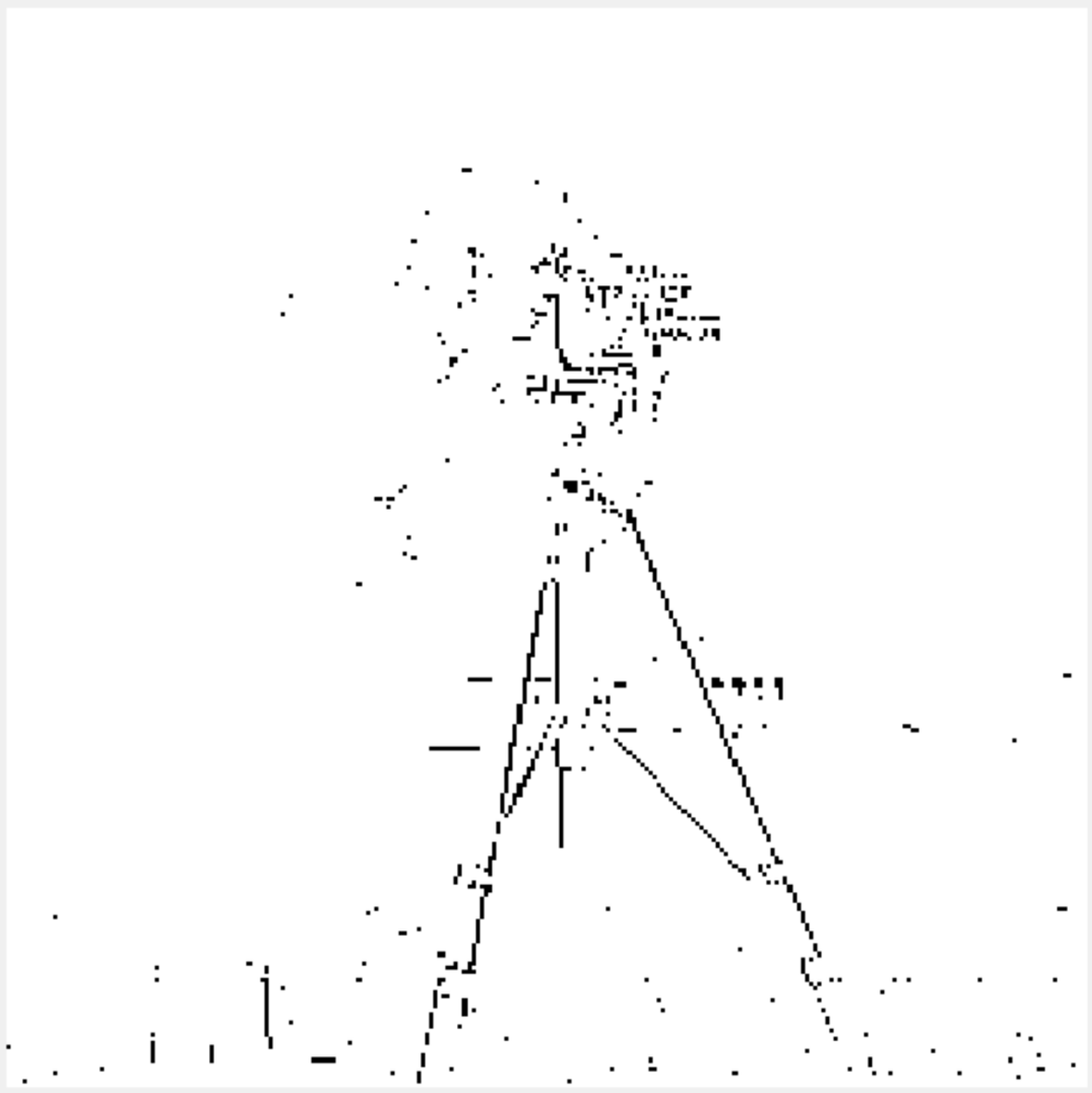


MEDIAN: 20X20, 0.15

EXAMPLE 2

DEFAULT: 3X3, 0.34

DEFAULT MEAN

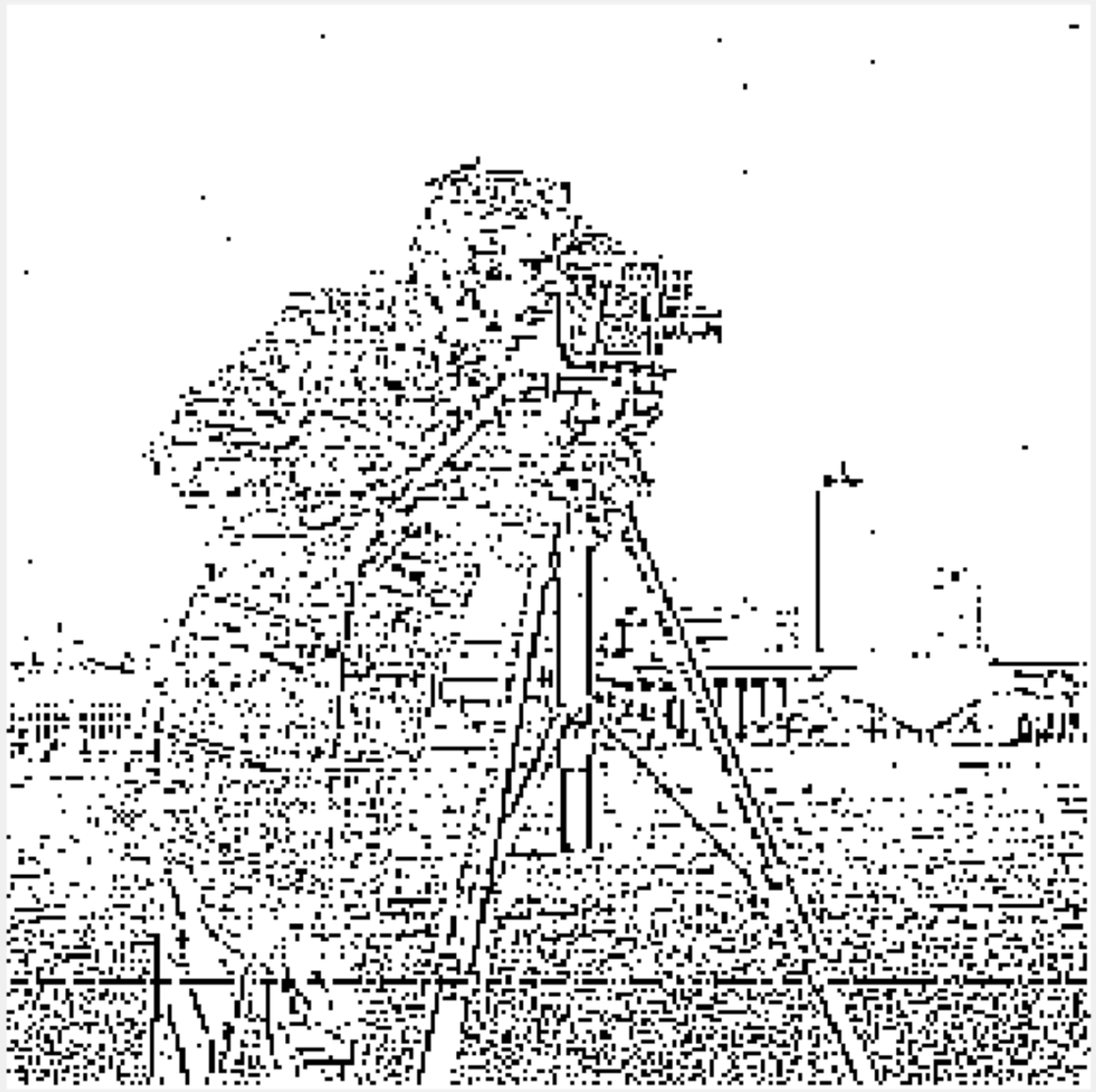


DEFAULT MEDIAN

EXAMPLE 2

"PLAYING" FOR BETTER RESULTS:

MEAN: 5X5, 0.50



MEDIAN: 9X9, 0.50

EXAMPLE 3

DEFAULT: 3X3, 0.34

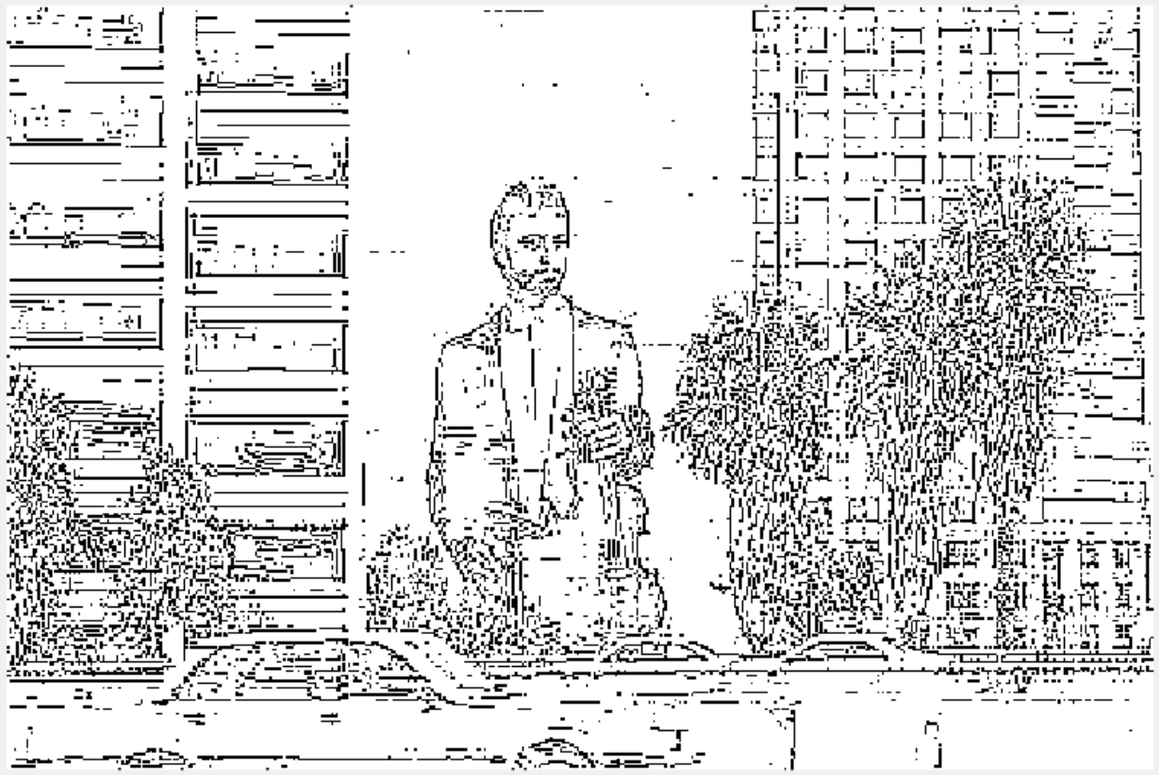
DEFAULT MEAN

DEFAULT MEDIAN

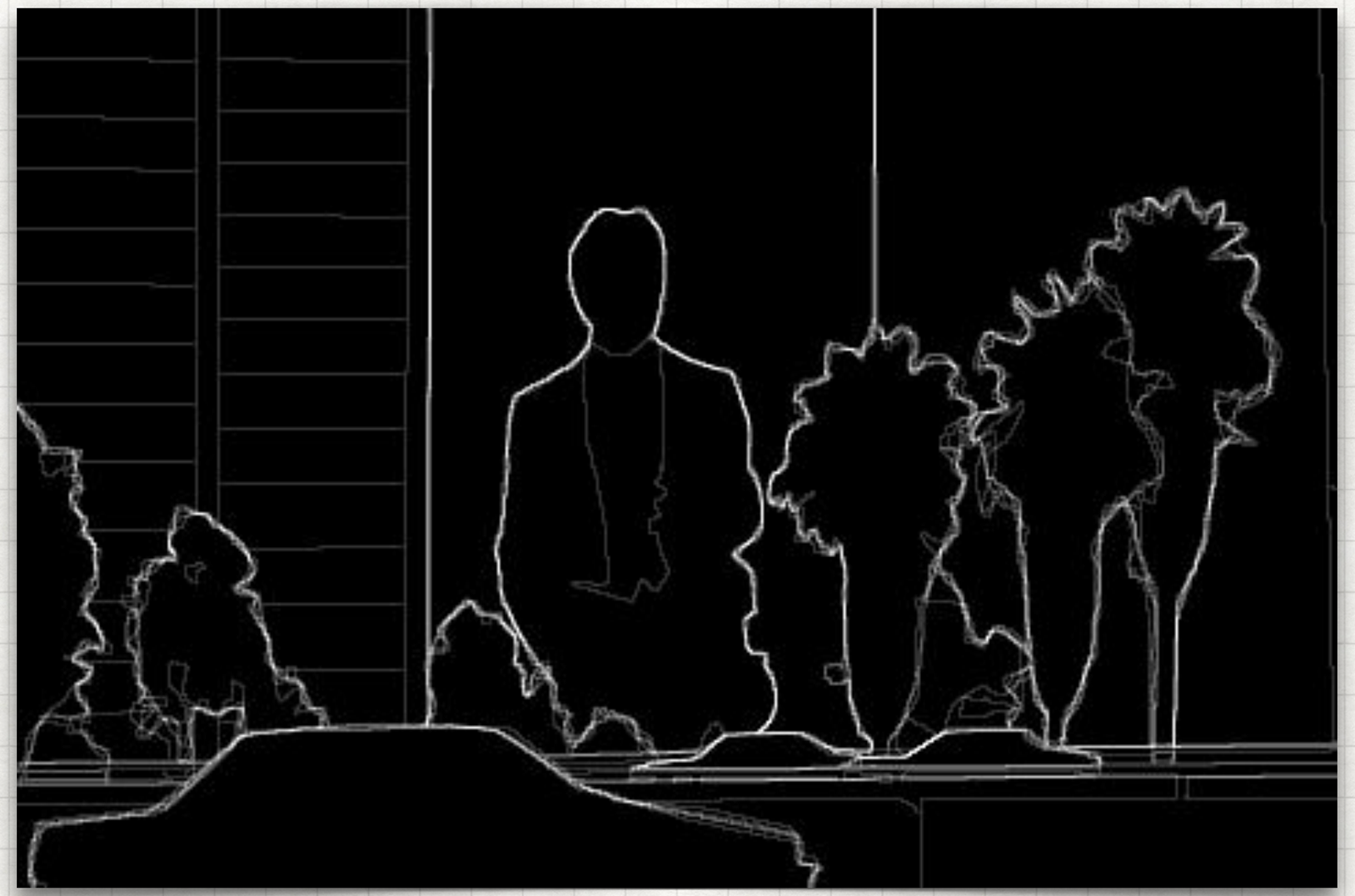
EXAMPLE 3

"PLAYING" FOR BETTER RESULTS:

MEAN: 7X7, 0.25

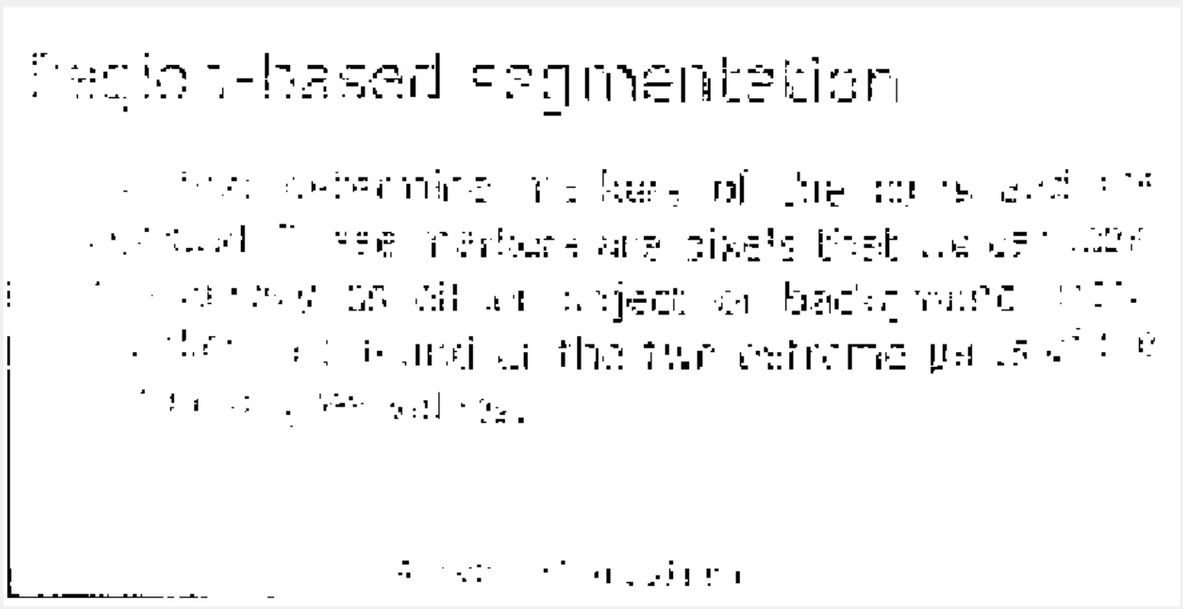


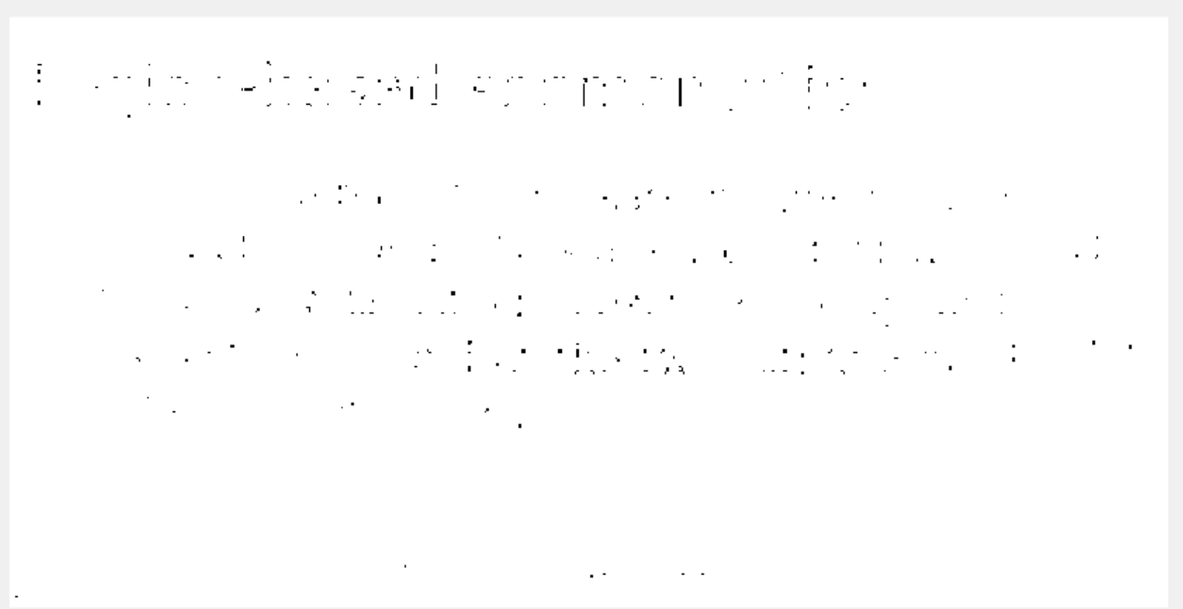
MEDIAN: 5X5, 0.10



EXAMPLE 4

DEFAULT: 3X3, 0.34



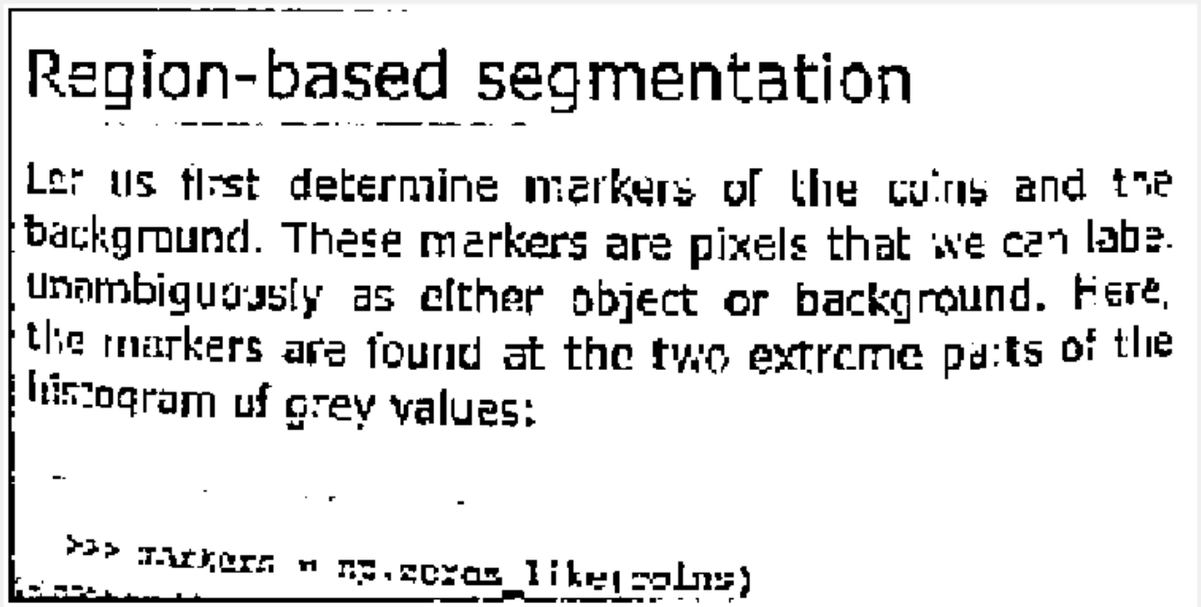


DEFAULT MEAN

DEFAULT MEDIAN

EXAMPLE 4

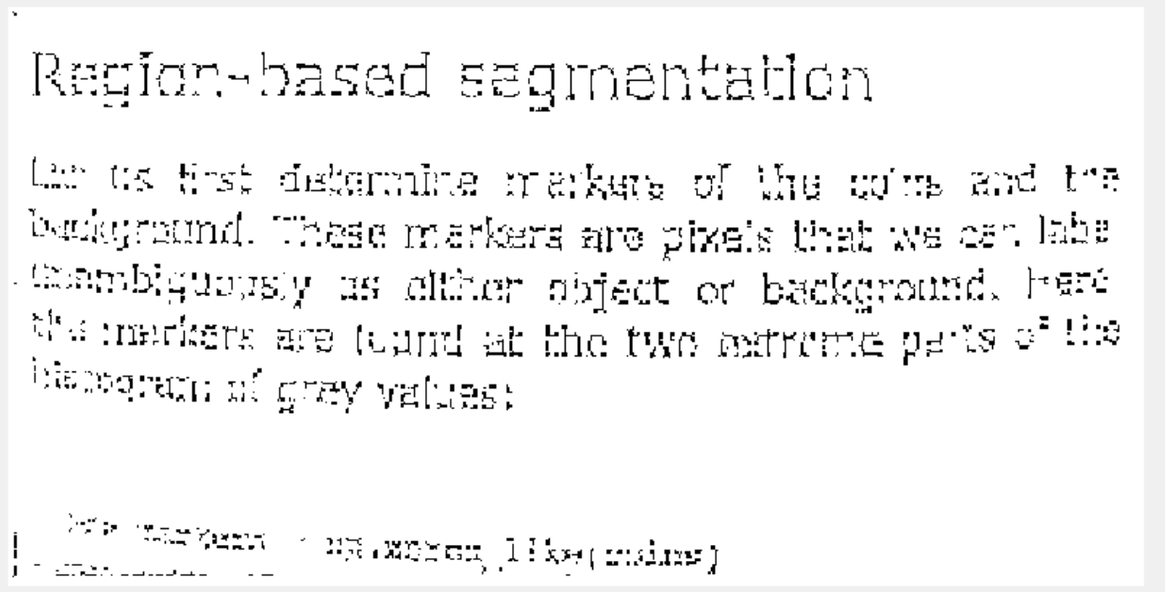
"PLAYING" FOR BETTER RESULTS:


Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

MEAN: 9X9, 0.30


Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

MEDIAN: 9X9, 0.30

THANK YOU