

COMP9517

Assignment1

Cong Cong z3414050



Stage1: Image filtering and thresholding

1.1

For global thresholding, I have used **threshold value** and same maximum value 255:

- Binary thresholding
 - If the pixel value is larger than **threshold value**, it will be set to 255 otherwise it will set to 0.
- Truncate thresholding
 - If the pixel value is larger than **threshold value**, it will be changed to **threshold value** otherwise it will not be changed.
- Threshold to Zero
 - If the pixel value is larger than **threshold value**, it will be set to 0 otherwise it will not be changed.

Though, there are three different methods of global thresholding, basically, Truncate and to zero have the similar idea as Binary thresholding.

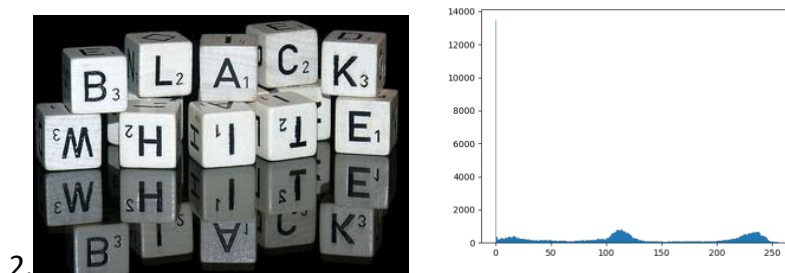
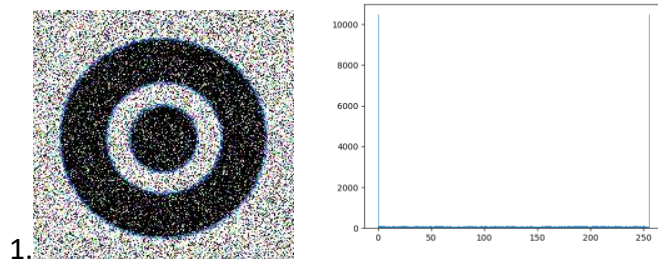
For Otsu thresholding, the thresholding value is not explicitly provided, it is automatically calculated using Otsu algorithm.

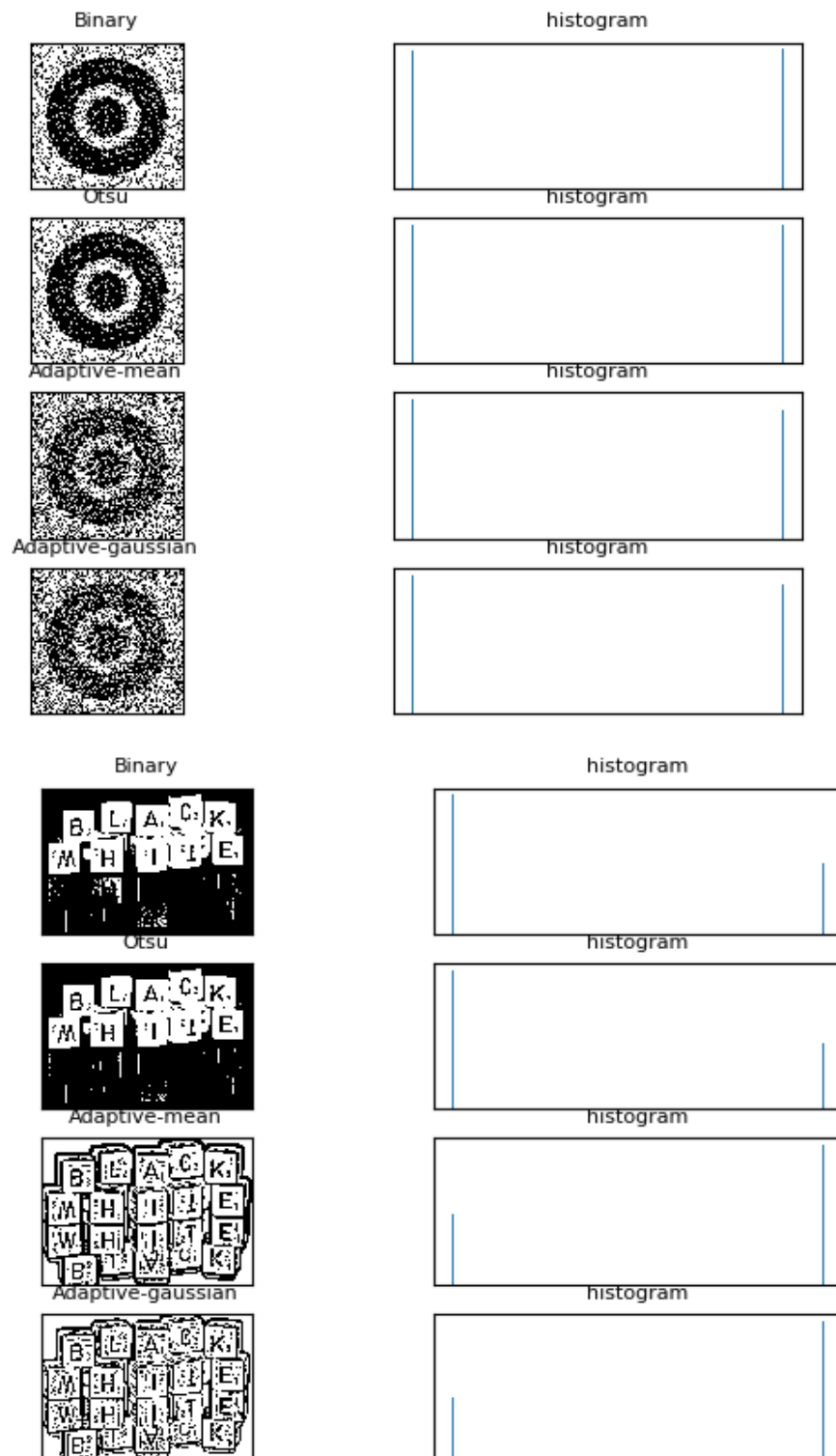
For adaptive thresholding, the threshold value is decided for different regions of the image, as a result of adaptive thresholding, different regions of the same image can end up having different threshold value for different regions:

- Adaptive threshold:
 - Mean: threshold value is the mean of the selected region
 - Gaussian: threshold value is the weighted sum of the selected region

In our case, I have tried different value of the pixel neighbourhood size and found out that size=11 with subtracted constant equals to 2 gives the best results for adaptive threshold (mean/ Gaussian).

Original images and histograms:

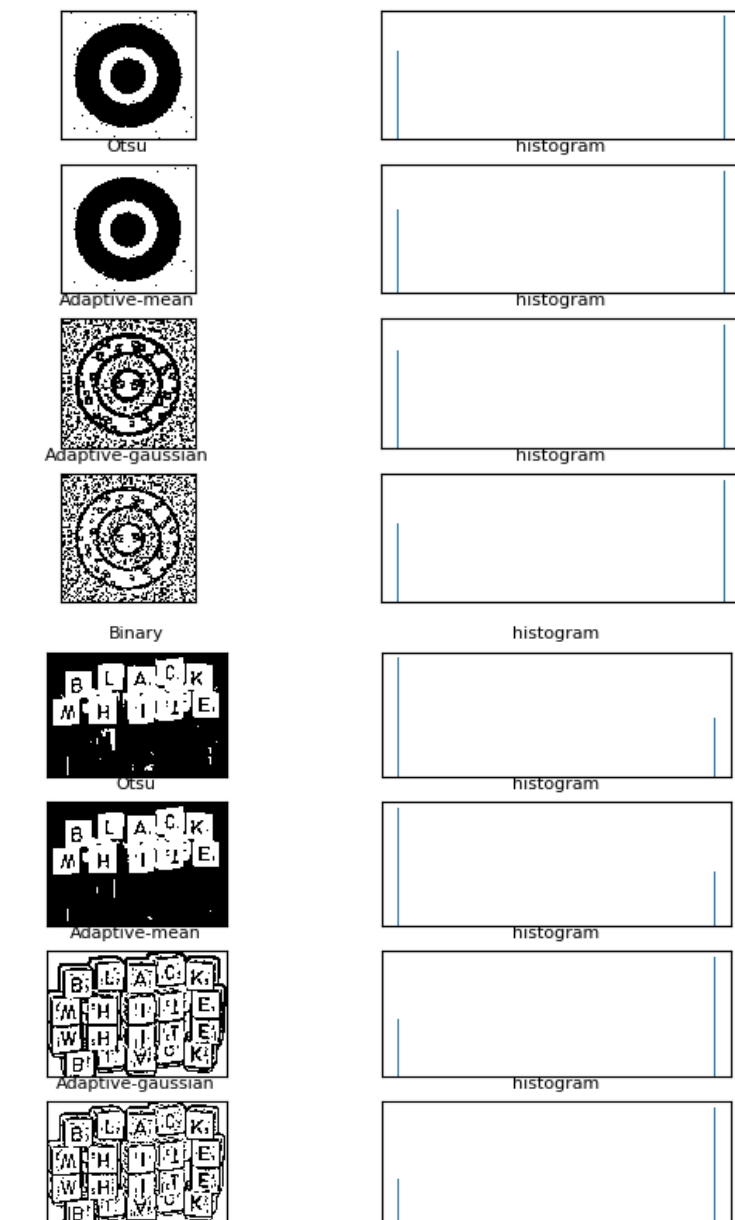
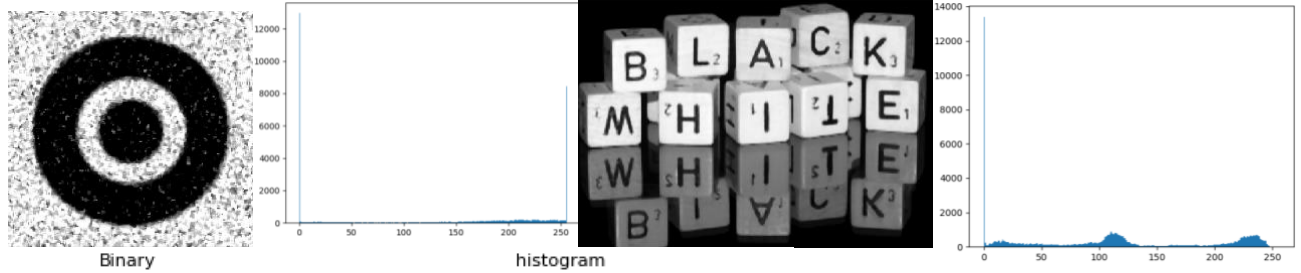




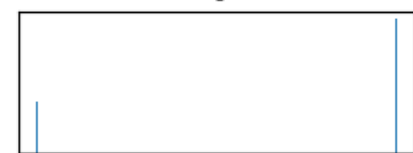
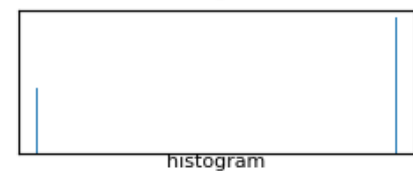
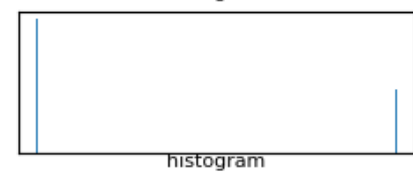
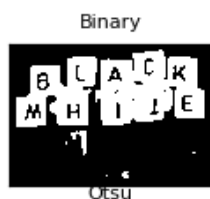
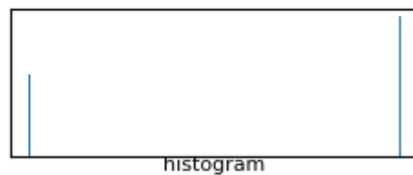
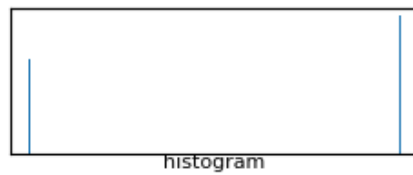
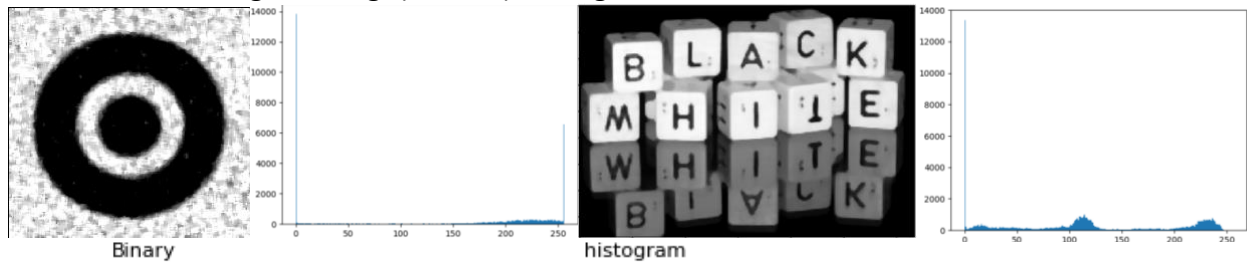
- **[img1]** Firstly, from the histogram results, it can be seen that all four methods have successfully separate the pixel values to just 0(black) and 255(white), thus from this perspective, they all work equally well. Secondly, from the result images' perspective, global thresholding algorithms and Otsu thresholding performs generally better than the Adaptive thresholding algorithm. As the results show less noise and clearer separation between foreground and background. In this case, I used threshold value equals to 127 and Otsu algorithm also produces a "most suitable" threshold value equals to 127. However, threshold value is needed to be set manually for global thresholding algorithm, but Otsu thresholding algorithm calculated the most suitable" threshold value automatically and this can be seen as a more desirable feature of Otsu thresholding algorithm over Global thresholding algorithm.
- **[img2]** Firstly, from the histogram results, it can be seen that all four methods have successfully separate the pixel values to just 0(black) and 255(white), thus from this perspective, they all work equally well. Secondly, from the result images' perspective, adaptive thresholding algorithms perform generally better than the other two. Because the results produced by adaptive thresholding algorithms show clearer separation between foreground and background for both the dice and its reflections.
- Overall, based on the shape of histograms of both images, it is reasonable to select a thresholding value in the middle of the intensity range in order to obtain a better performance. This can also explain why Otsu thresholding algorithm behaves similarly to global thresholding with threshold value equals to 127 and why does global thresholding with threshold value equals to 127 performs better than other thresholding values for both images. Moreover, **global thresholding** algorithm works better on image 1 than image2 because the pixel value distribution of image1 is more uniform than that of image2, it is simpler to distinguish which value of threshold value to choose in order for global thresholding algorithms to perform better. Furthermore, for **Otsu threshold algorithms**, it tends to find a threshold value which minimize intra-class variance and maximize inter-class variance (Otsu algorithm chooses 135 as threshold value) however, the reflection part of image2 are in grey, thus they are likely to be smaller than that threshold value, as a result the pixel intensity of these areas are likely to be set to 0 and they will be shown as black in the output result. Finally, for **adaptive threshold algorithms**, they are influenced by the selected region blocks, for the reflection part of image2, they ought to have higher pixel values than the background, thus if a selected region contains both the reflection part and background, the pixel value of the reflection part are very likely to be higher than the locally selected threshold value and they are very likely to be set to 255, this can explain why the reflection part are clearly separated from the background using this threshold algorithm.

1.2 Median filtering + thresholding

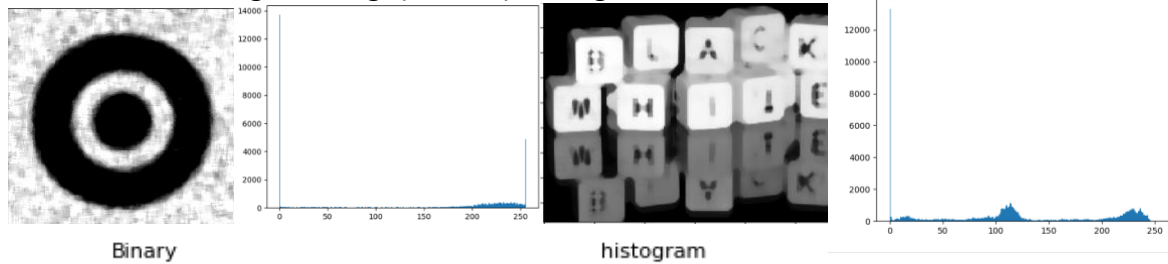
Kernel size = 3, Original image(blurred)+histogram



Kernel size = 5, Original image(blurred)+histogram



Kernel size = 7, Original image(blurred)+histogram



Binary



Otsu



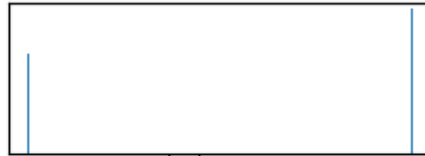
Adaptive-mean



Adaptive-gaussian



histogram



histogram



histogram



histogram



Binary



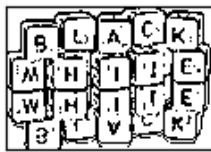
Otsu



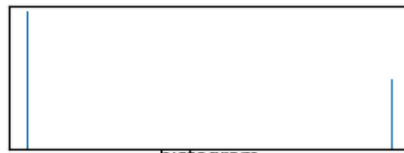
Adaptive-mean



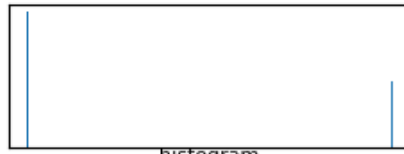
Adaptive-gaussian



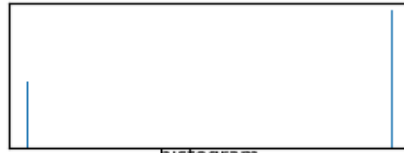
histogram



histogram



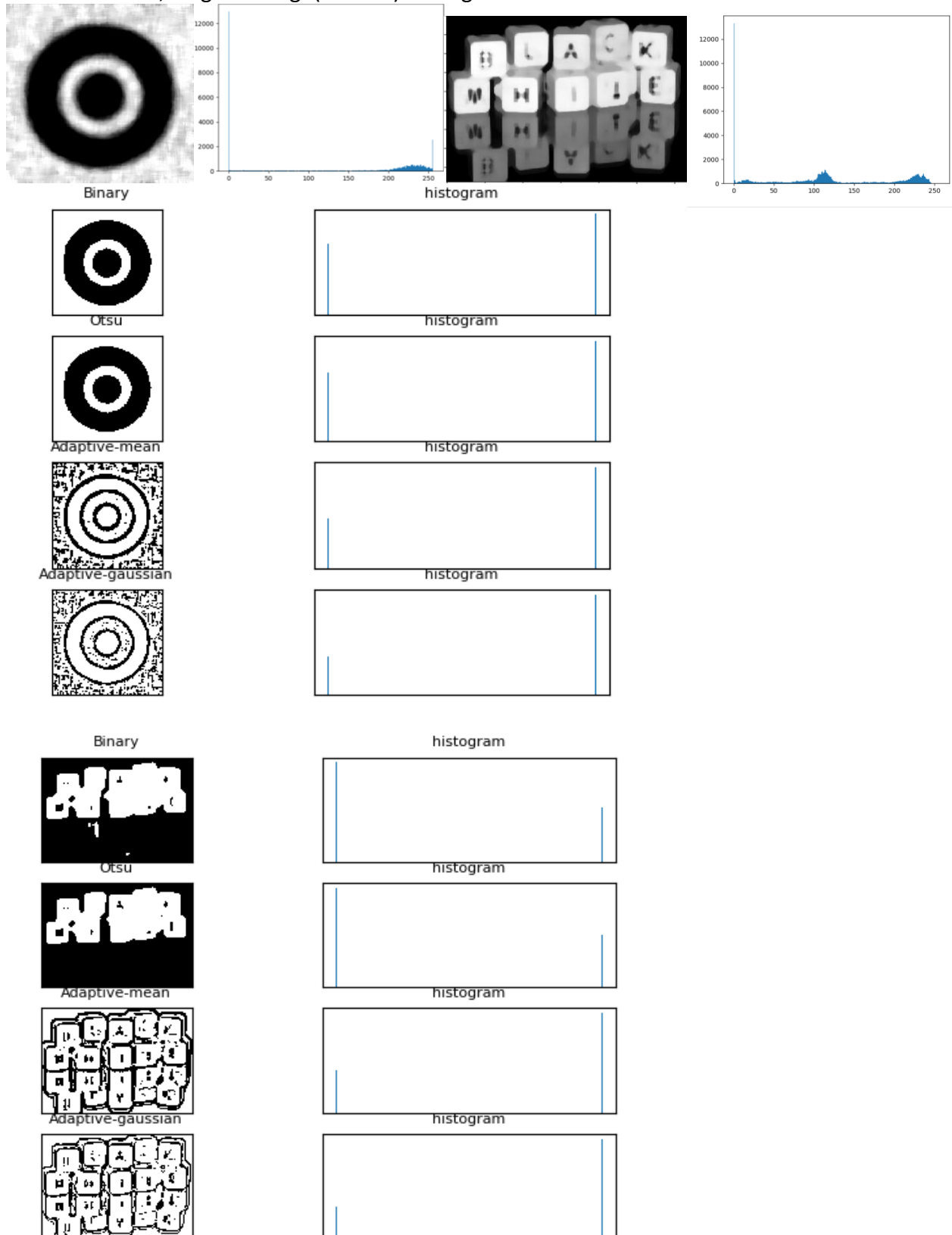
histogram



histogram

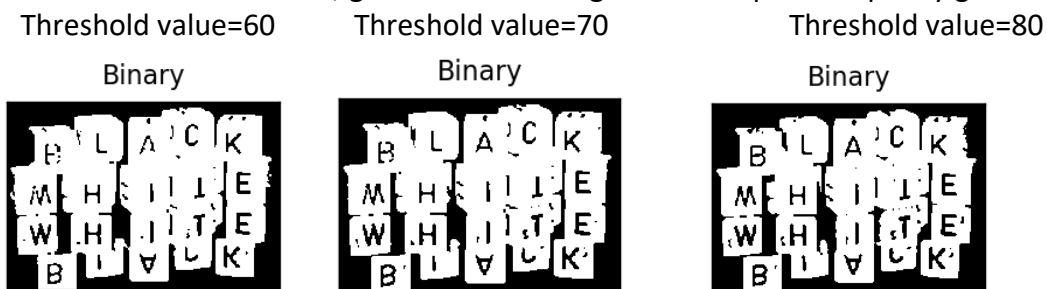


Kernel size = 11, Original image(blurred)+histogram



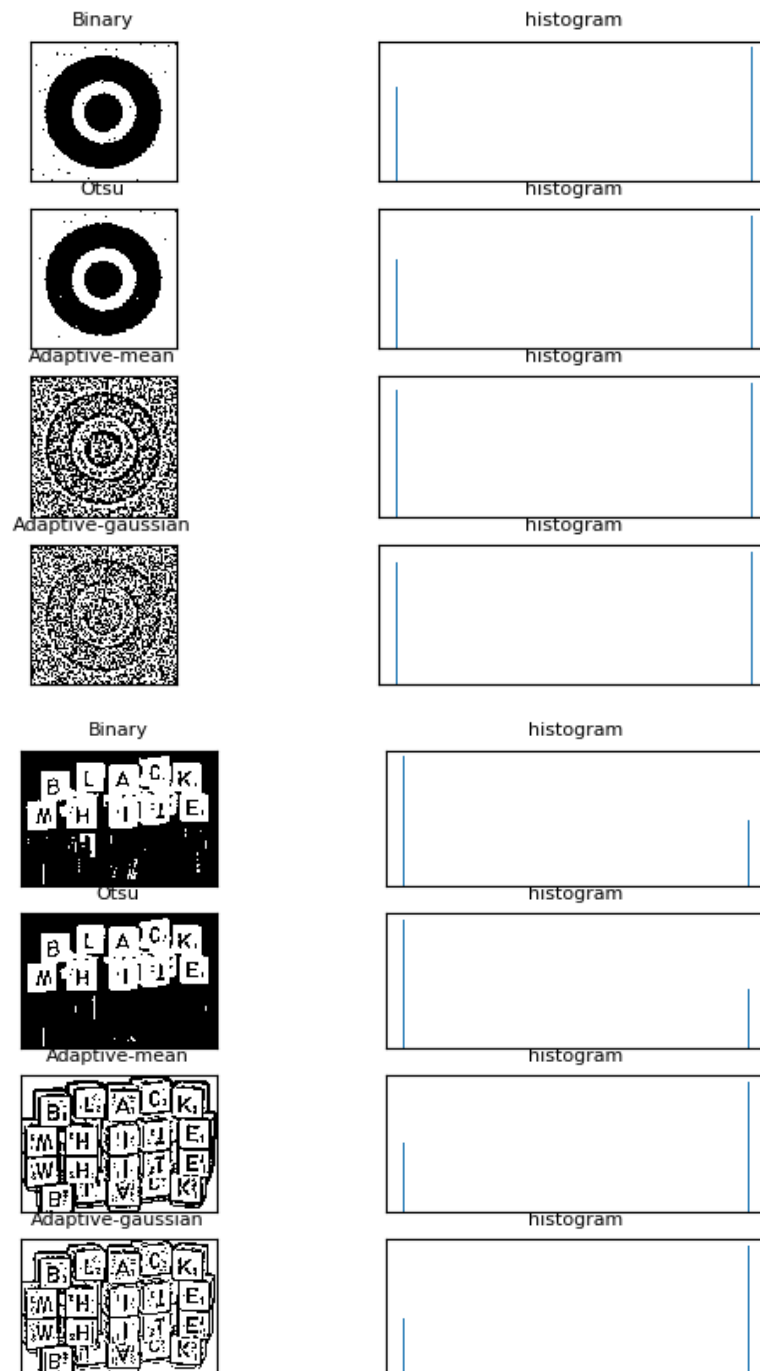
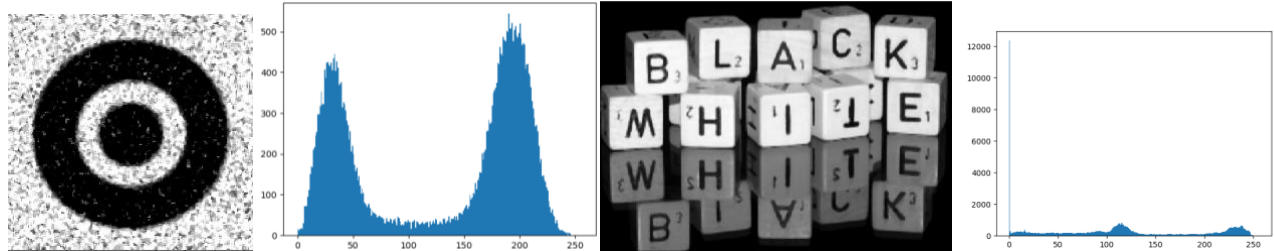
For median filtering, I tried different values of kernel size, and it can be found that:

- For image1,
 - when kernel size equals to 3, global threshold algorithm and Otsu threshold algorithm work better than others, this can be shown from the histogram of the original blurred image, it is reasonable to select a threshold value in the middle (around 127). However, the result still contains some noises.
 - As kernel size increases, it can be noticed that the amount of noise drops and Otsu thresholding performs similarly well or even better than global thresholding this can be shown from the histogram, selecting a thresholding value in the middle might not be the best choice, a value smaller than 127 might give a better result (Otsu algorithms chooses 117).
 - Overall, with kernel size in the range of [5-27], Otsu and global thresholding both work well with the blurred image. However, results are likely to have noise with smaller with kernel size and results are likely to be out of shape with larger with kernel size (lose of too much information), thus, it is reasonable to choose a kernel size around 11.
- For image2,
 - Adaptive threshold algorithm works better. Comparing with Other global thresholding algorithms, Adaptive threshold algorithm show clear segmentation between the foreground and background but contain a few noises.
 - As kernel size increases, the result becomes worse, I think the reason behind is that unlike image1, where the image is relatively simply, image2 contains more details. After blurring the images, image2 loses details.
 - Overall, it can be seen that with kernel size set to 3, Adaptive threshold algorithm produces clearer separation and images are not very blurry comparing with others.
- Overall, images show clearer separation between foreground and background, less noise after applying median filtering. Global and Otsu threshold algorithms behave equally well on image1, but Otsu threshold algorithm is more advantageous, as threshold value does not need to be set manually. However, if I manually set a smaller threshold value, global threshold algorithm can produce pretty good result:

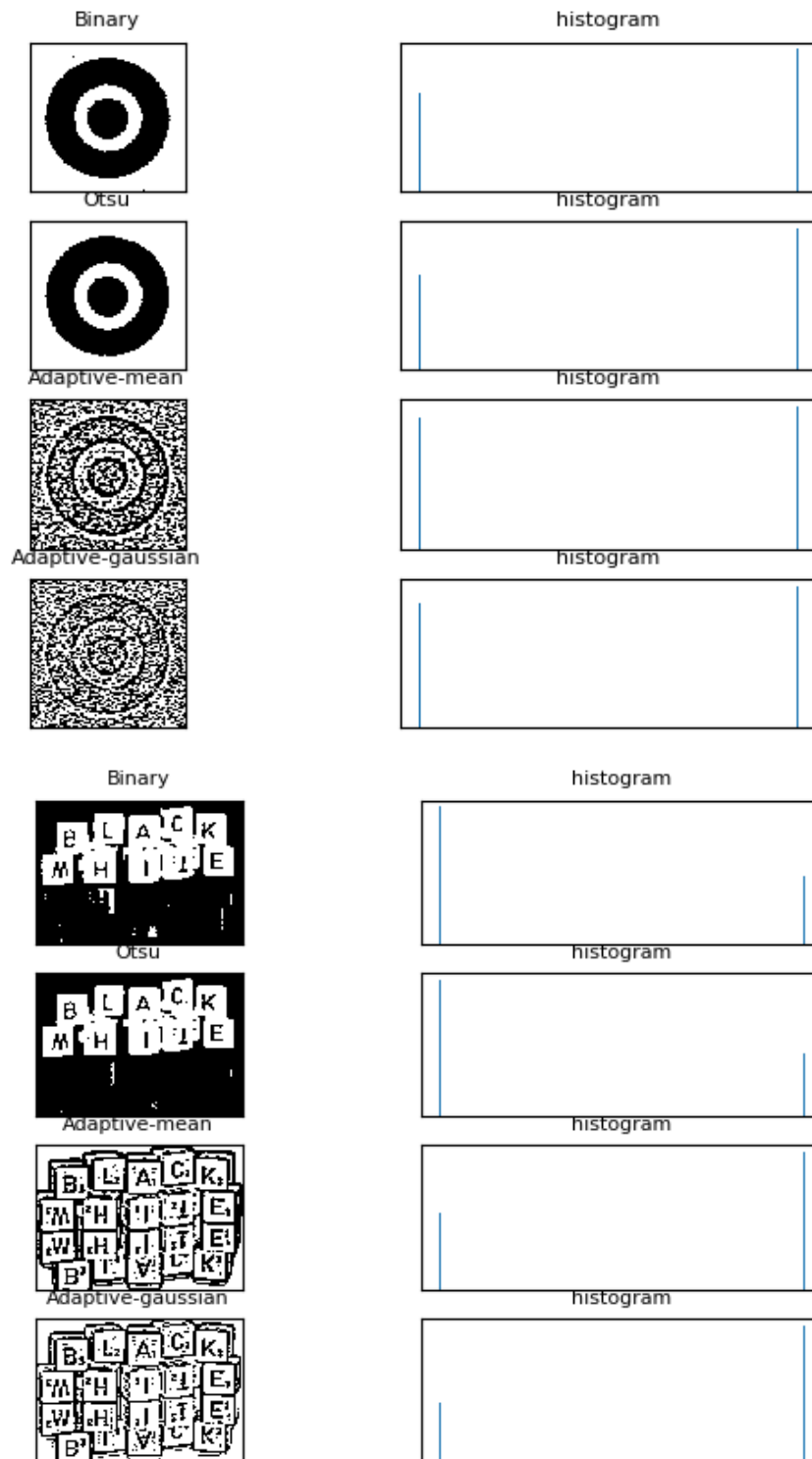
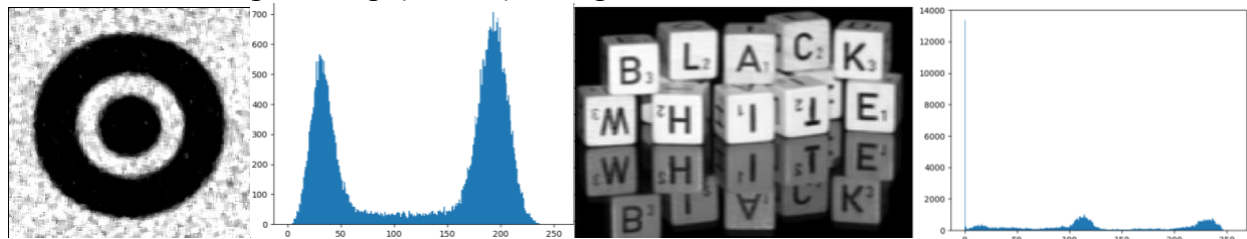


1.3 Gaussian filtering + thresholding

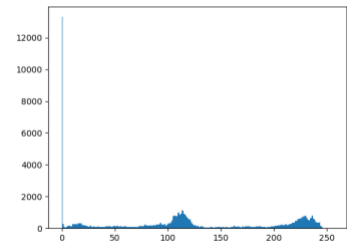
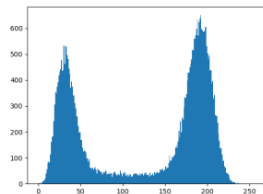
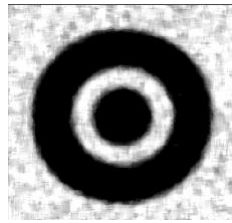
Kernel size = 3, Original image(blurred)+histogram



Kernel size = 5, Original image(blurred)+histogram



Kernel size = 7, Original image(blurred)+histogram



Binary



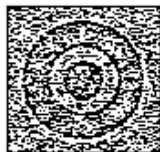
Otsu



Adaptive-mean



Adaptive-gaussian



histogram



histogram



histogram



histogram



Binary



Otsu



Adaptive-mean



Adaptive-gaussian



histogram



histogram



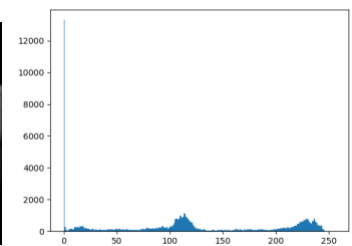
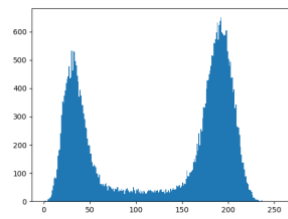
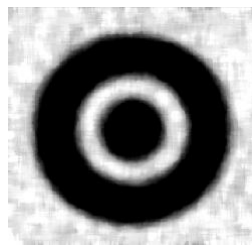
histogram



histogram



Kernel size = 11, Original image(blurred)+histogram



Binary



Otsu



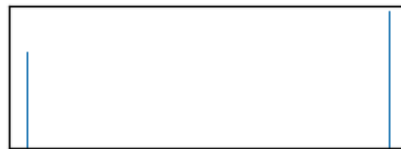
Adaptive-mean



Adaptive-gaussian



histogram



histogram



histogram



histogram



Binary



Otsu



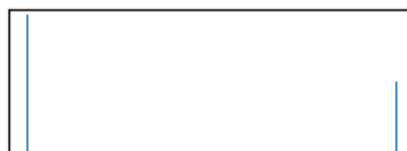
Adaptive-mean



Adaptive-gaussian



histogram



histogram

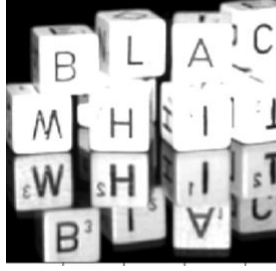





histogram



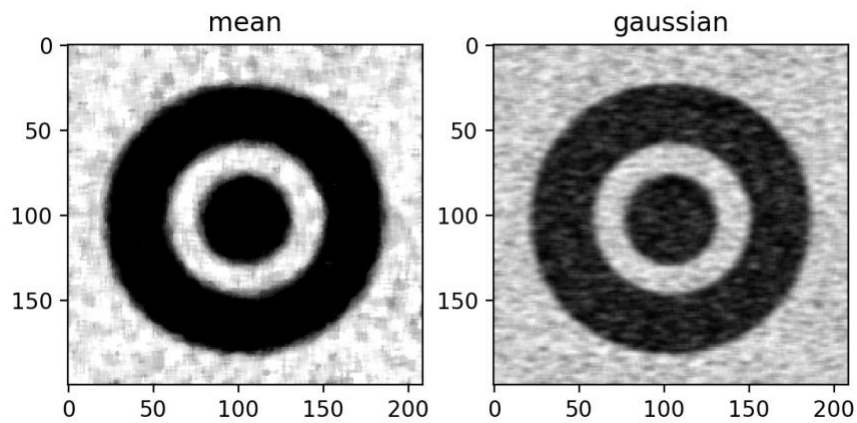
histogram



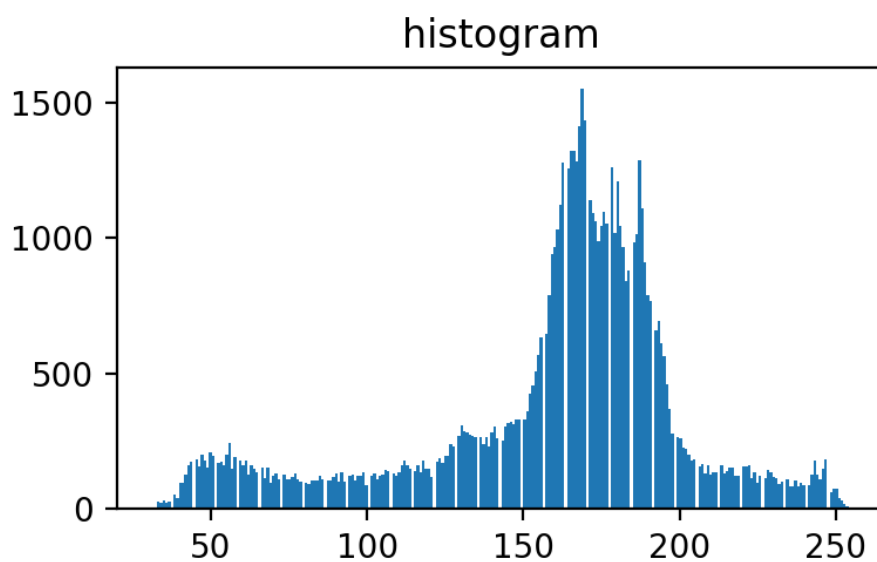
Gaussian kernel standard deviation = 3	Gaussian kernel standard deviation = 5	Gaussian kernel standard deviation = 7	Gaussian kernel standard deviation = 9
			

For gaussian filtering, I tried different values of kernel sizes and standard deviations, it can be found that:

- For image1,
 - From the shape of the histogram, it can be found that for global thresholding algorithms, a threshold value should be chosen around 127. In this case, I have chosen 127 as my threshold value, and the output of global thresholding works well. Moreover, Otsu thresholding behaves even better, with less noise. Because for a histogram with two peaks, it is suitable to use Otsu threshold algorithm to find a threshold value which gives minimal intra-class variance and maximal inter-class variance.
 - I have tried various values of kernel sizes and it can be found that the histogram shape is not influenced much with the increase of kernel size. Moreover, it can be seen that when kernel size is smaller than 7, the result images still contains some noises. Thus, an optimal performance can be obtained by setting threshold value equal to 7.
 - the result of global thresholding(binary) and Otsu thresholding performs similarly well with all different kernel sizes.
- For image2,
 - Adaptive threshold algorithm performs better than the other two and when kernel size = 3. With larger kernel sizes, the original image becomes excessively blurry and the outputs loses too much information.
- For gaussian filtering, another important factor is the standard deviation, I tried different standard deviations and found that increasing standard deviation reduces noise but also attenuates high frequency details(edges) and make image blurry. Thus, in this case, I chose standard deviation = 3.
- Overall, similarly to median filtering, after applying gaussian filtering, performance of all three thresholding algorithms show less noise and clear segmentation between foreground and background. Furthermore, it can be noticed that both filtering approach reduces noise dramatically, however, median filtering keeps edges relatively sharp. Below is a comparison between them with kernel size = 7:



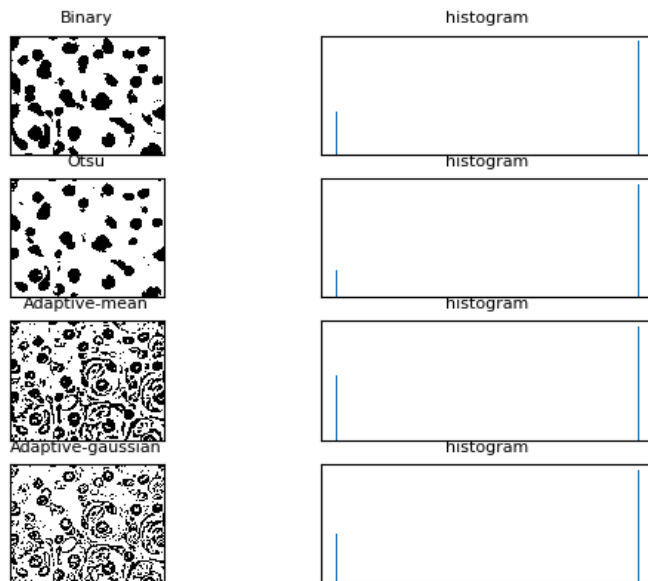
Stage 2: Use filtering and thresholding for segmentation



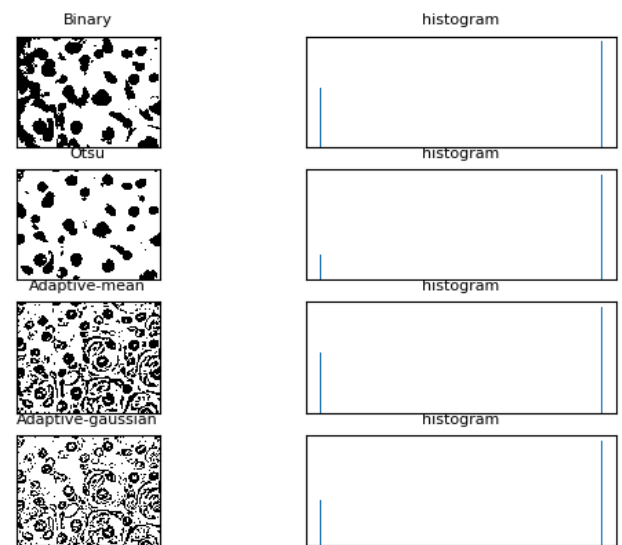
For this part, firstly, from plotted the histogram, it can be found that most of the grey and black rounded dots are **smaller than a value in between 150 and 200**. Thus, it is reasonable to set the thresholding value as some number in this range.

1. Kernel size = 3,
 - a. Median filtering

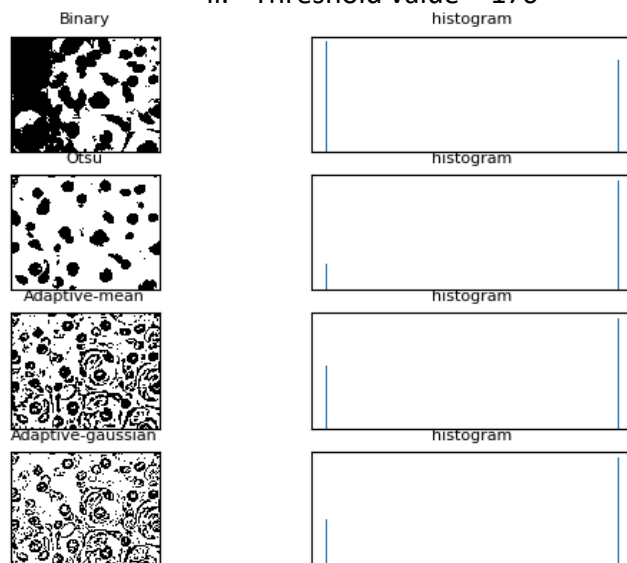
- i. Threshold value = 150



- Threshold value = 160

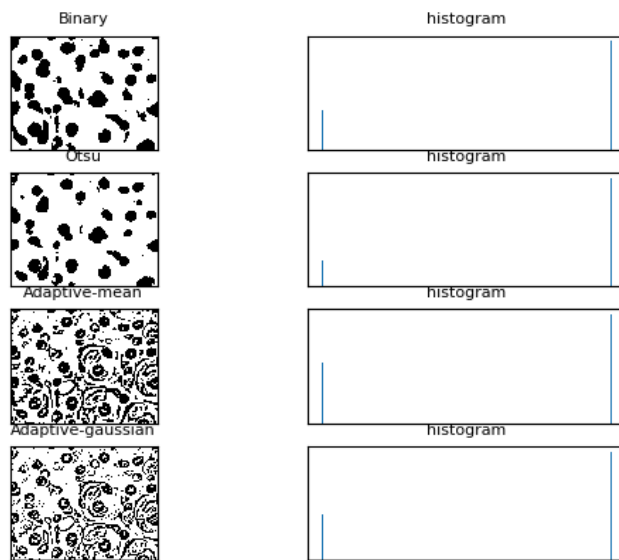


- ii. Threshold value = 170

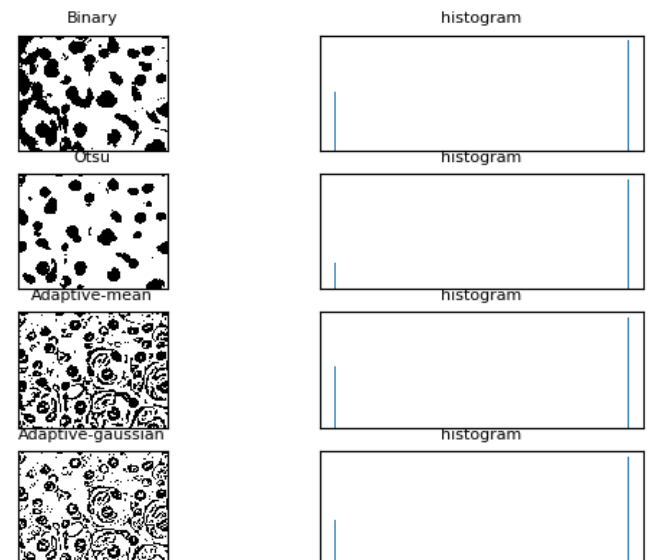


b. Gaussian filtering

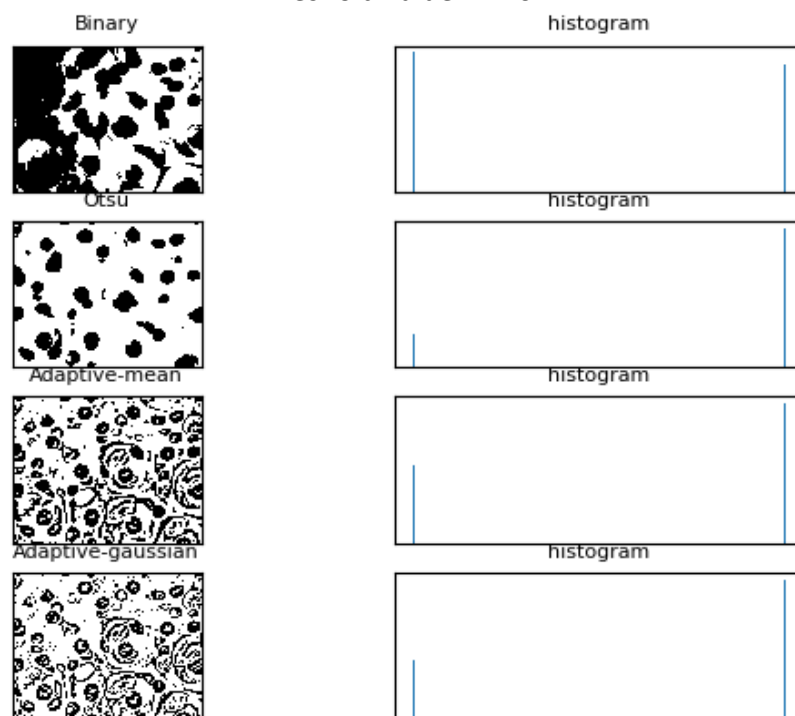
i. Threshold value = 150



Threshold value = 160



ii. Threshold value = 170

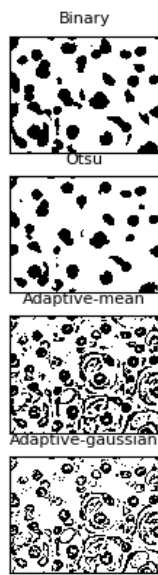


When kernel size is set to 3, it is noticed that when threshold value equals to 150, the result of Global threshold algorithm shows clearer edges around cells and it separates more cells from the background comparing with other threshold values. Moreover, the result of Global threshold algorithm after median filtering has less noise and sharp edges comparing with the same result after gaussian filtering.

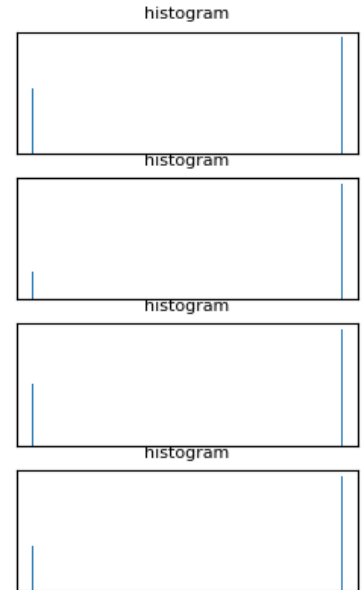
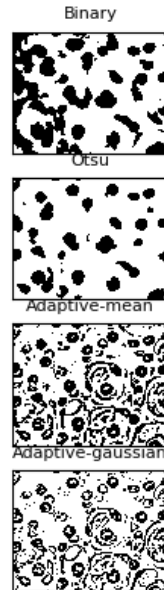
2. Kernel size = 5,

a. Median filtering

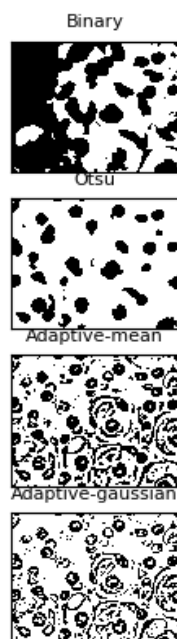
i. Threshold value = 150



Threshold value = 160

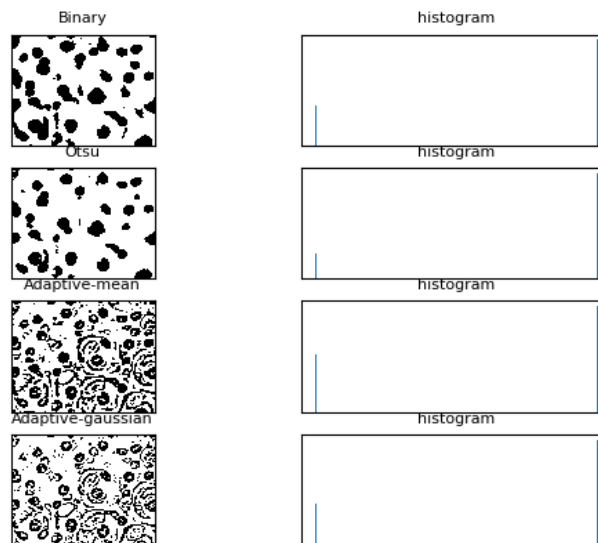


ii. Threshold value = 170

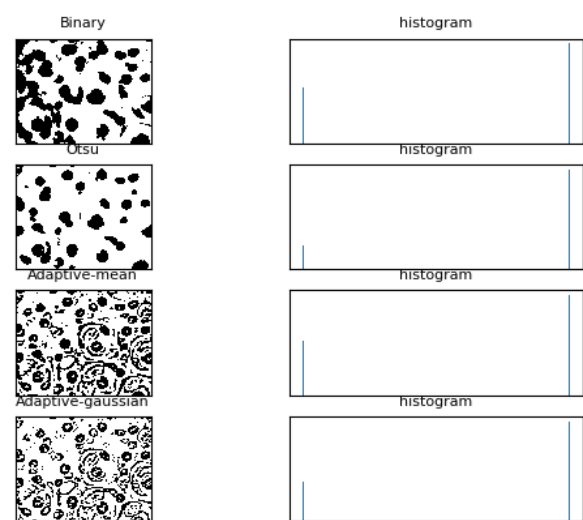


b. Gaussian filtering

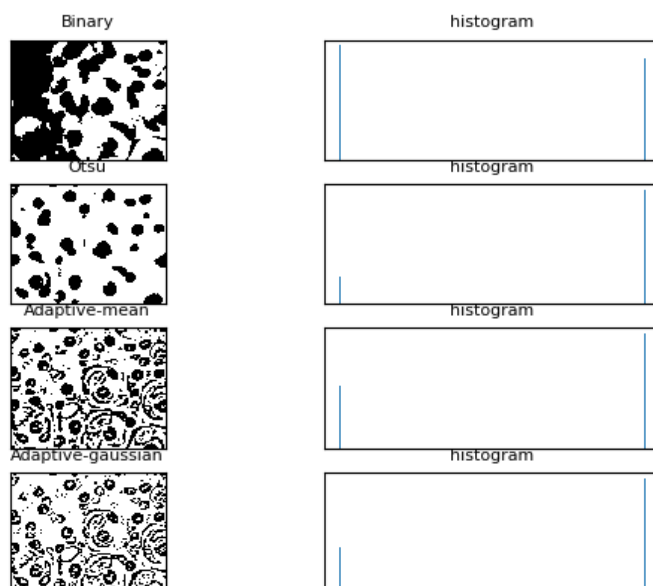
i. Threshold value = 150



Threshold value = 160



ii. Threshold value = 170

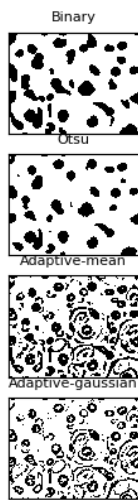


When kernel size is set to 5, the results are similar to kernel size equals to 3 but contain less noise. Global threshold algorithm with threshold value set to 150 performs better than other threshold algorithms. Similarly, the result of Global threshold algorithm after median filtering has less noise and sharp edges comparing with the same result after gaussian filtering.

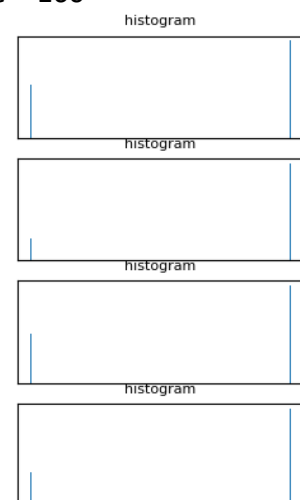
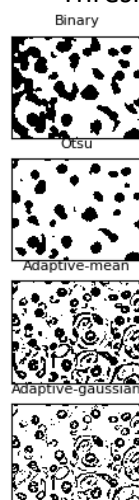
3. Kernel size = 7,

a. Median filtering

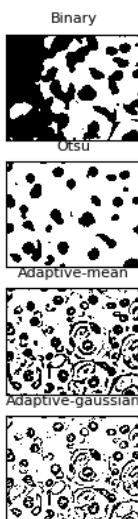
i. Threshold value = 150



Threshold value = 160

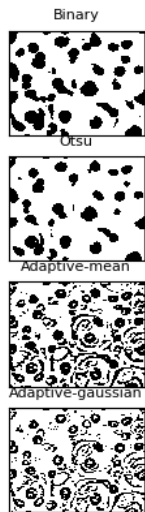


ii. Threshold value = 170

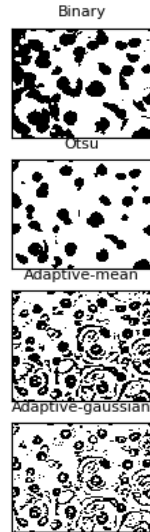


b. Gaussian filtering

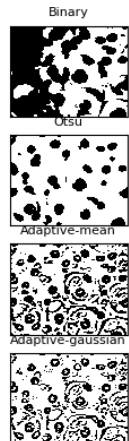
i. Threshold value = 150



Threshold value = 160



ii. Threshold value = 170



When kernel size is set to 7, it can be seen that Global threshold algorithm with threshold value set to 150 performs better in both median and gaussian filtering cases. However, the results after median filtering has less noise but edges are not very smooth.

Segmentation process:

1. Take input of kernel size.
2. Run median or gaussian filtering based on input kernel size.
3. Run different thresholding algorithms on blurred images
4. Compare results, choose the most suitable one based on the amount of noise, clearness of edges and the number of cells be segmented.

In conclusion, taking the best segmentation results from each different kernel sizes, it can be found that if the kernel size is smaller, the amount of noise in the result increases but the edges smoother and clearer. However, if the kernel sizes are larger, the amount of noise in the result decreases but the edge becomes blurry. In order to obtain relatively good segmentation result, **I have chosen kernel size equal to 5 for median filtering and threshold value of 150 for global threshold algorithm.** Median filtering works better than gaussian filtering because median filtering removes noise while keeping edges sharp, whereas gaussian filtering removes noise and reduce contrast.