Gall Bladder Shape Extraction

Rajat Jaiswal

2017CS50415

1 Introduction

In medical practices, gall bladder shape is quite often analysed to diagnose disorder, for e.g. turns or folds in the gall bladder. Therefore, it becomes quite important to simplify this process and automate it by building supporting libraries to extract this shape from ultrasound images. In this assignment, I have applied various image processing techniques to achieve this. Finally, contour based model for detecting the edges was successful. But in the process, I have tried various other image processing methods. Image filtering such as `Average blurring`, `Median filtering`, `Bilateral filtering`, `Gaussian blurring`. Edge and contour detection methods such as `Canny edge detection`, `Laplacian of gaussian`, `Difference of gaussian`. Thresholding techniques such as `Adaptive mean thresholding`, `Adaptive gaussian thresholding`, `Otsu thresholding`. To tune the hyperparameters, I have done `Grid Search` over the set of parameters.

The final pipeline has been described in section 2. The results obtained corresponding to this method is detailed in section 4. And all the methods tried are explained in section 3.

2 Method

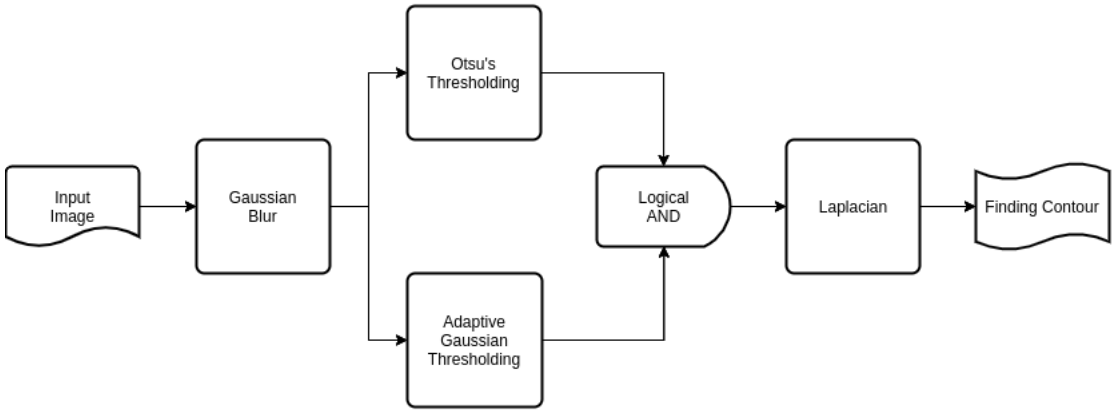


Figure 1: Pipeline of the method adopted.

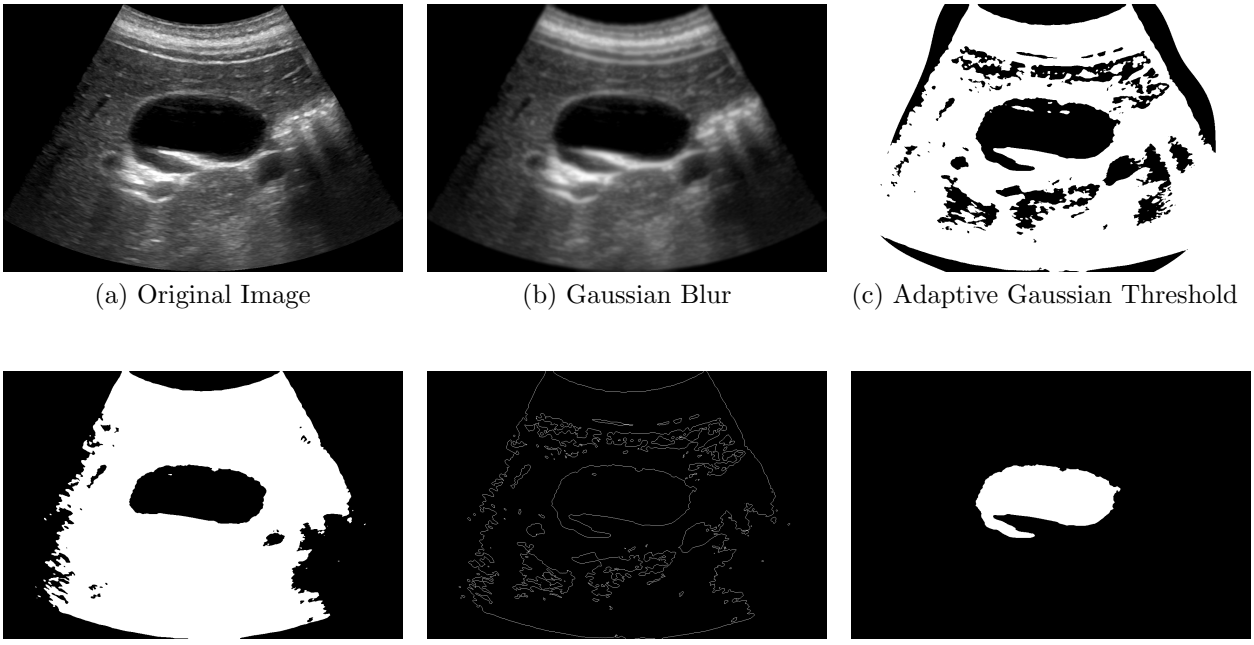
The workflow is shown in Figure 1. The input image is convoled with a gaussian filter of **sigma**($\sigma = 10$) to blur the image. It helps in removing the noise. Two different thresholding techniques are then applied on the output to convert the image into binary(black and white), which are as follows:

- **Adaptive Gaussian Threshold:** The threshold value is calculated for smaller regions leading to different threshold values for different regions adaptive to changes in local regions. It calculates threshold based on gaussian-weighted sum of neighborhood values. It is done using OpenCV's `cv2.ADAPTIVE_THRESH_GAUSSIAN_C`. The method has two hyperparameters, `block_size` and `constant`.

- **Otsu Threshold:** In Otsu Thresholding, a value of the threshold isn't chosen but is determined automatically for the entire image. In OpenCV's `cv2.threshold`, we pass `cv2.THRESH_OTSU` as an extra flag. This returns a threshold value. We multiply this threshold value with a constant between 0 to 1 and then use that as a threshold. We call this constant as `otsu_ratio`. This is another hyperparameter in the model.

The output of these two thresholding technique is combined by taking element-wise `Logical-AND`, and then it is convolved with `Laplacian` kernel to find the closed contours in the image. Note that, Gall bladder is also a closed contour. The contour corresponding to the gall bladder is extracted by finding the contour with the maximum area in a range of areas with respect to the size of the original image.

An example of this workflow is shown in the Figure 2 below.



(a) Original Image (b) Gaussian Blur (c) Adaptive Gaussian Threshold

(d) Otsu Threshold (e) Contours after Laplacian (f) Mask of the gall bladder.

Figure 2: Example of the workflow

Hyperparameter Optimization

As mentioned above, the model has 3 hyperparameters namely, `block_size`, `constant`, and `otsu_ratio`. To fine tune these hyperparameters, **Grid Search** was performed over the set of parameters. To do this efficiently, python's `multiprocessing` library was used to perform parallel computing. The hyperparameters obtained were: `block_size - 427`, `constant - 13`, and `otsu_ratio - 0.6`. The average `IoU` score obtained over the validation set for these set of parameters was **0.8396**.

3 Results

The metric used for evaluating the quality of these generated masks is `Intersection over Union(IoU)` score. `IoU` is the area of overlap between the generated segmentation mask and the ground truth divided by the area of union between the generated segmentation mask and the ground truth. The average `IoU` score obtained over the validation set is **0.8396**. Table 1 lists the `IoU` score for each image in the validation set.

Image Name	IoU Score
0000.jpg	0.7733
0001.jpg	0.8882
0002.jpg	0.8404
0003.jpg	0.7944
0004.jpg	0.8759
0005.jpg	0.8247
0006.jpg	0.8432
0007.jpg	0.8902
0008.jpg	0.8452
0009.jpg	0.8210
Average Score	**0.8396**

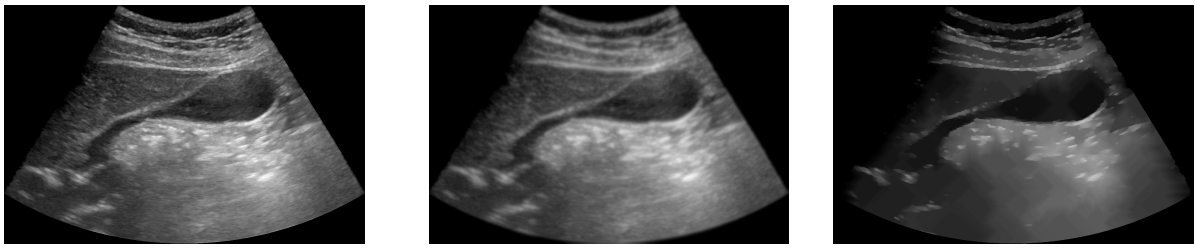
Table 1: IoU score for images in validation set.

4 Experiments

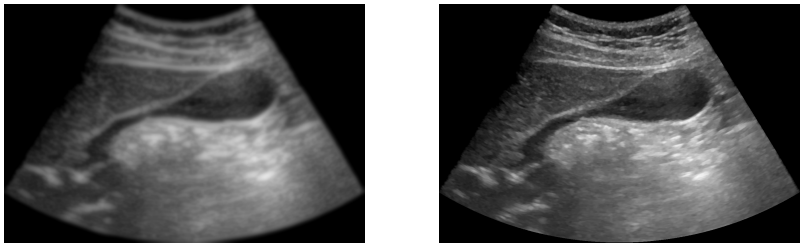
I tried various image processing techniques in filtering, thresholding, and edge & corner detection to construct the pipeline. Some of the are mentioned in the following subsections:

Image Filtering

The different techniques used were `Average blurring`, `Median filtering`, `Bilateral filtering`, and `Gaussian blurring`. `Gaussian filter` works the best in removing the noises and preserving the edges. `Bilateral filtering` removes almost all the noise but over-preserves the edges or can say introduce new ones, which create havoc down the pipeline. `Average blurring` over blurs the image and it blurs the entire image the same, hence there is no specific scale at which the gall bladder gets the focus. `Median filtering` does not seem to remove the noises, maybe it is not good for these kind of noises. Figure 3 below summarises the findings.



(a) Original Image (b) Gaussian Blur (c) Bilateral Filtering



(d) Average Filtering (e) Median Filtering

Figure 3: Example for various kind of image filtering.

Image Thresholding

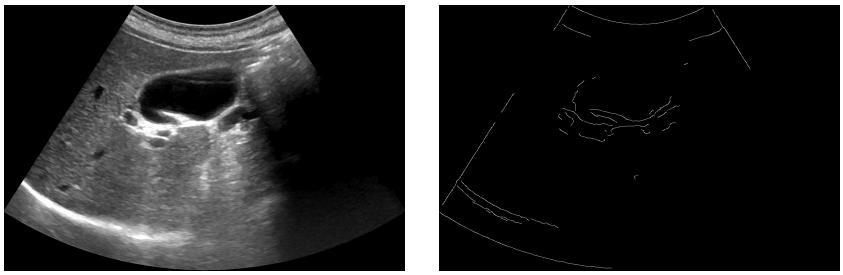
The different techniques used were `Adaptive mean thresholding`, `Adaptive gaussian thresholding`, and `Otsu thresholding`. The combination of `Adaptive gaussian thresholding` and `Otsu thresholding` works the best. `Otsu thresholding` clears the image of all noises, but over clears in a few cases. `Adaptive mean thresholding` mixes the boundaries and hence when distance between boundary is very less, it fails. `Adaptive gaussian thresholding` maintains the balance and create small blobs for the noises. Figure 4 below depicts thresholding after the gaussian blur.

(a) Original Image (b) Adaptive Mean (c) Adaptive Gaussian (d) Otsu threshold

Figure 4: Example for various kind of image thresholding after Gaussian Blur.

Edge & Corner Detection

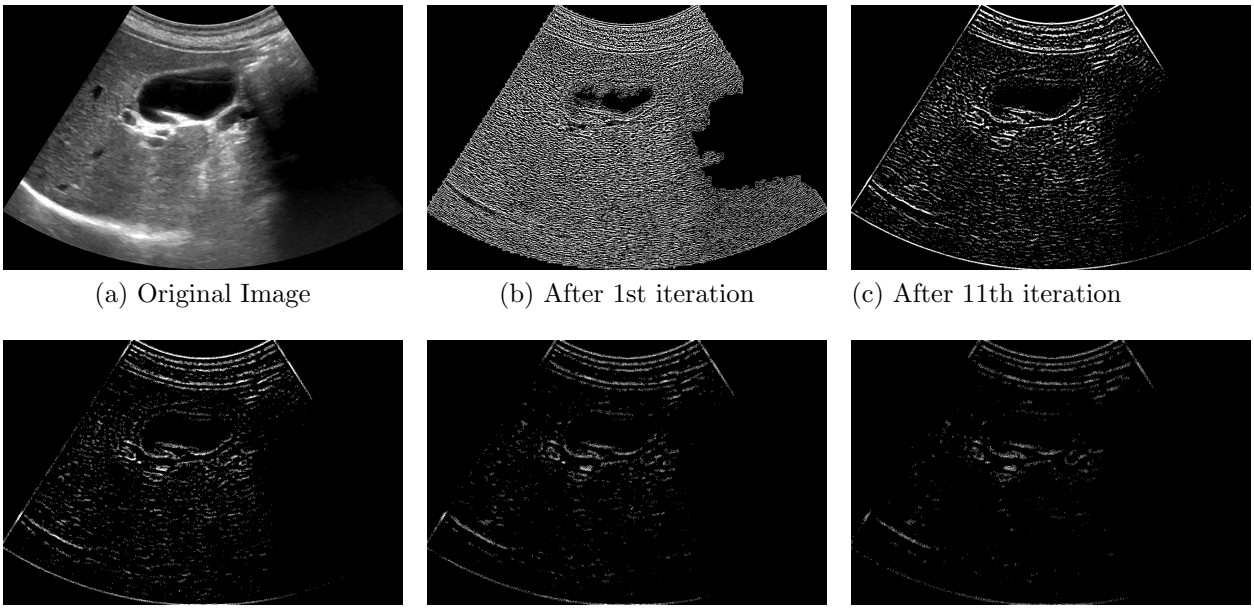
- **Canny Edge Detection:** After applying the Gaussian blur, the image is passed through canny edge detection module of `OpenCV`. The lower and upper threshold for the hysteresis came from the `Otsu threshold` method. The threshold were taken in a 1:2 ratio. The method failed miserably in detecting the closed boundary of the gall bladder. An example is shown in the Figure 5 below.



(a) Original Image (b) Canny Edge method.

Figure 5: Example of Canny Edge method after Gaussian blur.

- **Laplacian of Gaussian:** This is the method that is adopted and works very well. It forms closed contours which is property of the Laplacian. The kernel size is taken to be `3`. The results are the same as shown in Figure 2 above.
- **Difference of Gaussian:** One of the biggest issue with this method is to decide upon the number of iteration at which we should stop. It varies significantly with the image. Also at each iteration some edge of the same blob is more prominent than the other. Along with that edges of different blobs/noises are equally prominent at that iteration. Hence, in this method, it becomes very difficult to extract the shape of the blob we are concerned with. The figure 6 below shows the output after every 10th iteration.



(a) Original Image (b) After 1st iteration (c) After 11th iteration

(d) After 21st iteration (e) After 31st iteration (f) After 41st iteration

Figure 2: Example of the difference of Gaussian after every 10th iteration.

Conclusion

This assignment was very helpful in understanding the classical image processing techniques and their application to real-world problems. They seem to work very effectively to obtain the goal. But it is observed that a certain technique/set of parameters works really well for some images whereas fails for other images. It is because of varying noise/lighting conditions in the image. It can be concluded that a certain pipeline/model may fail over large number of images because of the variation in the images. Learning-based models can capture these variations when exposed to a training set which includes almost all kinds of variations. Therefore, if we include a learning-based approach along with these image processing techniques, it can do wonders.