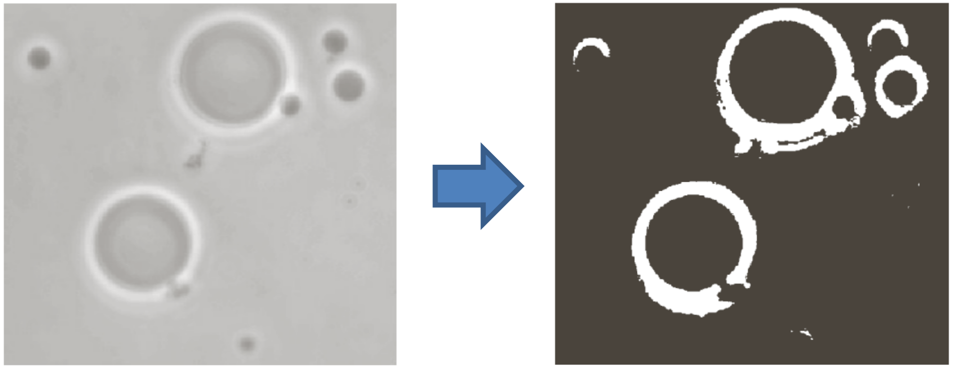
Image Segmentation

Sung Soo Hwang

Introduction

- What is image/video segmentation?
  - Process of partitioning a digital image into multiple regions
  - Application
    - Object classification

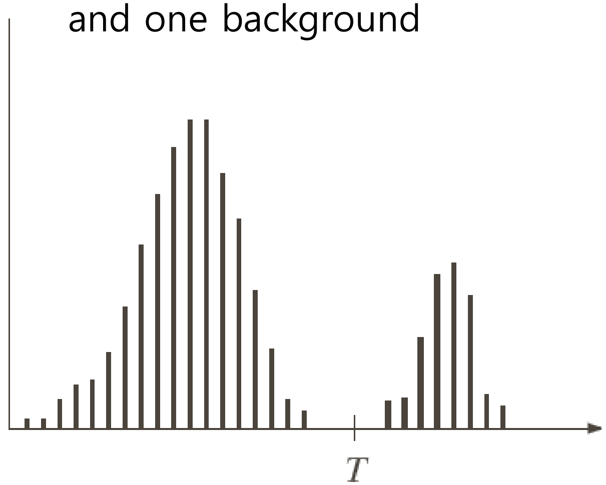


Introduction

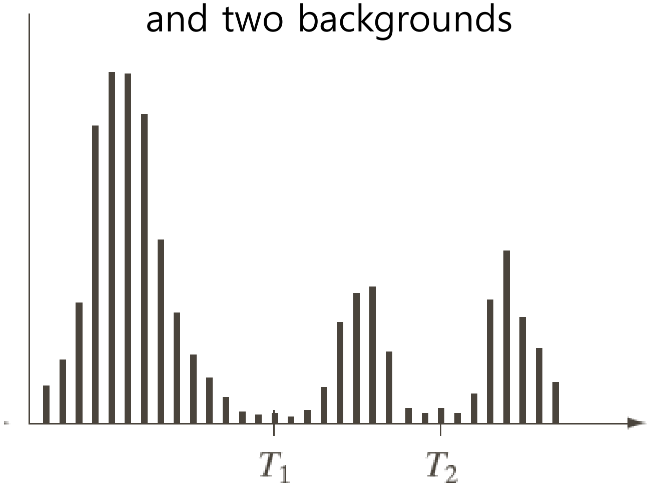
- What is image/video segmentation?
  - Input images are assumed to be gray-scale
    - Input: gray-scale image
    - Output: binary image (images with 0 and 255 (or 0 and 1) only)

Thresholding

- Basic concepts
  - Assumption
    - Intensity of background and object is different
    - Background and object are homogenous



$$g(x,y) = \begin{cases} 1 & if\ f(x,y) > T \\ 0 & otherwise \end{cases}$$

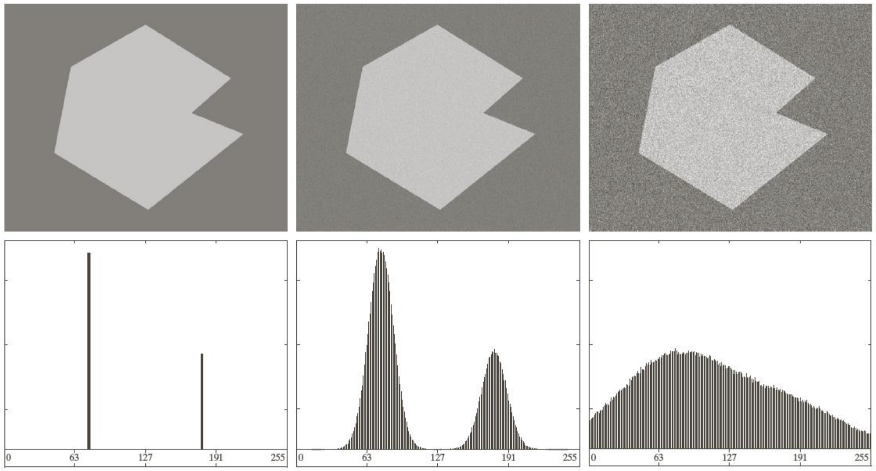


$$g(x,y) = \begin{cases} a & if\ f(x,y) > T_2 \\ b & if\ T_1 < f(x,y) \le T_2 \\ c & otherwise \end{cases}$$

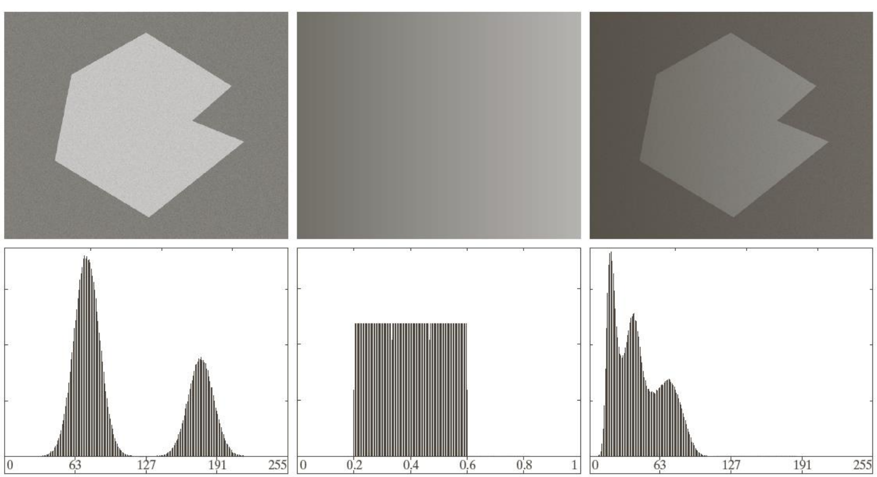
- Finding the proper threshold is important

Thresholding

- Challenges
  - Noise

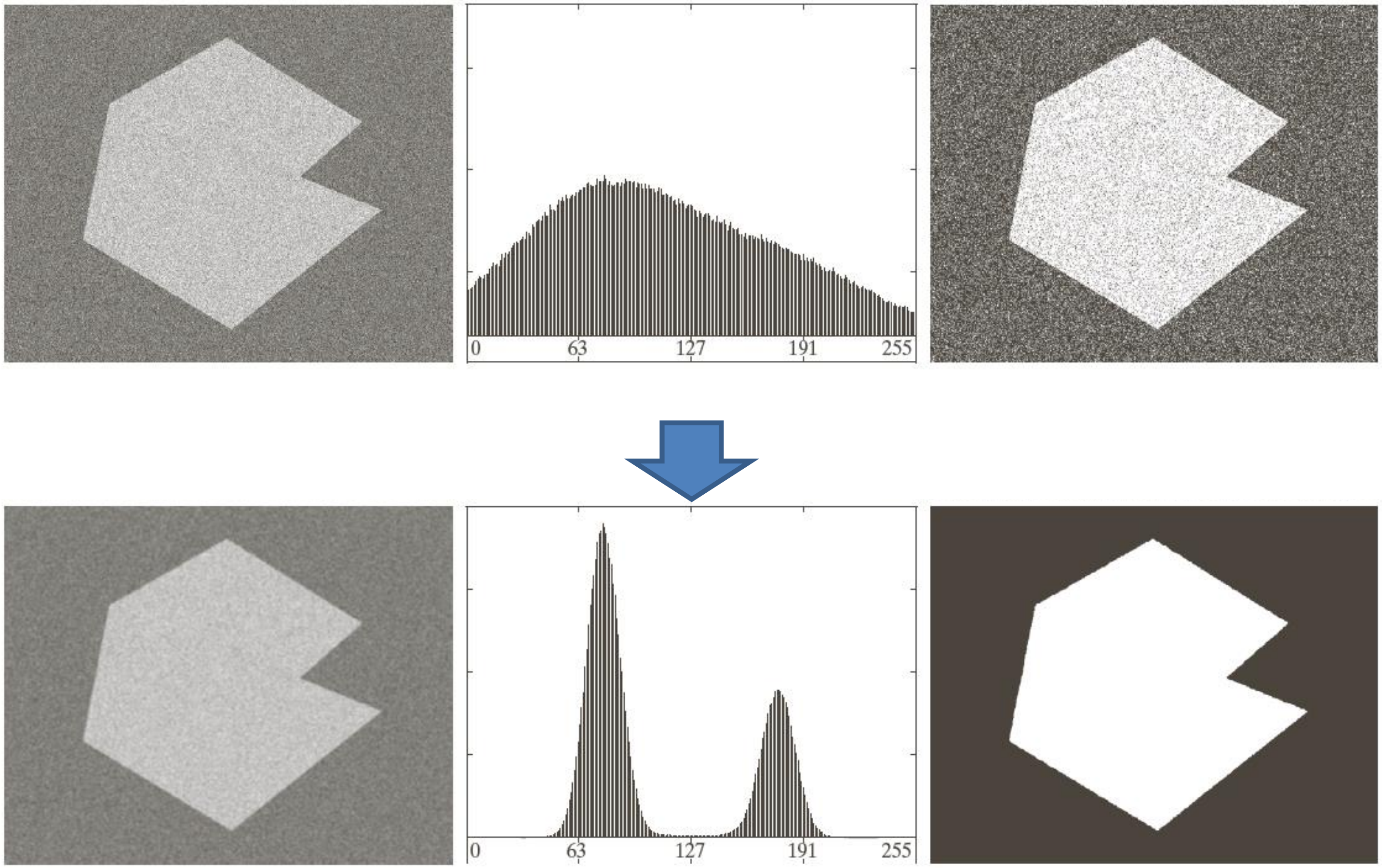


  - Illumination and reflectance



Thresholding

- Thresholding after applying smoothing



- By applying smoothing before thresholding, we may obtain the better result

Thresholding

- Global thresholding
  - Use same threshold for every pixel

- Local (adaptive) thresholding
  - Use different threshold for each pixel

Global Thresholding

- Basic method

  1.Select an initial estimate for the global threshold T

  2. Segment the image using T into two groups

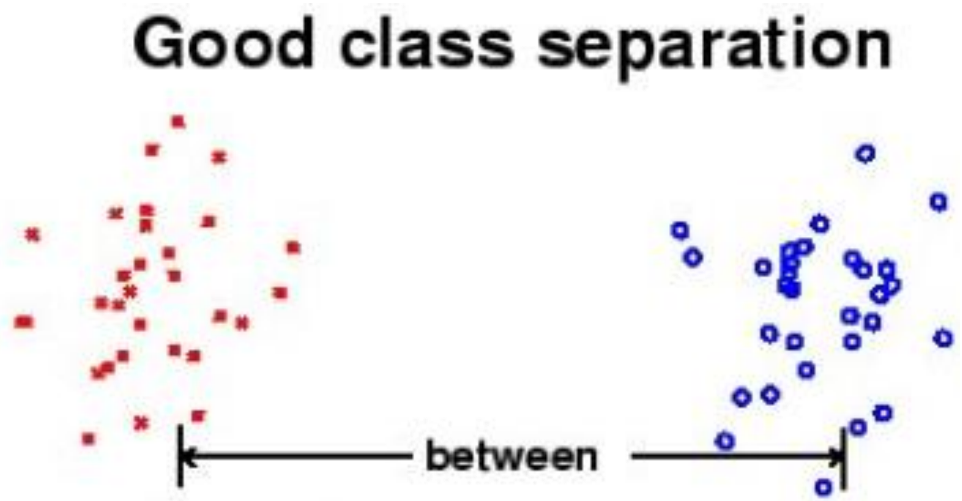
  3. compute the mean(m1,m2) for each group

  4. compute new threshold as T=0.5X(m1+m2)

  5. repeat step 2 through 4 until the difference between values of T in successive iterations is small

Global Thresholding

- Otsu's method
  - Concept
    - Well-thresholded classes should be distinct with respect to the intensity values of their pixels
    - Conversely, a threshold giving the best separation between classes would be the best threshold
    - It is based on computations performed on the histogram of an image

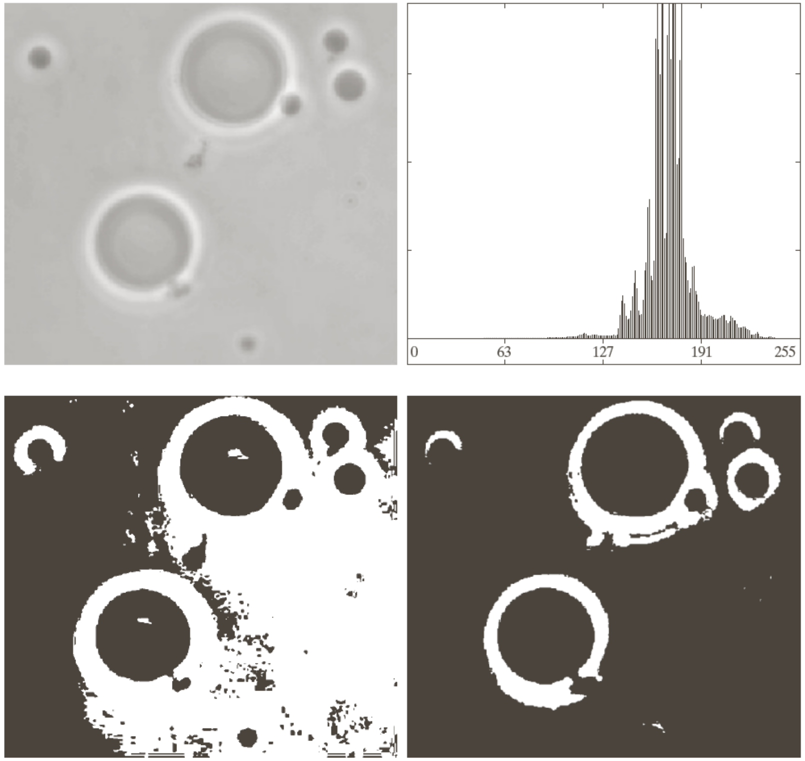


https://www.dtreg.com/solution/view/27

Global Thresholding

- Otsu's method

1. Compute the normalized histogram
2. For each threshold k, compute between-class variance $\sigma^2{}_B$
3. Obtain the Otsu threshold k for which $\sigma^2{}_B$ is maximized



Local(Adaptive) Thresholding

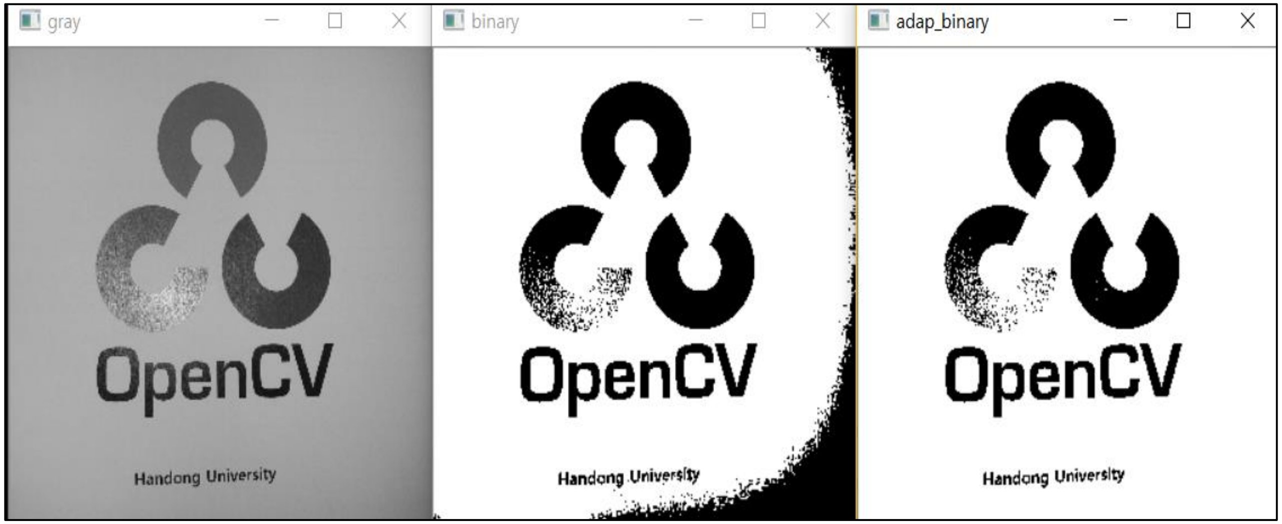
- Set a threshold for each point depending on the intensity distributions of adjacent pixels

ADAPTIVE_THRESH_MEAN_C :

$T(x, y) = mean\ of\ the\ blocksize \times blocksize\ neighborhood\ of\ (x, y) - C$

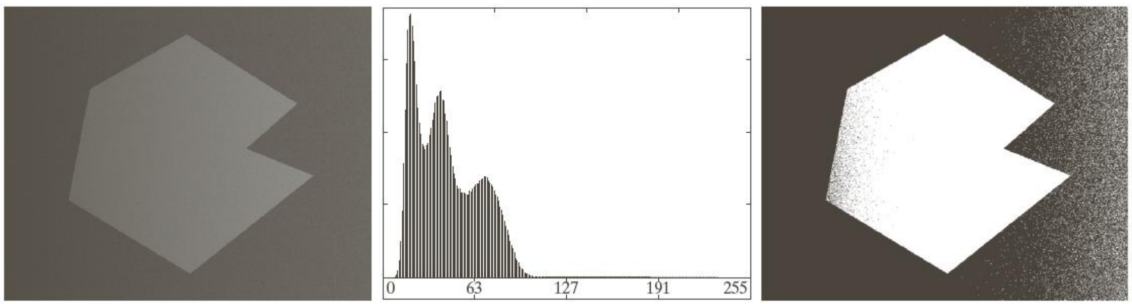
ADAPTIVE_THRESH_GAUSSIAN_C :

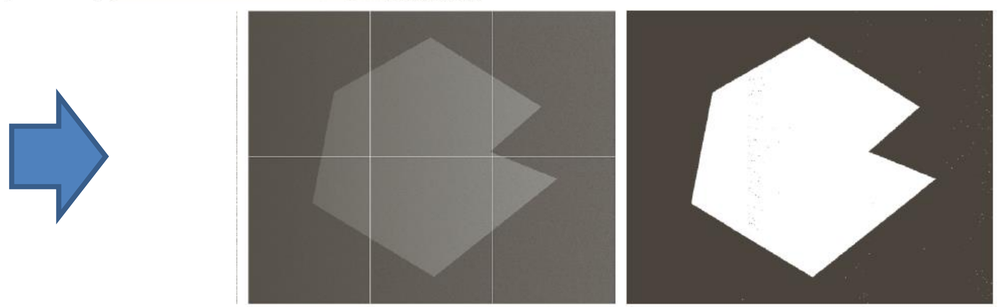
$T(x, y)$
$= a\ weighted\ sum(cross - correlation\ with\ a\ Gaussian\ windonw)\ of\ the\ blocksize$
$\times blocksize\ neighborhood\ of\ (x, y) - C$

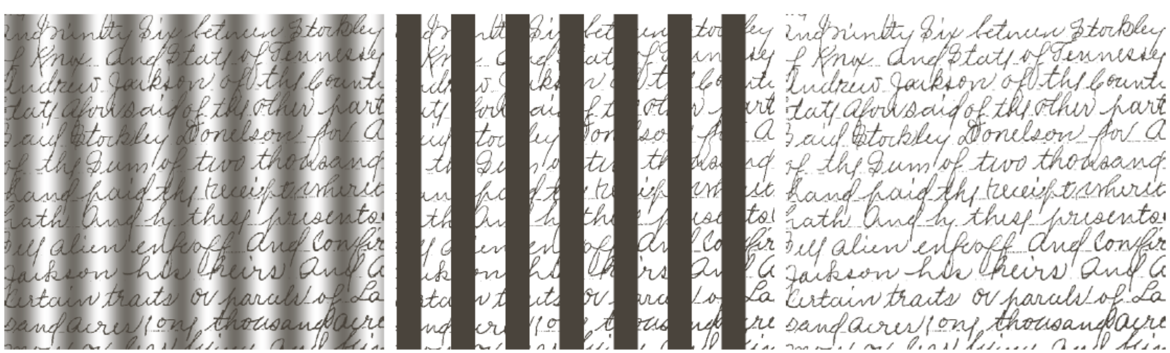


Local(Adaptive) Thresholding

- Set a threshold for each point depending on the intensity distributions of adjacent pixels
  - Image partitioning







Global Thresholding

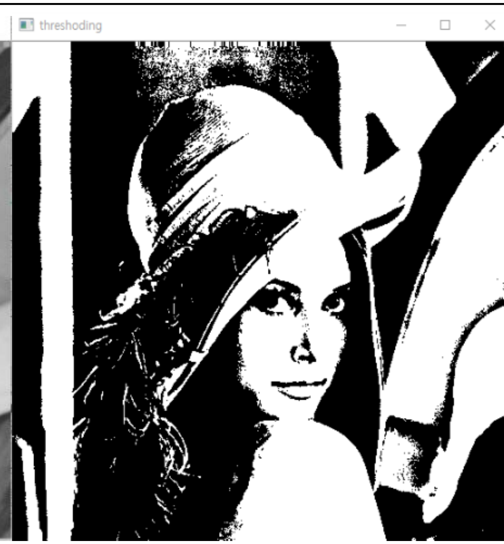
- Basic method
  - Example code

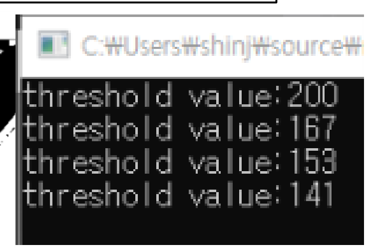
```
int main() {
	Mat image, thresh;
	int thresh_T, low_cnt, high_cnt, low_sum, high_sum, i, j, th;

	thresh_T = 200;
	th = 10;
	low_cnt = high_cnt = low_sum = high_sum = 0;

	image = imread("lena.png", 0);
	cout << "threshold value:" << thresh_T << endl;

	while (1) {
		for (j = 0; j < image.rows; j++) {
			for (i = 0; i < image.cols; i++) {
				if (image.at<uchar>(j, i) < thresh_T) {
					low_sum += image.at<uchar>(j, i);
					low_cnt++;
				}
				else {
					high_sum += image.at<uchar>(j, i);
					high_cnt++;
				}
			}
		}
```

Global Thresholding

- Basic method
  - Example code

```
            if (abs(thresh_T - (low_sum / low_cnt + high_sum / high_cnt) / 2.0f) < th) {
                break;
            }
            else {
                thresh_T = (low_sum / low_cnt + high_sum / high_cnt) / 2.0f;
                cout << "threshold value:" << thresh_T << endl;
                low_cnt = high_cnt = low_sum = high_sum = 0;
            }
        }
        threshold(image, thresh, thresh_T, 255, THRESH_BINARY);

        imshow("Input image", image);
        imshow("thresholding", thresh);
        waitKey(0);
}
```





Global Thresholding

- Otsu's algorithm
  - Example code

```
int main() {
	Mat image, result;
	image = imread("lena.png", 0);
	threshold(image, result, 0, 255, THRESH_BINARY | THRESH_OTSU);
	imshow("Input image", image);
	imshow("result", result);

	waitKey(0);
}
```

Local(Adaptive) Thresholding

- Set a threshold for each point depending on the intensity distributions of adjacent pixels
  - Example code

```
int main() {
        Mat image, binary, adaptive_binary;
        image = imread("opencv.jpg", 0);

        threshold(image, binary, 150, 255, THRESH_BINARY);
        adaptiveThreshold(image, adaptive_binary, 255, ADAPTIVE_THRESH_MEAN_C, THRESH_BINARY, 85, 15);

        imshow("Input image", image);
        imshow("binary", binary);
        imshow("adaptive binary", adaptive_binary);
        waitKey(0);
}
```