KENNESAW STATE
UNIVERSITY

CS 4732
MACHINE VISION

Assignment 3

Morphological Filters & Spatial Filtering

INSTRUCTOR

Dr. Sanghoon Lee

Your Name: Max Haviv

KSU ID: 001029496

1. ABSTRACT

In this assignment I learn morphological and spatial filtering. For example Erosion, Dilation, Closing, and Opening. These filtering methods are used to

2. TEST RESULTS

2.1 Test Results for number two image

Erosion Filter on Image

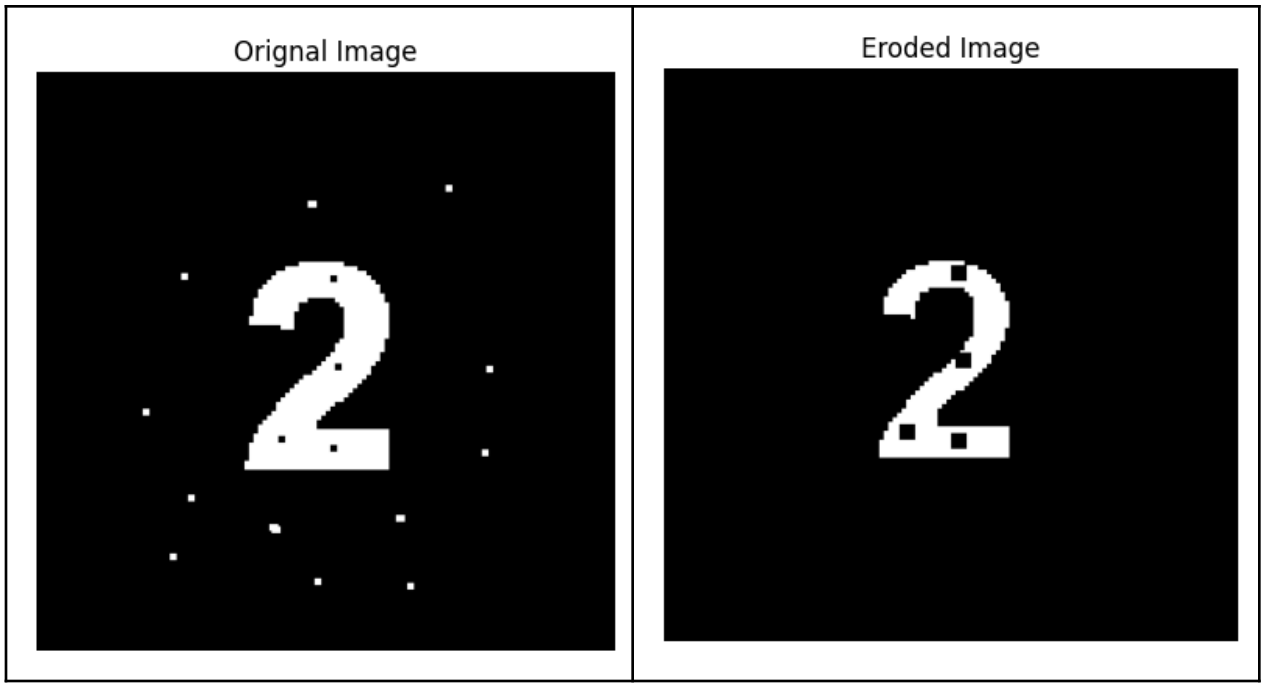


Fig (1) Original Image and image after erosion filter is applied

Opening Filter on Image

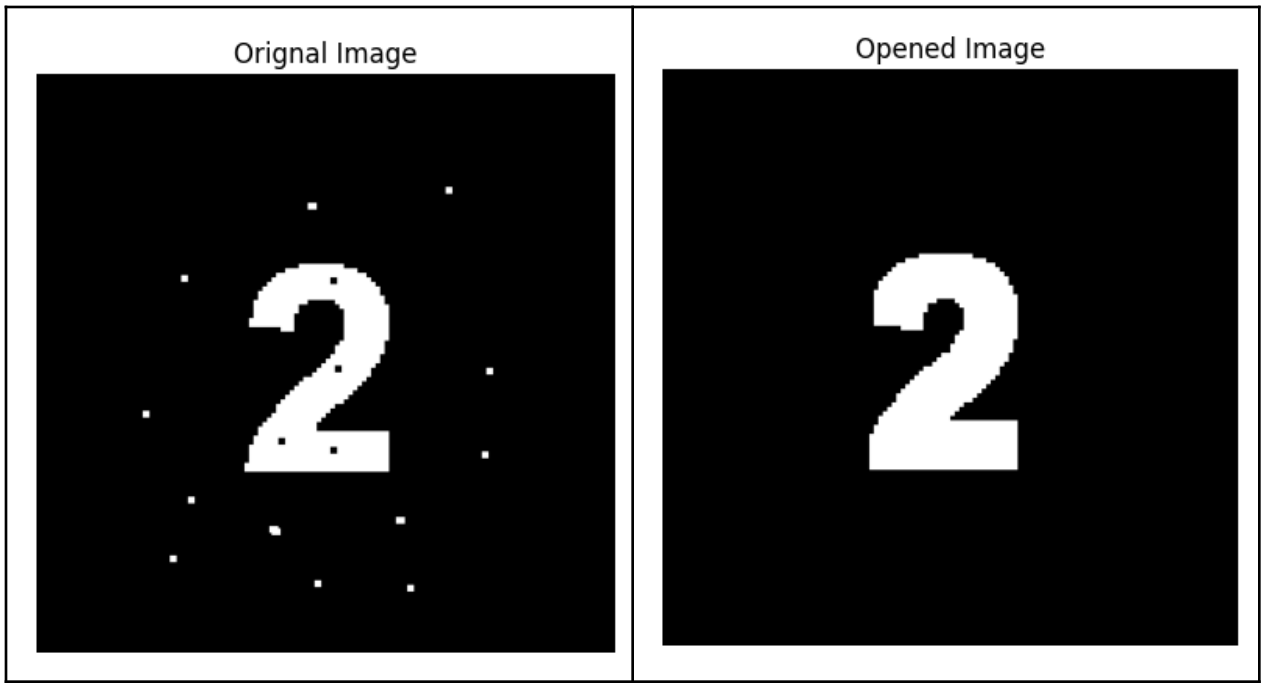


Fig (2) Original Image after opening filter is applied

Two filtering methods were used on the original image. In figure 1 the original image to remove the white dots that surround the number two. This works perfectly but also ends up making the black dots inside of the two larger and the actual number becomes thinner. In figure two the Opening filter is applied to the image to remove both the white and black dots. This works perfectly as well as the black and white dots have been removed. But the number becomes a bit thicker as a result.

2.2 Test Results for Fingerprint Image

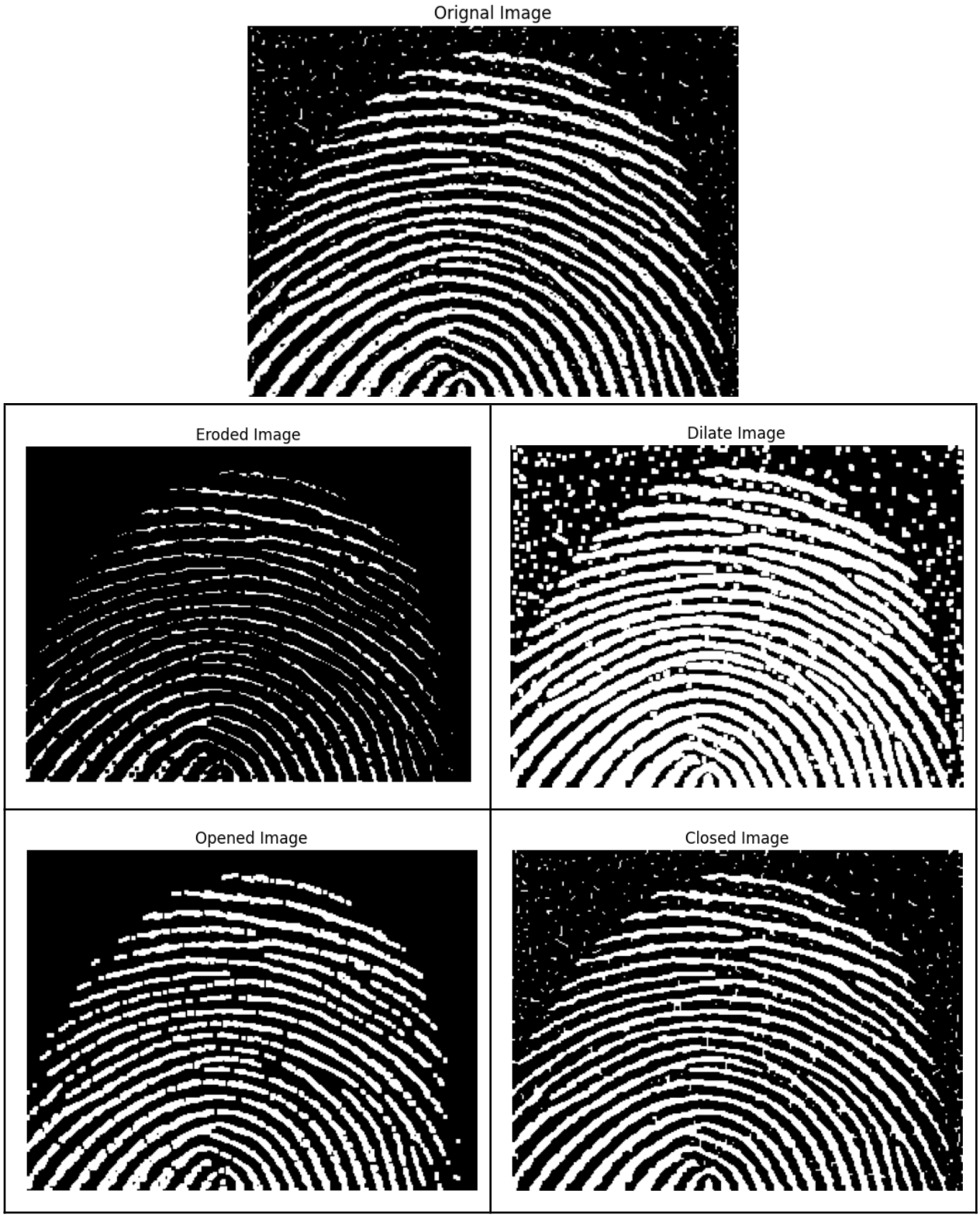
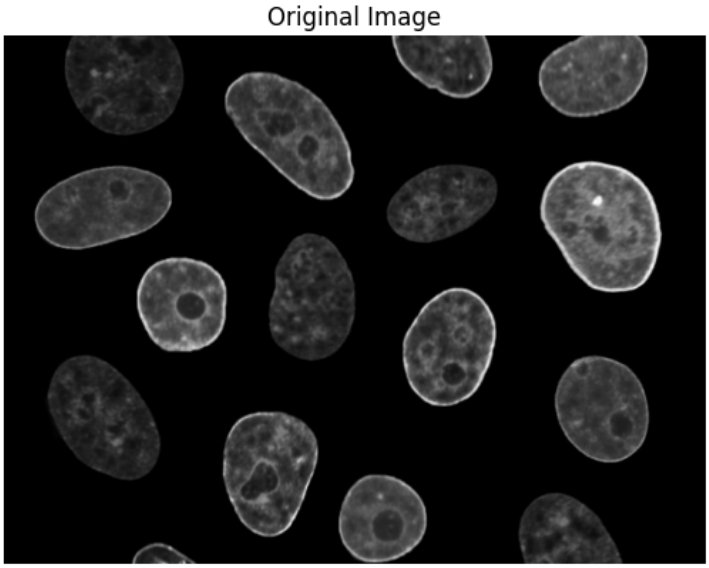
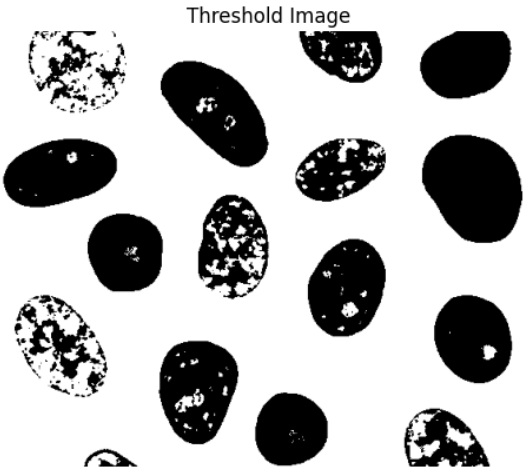


Fig (3) Original Image followed by multiple morphological filters applied to it

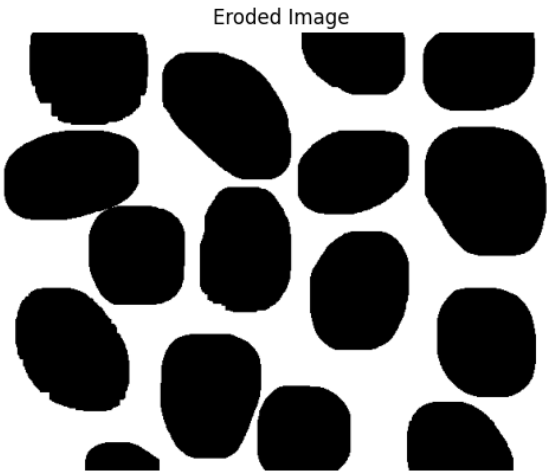
As shown in figure 3 a variety of filters are applied to the original fingerprint image. The best results in my opinion come from eroding the original image and opening the original image. Both bring clarity to the image without destroying the original detail of the image. Dilating and Closing the image on the other hand magnifies the white disturbance pixels which I do not believe is beneficial.

2.3 Test Results for counting cells





(1)



(2)

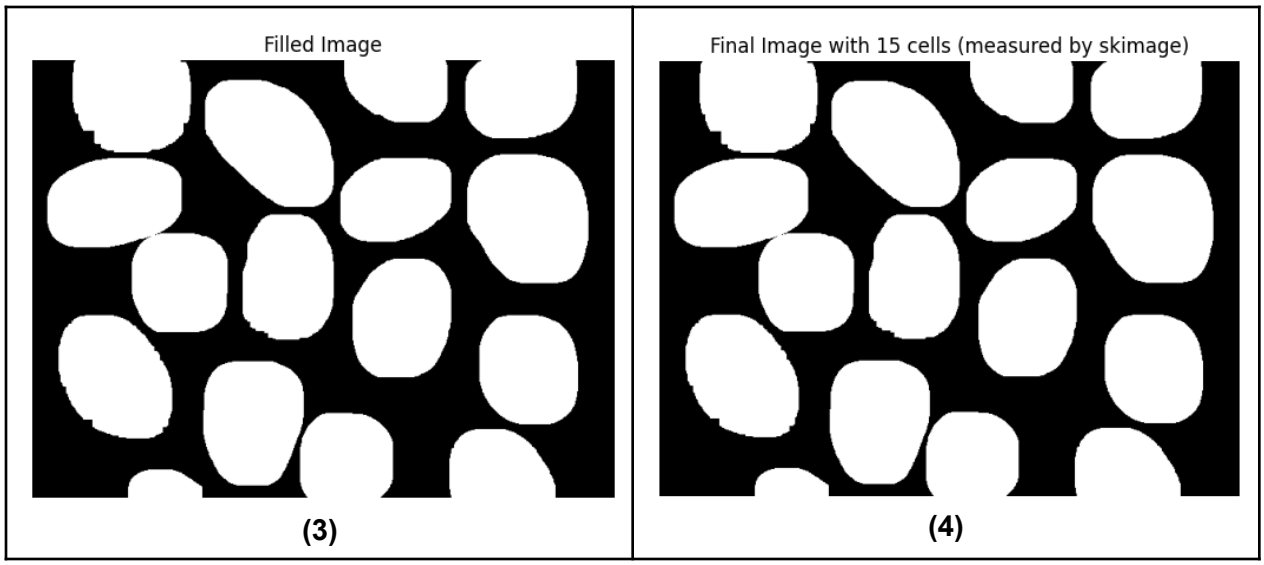


Fig (4) Original Image followed by the transformations applied to count number of cells (1-4)

For counting the number of cells in the image cell.jpg I decided to apply multiple filtering techniques to the image to get a clean image to apply a skimage measure to count the number of distinct regions in the image. As seen in figure 4 following from 1 to 4 you can see the steps I took to get to the final count. The first step I took was thresholding the image so I would only have two values black and white. Then I eroded the image until the cells all made their own distinct filled circles. I then filled the cells to make sure that there were no empty holes and inverted the colors to make it easier for skimage to measure the image. Finally the image was measured and the final cell count was 15. This happens to be off by one which I believe is acceptable for this homeworks assignment given the tools that we are meant to use.

3. Discussion

I learned a lot from this assignment when it comes to applying different filters to images for better object detection. Seeing different methods in action and the intended or unintended side effects has given me a new grasp on what it means to enhance an image for image detection. Especially in the case of the fingerprint. I see how trying to remove the noise can have side effects of making the actual fingerprint too hard to read by making it very bloated. While other methods that remove the noise make the fingerprint thinner and maybe even too thin to read. So you need to find a middle ground where the noise is removed but the fingerprint is still readable. Lasly when it comes to counting the cells, I feel as if I learned the most here because I needed to come up with my own way to adjust and filter the image so that I could count the number of cells using only code. This is something before taking this class I would have zero clue how to do. While the final count was wrong I am still very happy with the results since it's only off by one and I have just learned how I would even attempt this kind of problem. I believe with better filtering methods and maybe a better measuring method I could get 100% accuracy with this cell count.

4. CODE

Question 1

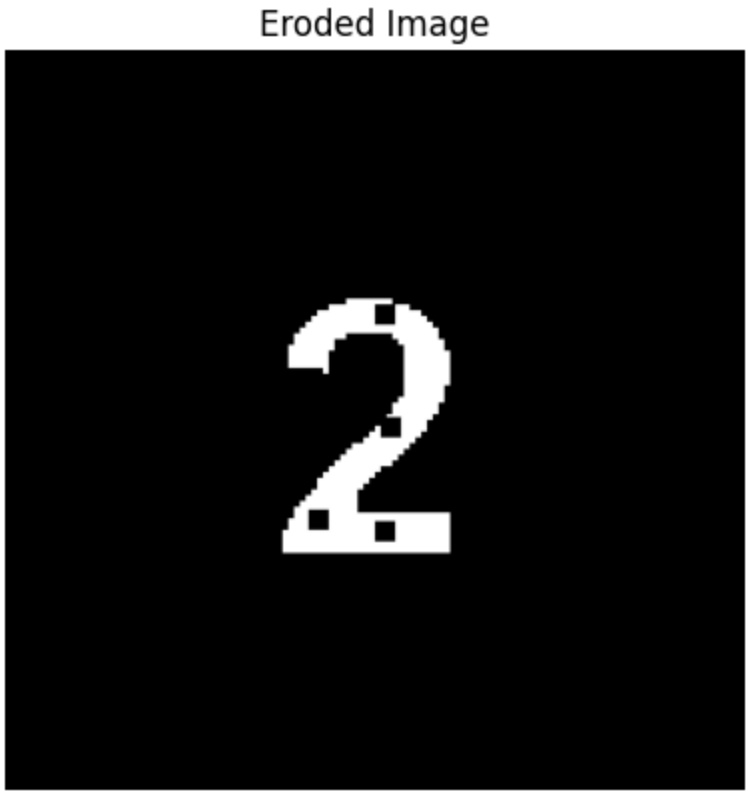
Identify the morphological operators/filters used for the given input and output images and implement the effects using morphological operators/filters on the morphology.png image.

In [31]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2 as cv

img = mpimg.imread('morphology.png')
plt.imshow(img)
plt.title("Orignal Image")
plt.axis('off')
plt.show()
```

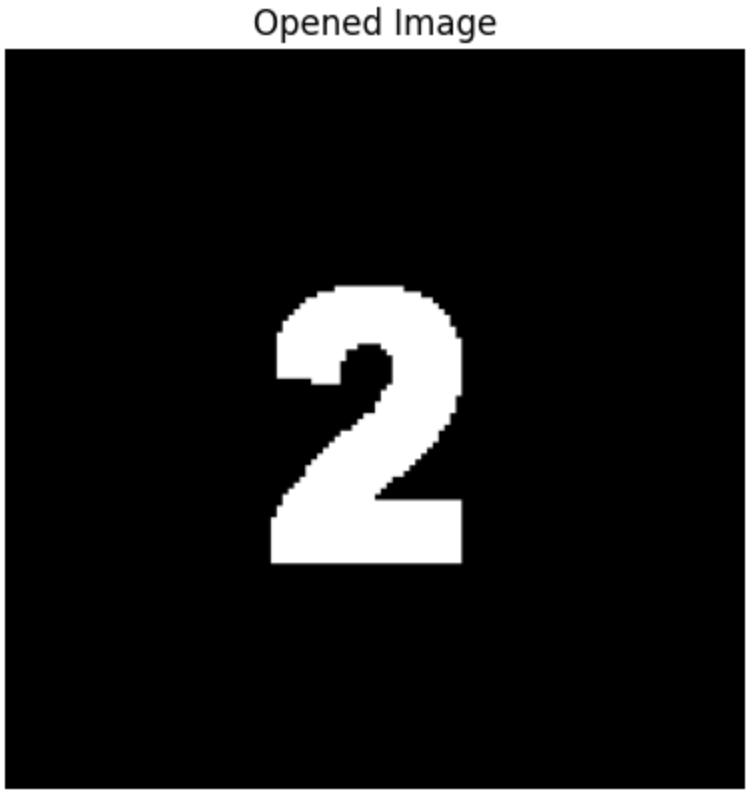
Two iterations of erosion

In [32]:
```
kernel = np.ones((3,3), np.uint8)

# two iterations of open cv erosion
erosion = cv.morphologyEx(img, cv.MORPH_ERODE, kernel, iterations=2)

plt.imshow(erosion, cmap='gray')
plt.title("Eroded Image")
```

```
plt.axis('off')
plt.show()
```



In [33]:
```
kernel = np.ones((3,3), np.uint8)

erosion = cv.morphologyEx(img, cv.MORPH_ERODE, kernel, iterations=2)
dialate = cv.morphologyEx(erosion, cv.MORPH_DILATE, kernel, iterations=4)

plt.imshow(dialate, cmap='gray')
plt.title("Opened Image")
plt.axis('off')
plt.show()
```



Question 2

Apply both morphological and median filters on the fingerprint image (fingerprint_BW.png). Compare the result and comment under what condition, one filter might perform better than the othe

In [34]:
```
fingerprints = mpimg.imread('fingerprint_BW.png')

plt.imshow(fingerprints)
plt.title("Orignal Image")
plt.axis('off')
plt.show()
```

In [35]:
```
erosion = cv.morphologyEx(fingerprints, cv.MORPH_ERODE, kernel, iterations=1

plt.imshow(erosion, cmap='gray')
plt.title('Eroded Image')
plt.axis('off')
plt.show()
```

In [36]:
```
dialate = cv.morphologyEx(fingerprints, cv.MORPH_DILATE, kernel, iterations=

plt.imshow(dialate, cmap='gray')
plt.title('Dilate Image')
plt.axis('off')
plt.show()
```

In [37]:
```
opening = cv.morphologyEx(fingerprints, cv.MORPH_OPEN, kernel, iterations=1)

plt.imshow(opening, cmap='gray')
plt.title('Opened Image')
plt.axis('off')
plt.show()
```

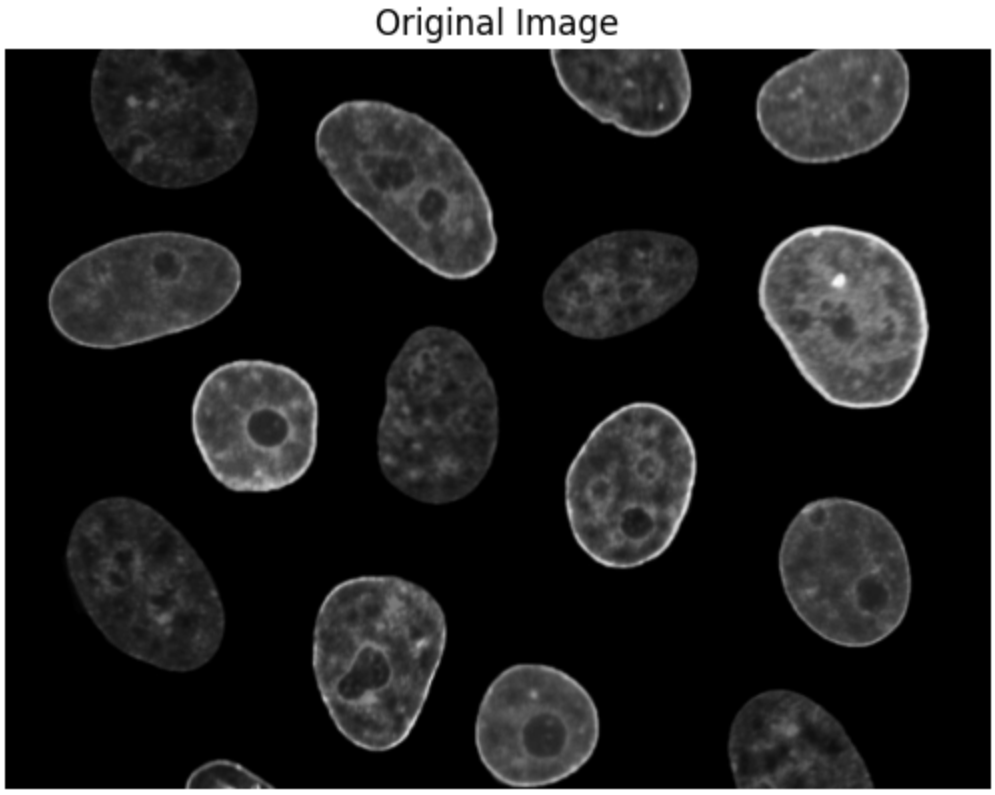
Opened Image

In [38]:
```
closing = cv.morphologyEx(fingerprints, cv.MORPH_CLOSE, kernel, iterations=1

plt.imshow(closing, cmap='gray')
plt.title('Closed Image')
plt.axis('off')
plt.show()
```

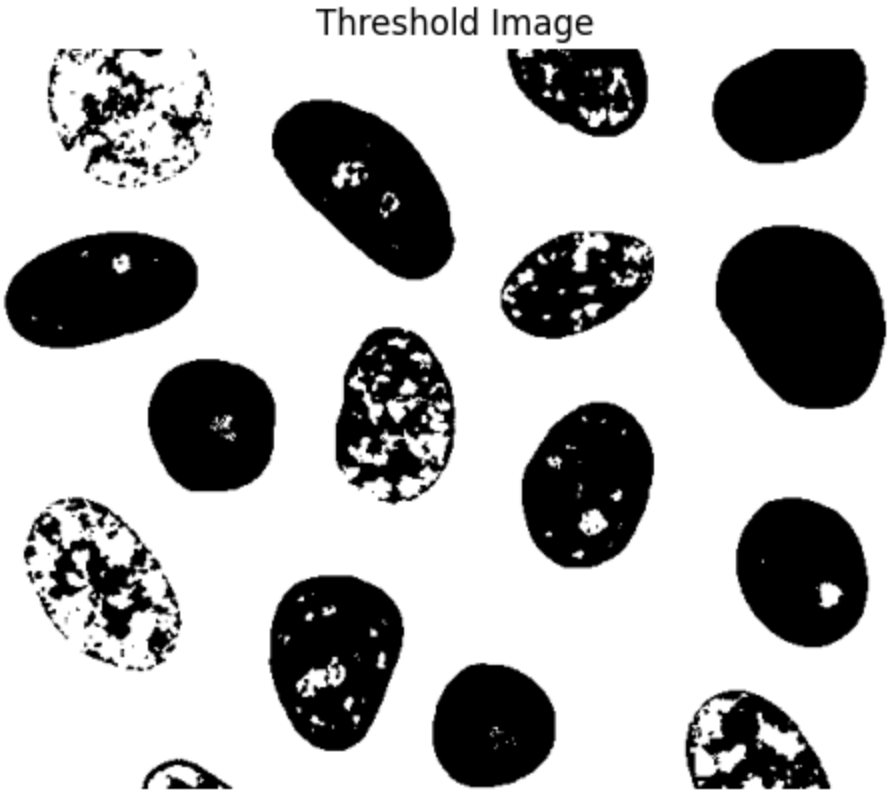
Question 3

For the cell.jpg image, write a code to count the total number of cells, calculate the size of each cell in pixels, and show the boundary of the biggest cell in an output image. In your code, you might use any techniques covered in this class. Hint: Thresholding, morphological filters, connected components, etc.

In [39]:
```
cellImg = cv.imread('cell.jpg', cv.IMREAD_GRAYSCALE)

plt.imshow(cellImg, cmap='gray')
plt.axis('off')
plt.title('Original Image')
plt.show()
```



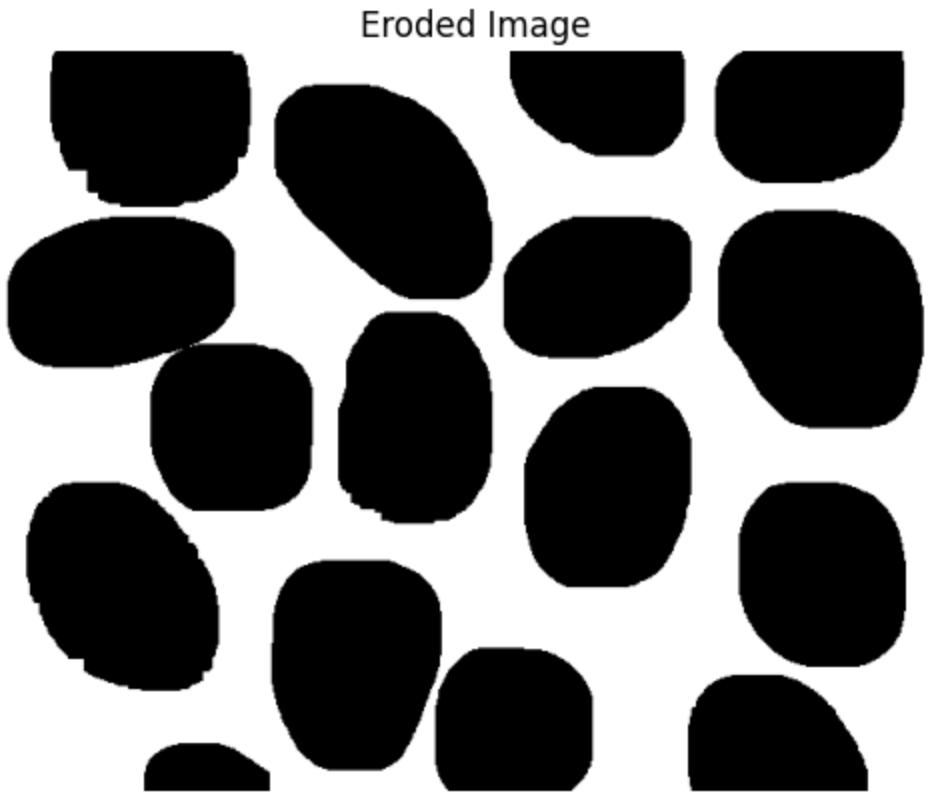
In [40]:
```
_, threshCell = cv.threshold(cellImg, 0, 255, cv.THRESH_BINARY_INV + cv.THRE

plt.imshow(threshCell, cmap='gray')
plt.axis('off')
plt.title('Threshold Image')
plt.show()
```



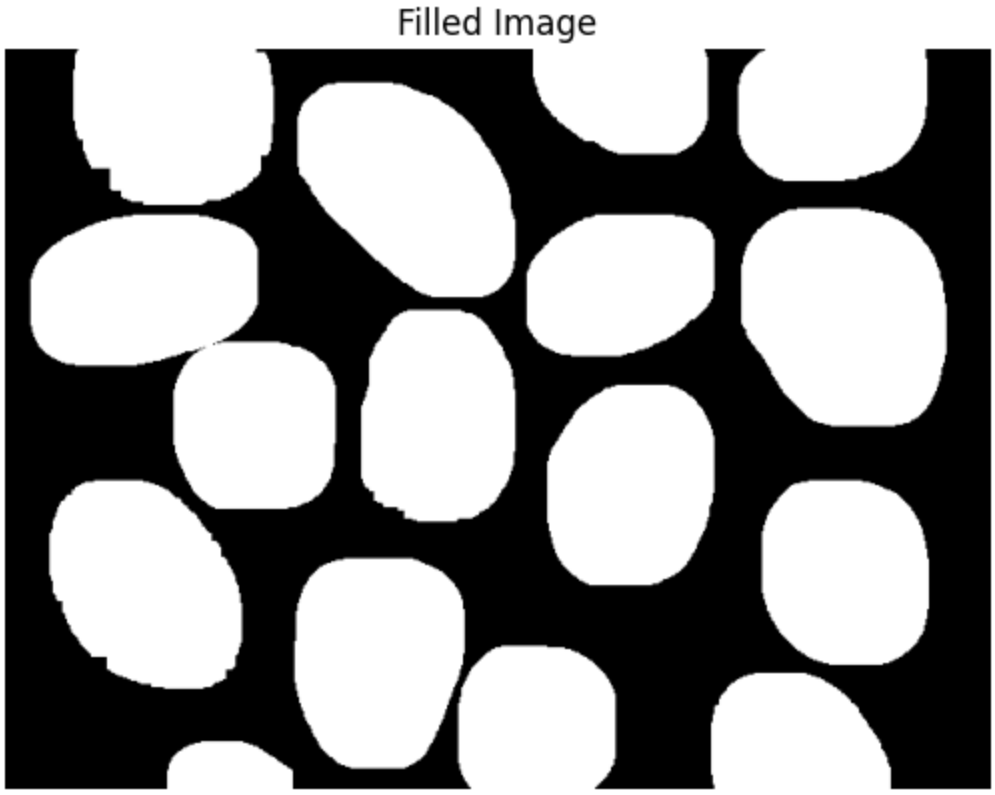
In [41]:
```
# cellEdges = cv.Canny(threshCell, 100, 200)

# plt.imshow(cellEdges, cmap='gray')
# plt.axis('off')
# plt.title('Threshold Image')
# plt.show()
```

In [42]:
```
# _, threshCell = cv.threshold(cellEdges, 0, 255, cv.THRESH_BINARY_INV + cv.

# plt.imshow(threshCell, cmap='gray')
# plt.axis('off')
# plt.title('Threshold Image')
# plt.show()
```

In [43]:
```
morphCell = cv.morphologyEx(threshCell, cv.MORPH_ERODE, kernel, iterations=1
# morphCell = cv.morphologyEx(morphCell, cv.MORPH_DILATE, kernel, iterations



plt.imshow(morphCell, cmap='gray')
plt.axis('off')
plt.title('Eroded Image')
plt.show()
```



In [44]:
```
height, width = threshCell.shape
mask = np.zeros((height + 2, width + 2), np.uint8)

# Flood fill from point (0, 0)
cv.floodFill(morphCell, mask, (0, 0), 255)

# Invert the image to get filled areas
filledCell = cv.bitwise_not(morphCell)

plt.imshow(filledCell, cmap='gray')
plt.axis('off')
plt.title('Filled Image')
plt.show()
```



In [45]:
```
from skimage import measure

# Count total cells (connected components excluding the background)
labels = measure.label(filledCell , connectivity=2, background=0)

# Count total cells (connected components excluding the background)
cell_count = labels.max()

plt.imshow(filledCell, cmap='gray')
plt.axis('off')
plt.title(f'Final Image with {cell_count} cells (measured by skimage)')
plt.show()
```