COMP61342 : Computer Vision

First Assessed Practical

Bowen Zhang 10289193

1. Image1

In this section, I choose the function ‘imbinarize’ to binarize the grayscale image by thresholding.

Firstly, I use the default thresholding to binarize the image. The following images are the code and the result of this method.

```
f=imread('brains.pgm');

figure(1);
result=imbinarize(f);

imshow(result);
```

After analyzing the results, I found that the binarized image lost a lot of the original image content, so I modified the threshold. I use the function ‘graythresh’ to calculate the threshold. The following images are the code and the result of this method.

```
f=imread('brains.pgm');

figure(1);
level=graythresh(f);
result=imbinarize(f,level);

imshow(result);
```

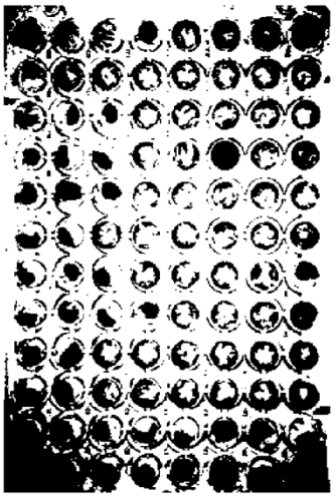
Through this attempt, I found that the results did not improve significantly, so I decided to improve the binarization results by continuously adjusting the threshold.

Finally, I found that the binarization result is best when the threshold is 0.74. On the right is the result when the threshold is modified.

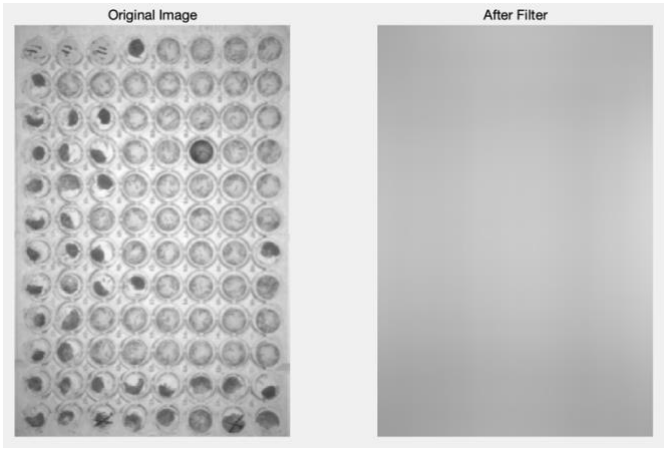
2. Image2

2.1 Method 1

After analyzing the second image, I found that the result of direct binarization was not good. There were large black areas in the result, as shown on the right.

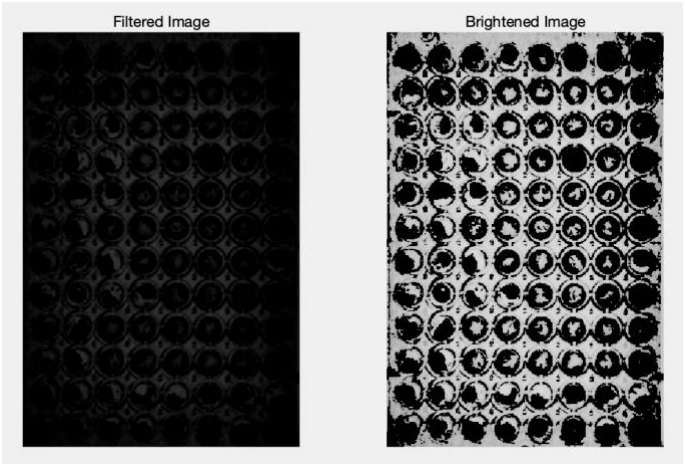


I decided to use Gaussian filtering to perform noise reduction on the picture. Gaussian filtering is a process of weighted average of the entire image. The value of each pixel is obtained by weighted average of itself and other pixels in the neighborhood. The following images are the code and comparison of the original image and the image after Gaussian filtering.



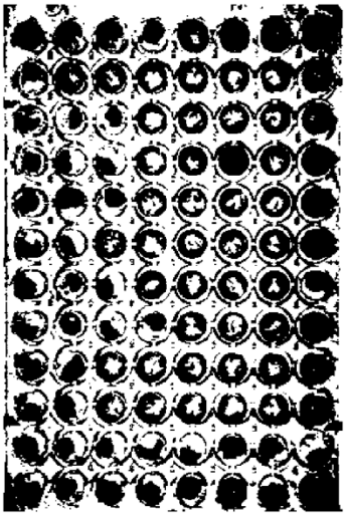

```
I = imread("tray.pgm");
W = fspecial('gaussian', [150,150], 100);
K = imfilter(I, W, 'replicate');

figure(1);
subplot(121); imshow(I); title('Original Image');
subplot(122); imshow(K); title('After Filter');
```

I used the function 'imsubtract' to process the original image to get the final image, but the final image brightness is very low, so I used the function 'brighten' to brighten the image. The following is the image obtained after processing.




```
final = imsubtract(I, K);

F=im2double(final);
N = brighten(F,0.9);

figure(2);
subplot(121); imshow(F); title('Filtered Image');
subplot(122); imshow(N); title('Brightened Image');
```

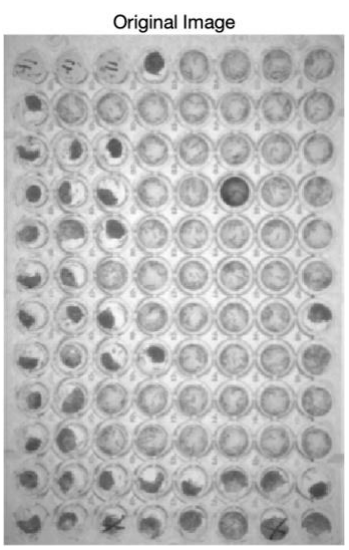
After obtaining the pre-processed image, I used the function 'imbinarize' to binarize the image. The result is shown on the right.

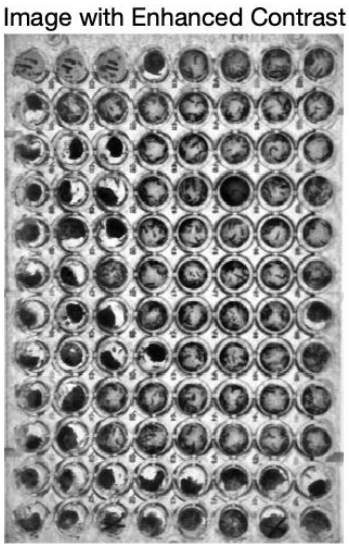


2.2 Method 2

As we can see, the result after Gaussian filtering still has some black areas. No matter how I adjust the parameters, I cannot get the image that I am satisfied with, so I decided to try it with other methods.

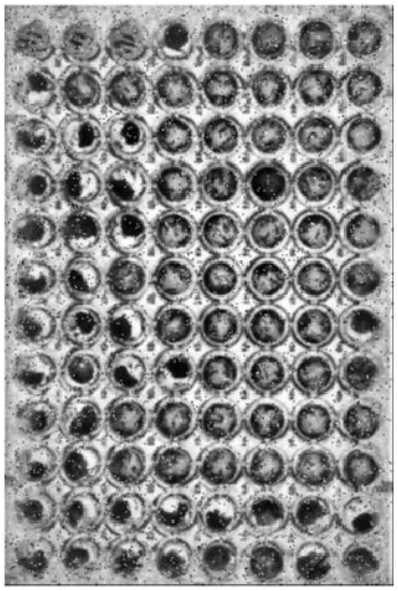
I first enhanced the contrast of the image using the function “adapthisteq”. Unlike “histeq”, it performs operations on small data areas rather than the entire image. It enhances the contrast of each tile so that the histogram of each output area approximately matches the specified histogram. I can limit the contrast enhancement to avoid noise that may be present in the enlarged image. The following image is the original image and the image with enhanced contrast.





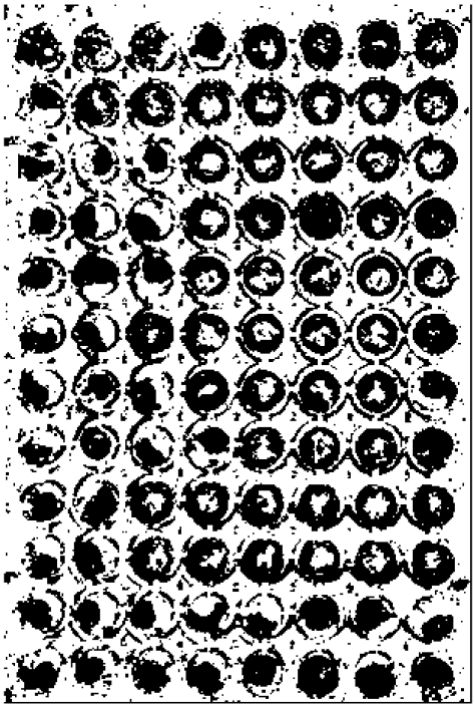

```
I=imread('tray.pgm');

Ada = adapthisteq(I);

figure(1);
subplot(121); imshow(I); title('Original Image');
subplot(122); imshow(Ada); title('After Filter');
```

Then I added salt and pepper noise to the image, and then processed the image using mean filtering to get the following image. Mean filtering is a linear average filter, which obtains the pixel value of the central pixel by averaging all the pixels in the window. This has the advantage that it can effectively smooth the image, reduce the sharpness of the image, and reduce noise, but it cannot completely eliminate noise.



```
J = imnoise(Ada,'salt & pepper',0.05);
Kaverage = filter2(fspecial('average',2),J)/255;
figure(3)
imshow(Kaverage)
```

After that, binarize the preprocessed image, and the following results are obtained by adjusting parameters.



After preprocessing using mean filtering, the result of the binarization has almost no additional black areas, and basically achieves the expected effect. By comparison, the image preprocessed using mean filtering is better than the image preprocessed using Gaussian filtering.

In the same type of grayscale image binarization, such as images with uneven illumination and blurred borders, the mean filtering should be given priority in preprocessing, and then other methods such as Gaussian filtering are considered. I think the performance of mean filtering is better than Gaussian filtering.