Heuristic Approach 1: Simple Thresholding + Morphological Ops

By using o-contour try to generate i-contour with simple thresholding.

1) Take pixels inside contour-o.

2) Try different thresholds which will maximize the dice score with ground truth.

3) Apply morphological operations to increase the score.

Advantage

- Easy to implement
- Faster than neural net inference time
- Gives good results

Disadvantage

Q: Are intensity distributions are same for each image across all slices and all patients/studies ? If not this will be a problem !
A: Indeed, we saw that it's a problem - a box plot is provided for this analysis.

- Very manual task
- Depends on many assumptions such as pixel intensity distributions being similar between and within images
- Needs post-processing
- Hard to tune for threshold and morphological operation orders/parameters
- Picking the threshold is very hard, variance is too high -> normalization + clustering ?

Evaluation

For evaluation dice score is used to compare different methods and parameters. As baseline , I've set the baseline model to have predictions for i-contours to be same as o-contours. In this way I was able to have a sense of how much the heuristics are helping or increasing the performance.

Sorensen Dice Score:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|}$$

ROI: In this context ROI means pixels inside o-contour.

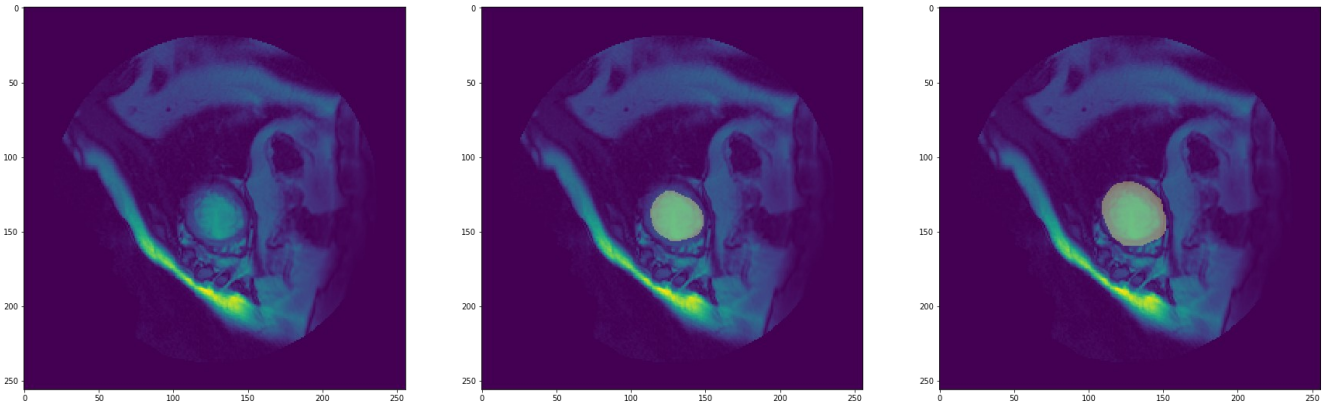

Fig 1. *Raw image, i-contoured image and o-contoured image.*

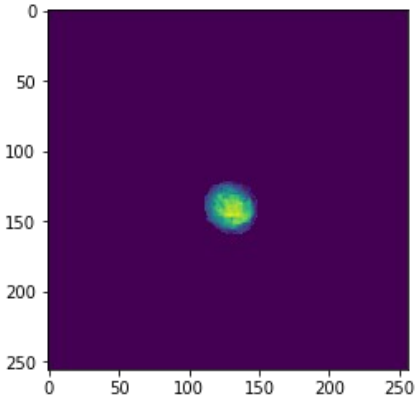

Fig 2. *An example ROI extracted from a slice. It includes all the pixels inside of o-contour.*

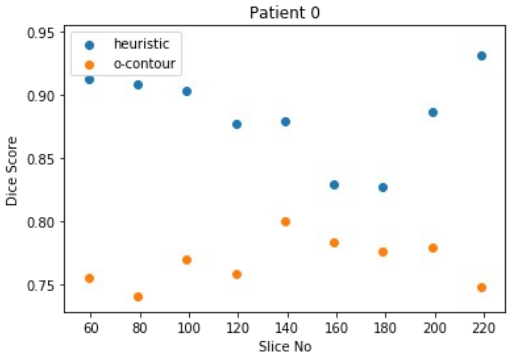

Fig 3. *Simple heuristic dice score compared to baseline (predicting i-contour as o-contour) across different slices for a patient.*

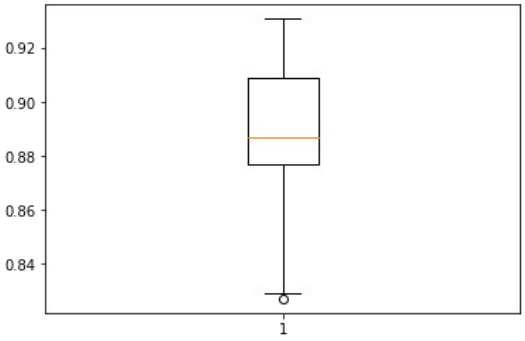

Fig 4*. Box-plot view of Fig 3.*

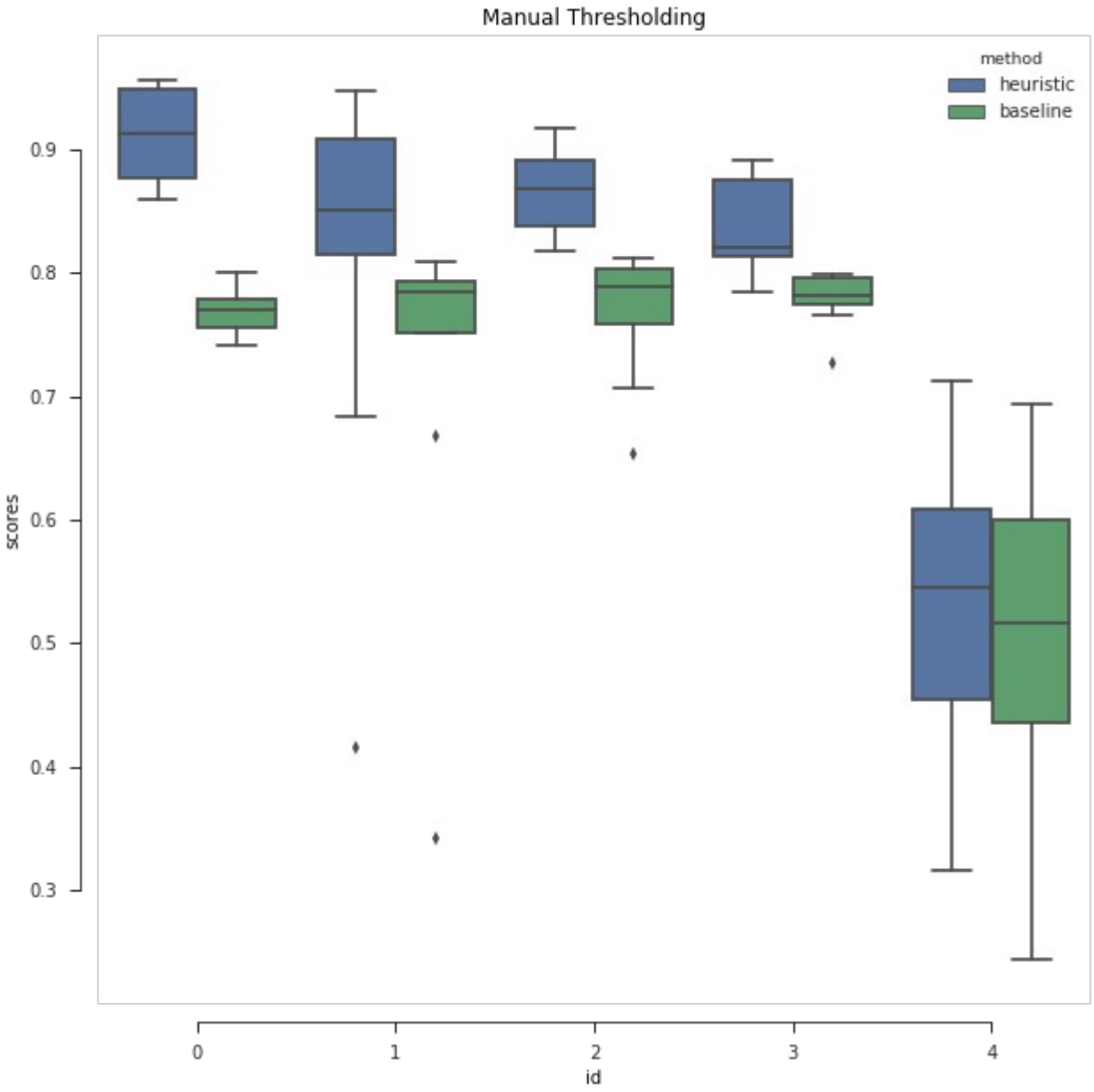

Fig 5. *Dice scores for each patient/study compared with their baseline.*

Above, we see that patient 1 and patient 4 are effected by within and between pixel intensity variance problem. This was the first indicator that actually showed me that a fixed threshold approach is not enough.

	mean	std
method		
baseline	0.707848	0.144136
heuristic	0.790543	0.167922

Fig 6*. Mean and standard deviation comparison all slices combined.*

Further inspection of patient 1 and patient 4

```
Patient 0
Slice 59 Dice Score 0.949
Slice 79 Dice Score 0.956
Slice 99 Dice Score 0.956
Slice 119 Dice Score 0.912
Slice 139 Dice Score 0.894
Slice 159 Dice Score 0.865
Slice 179 Dice Score 0.876
Slice 199 Dice Score 0.916
Slice 219 Dice Score 0.86

Dice Mean: 0.909, Std: 0.04
Baseline Mean: 0.768, Std: 0.02

Patient 1
Slice 60 Dice Score 0.947
Slice 80 Dice Score 0.936
Slice 100 Dice Score 0.908
Slice 120 Dice Score 0.838
Slice 140 Dice Score 0.879
Slice 160 Dice Score 0.851
Slice 180 Dice Score 0.814
Slice 200 Dice Score 0.683
Slice 220 Dice Score 0.416

Dice Mean: 0.808, Std: 0.16
Baseline Mean: 0.722, Std: 0.14

Patient 2
Slice 20 Dice Score 0.868
Slice 40 Dice Score 0.818
Slice 60 Dice Score 0.822
Slice 80 Dice Score 0.844
Slice 100 Dice Score 0.877
Slice 120 Dice Score 0.867
Slice 140 Dice Score 0.896
Slice 160 Dice Score 0.917
Slice 180 Dice Score 0.899
Slice 200 Dice Score 0.835

Dice Mean: 0.864, Std: 0.03
Baseline Mean: 0.768, Std: 0.05

Patient 3
Slice 40 Dice Score 0.891
Slice 60 Dice Score 0.86
Slice 80 Dice Score 0.817
Slice 100 Dice Score 0.784
Slice 120 Dice Score 0.808
Slice 140 Dice Score 0.813
```

```
Slice 160 Dice Score 0.82
Slice 180 Dice Score 0.88
Slice 200 Dice Score 0.875

Dice Mean: 0.839, Std: 0.04
Baseline Mean: 0.778, Std: 0.02

Patient 4
Slice 59 Dice Score 0.712
Slice 79 Dice Score 0.697
Slice 99 Dice Score 0.608
Slice 119 Dice Score 0.592
Slice 139 Dice Score 0.545
Slice 159 Dice Score 0.471
Slice 179 Dice Score 0.455
Slice 199 Dice Score 0.316
Slice 219 Dice Score 0.322

Dice Mean: 0.524, Std: 0.14
Baseline Mean: 0.496, Std: 0.15
```

Patient 0

Patient 0 suffers from the last two slices having very low dice scores, these are the outliers from our box-plot above. This might due to wrong fixed threshold or due to low signal in these slices.

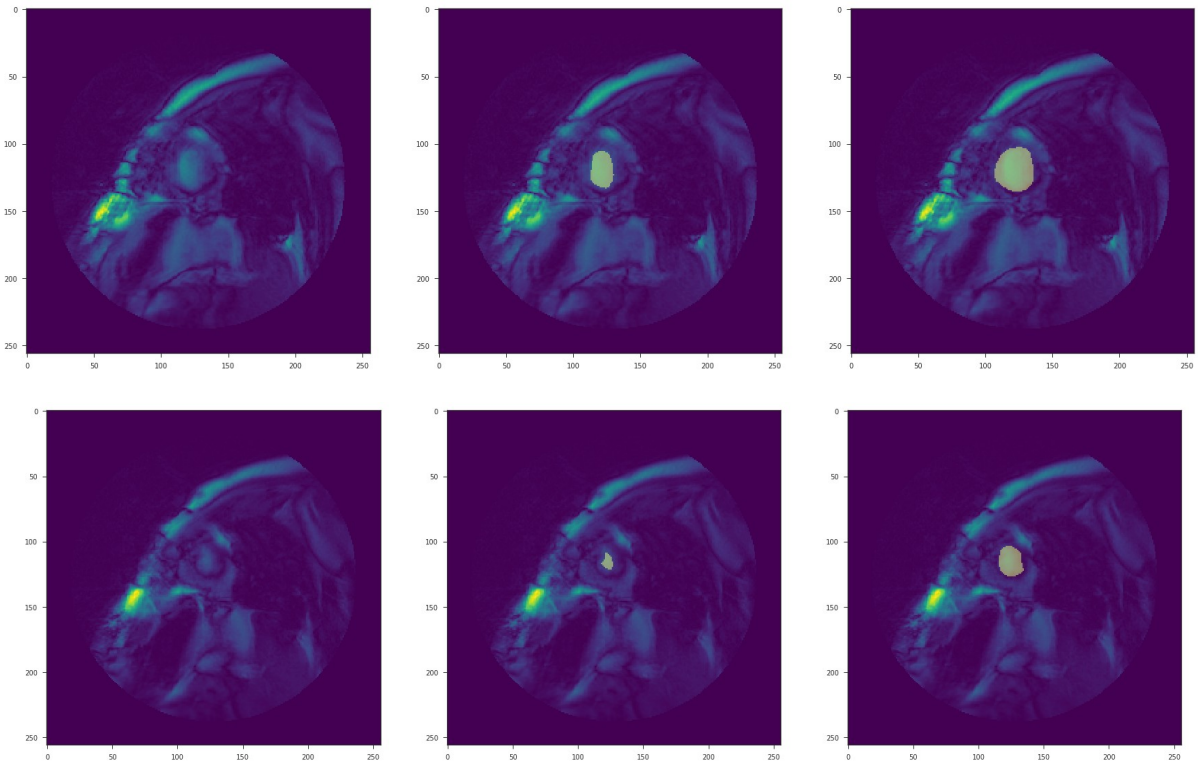

Fig 7*. Last two slices of patient 1.*

Patient 4

Patient 4, as I've mentioned in challenge phase 1, has shifted o-contours and this is causing very low dice score for our proposals since here we use o-contours as our priors for proposal generation.

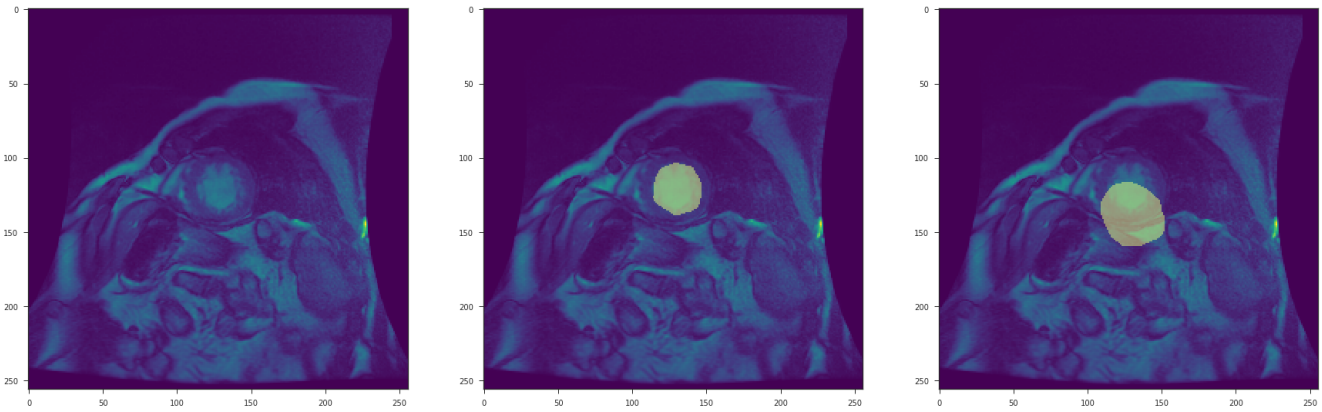

***Fig 8**. Apparent shift in patient 4 o-contour.*

Heuristic Approach 2: Dynamic Thresholding: Otsu

This simple approach and results I've gathered in part 1 motivated me into looking at ROI pixel intensities of each study to understand the variance and come up with a better approach for solving this problem. Below you may see bimodal pixel distributions for each slice and each patient/study. This bimodal behavior is what motivates the Otsu method to find the threshold which will maximize the maximum likelihood. Unfortunately not every slice has a bimodal distribution and this is one drawback of this heuristic approach.

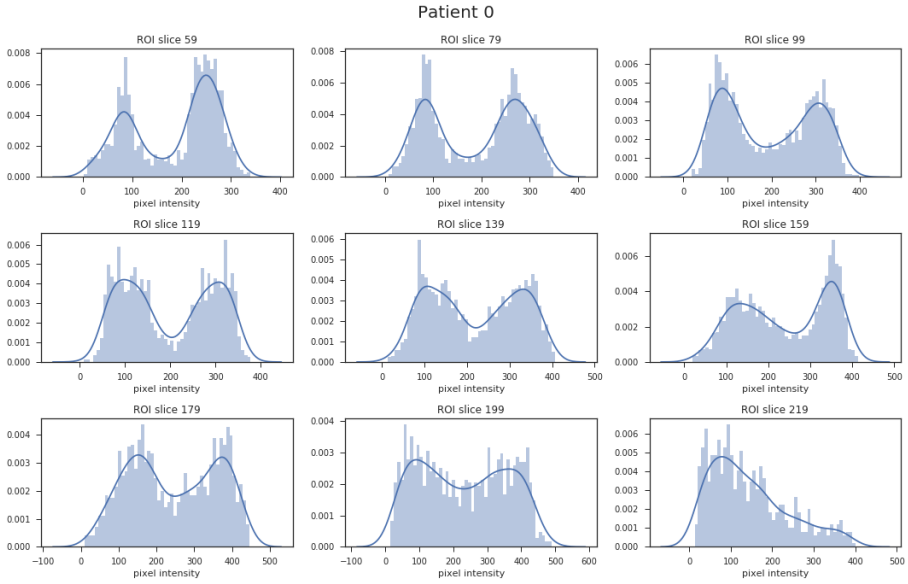

***Fig 9.** ROI pixel intensity densities*

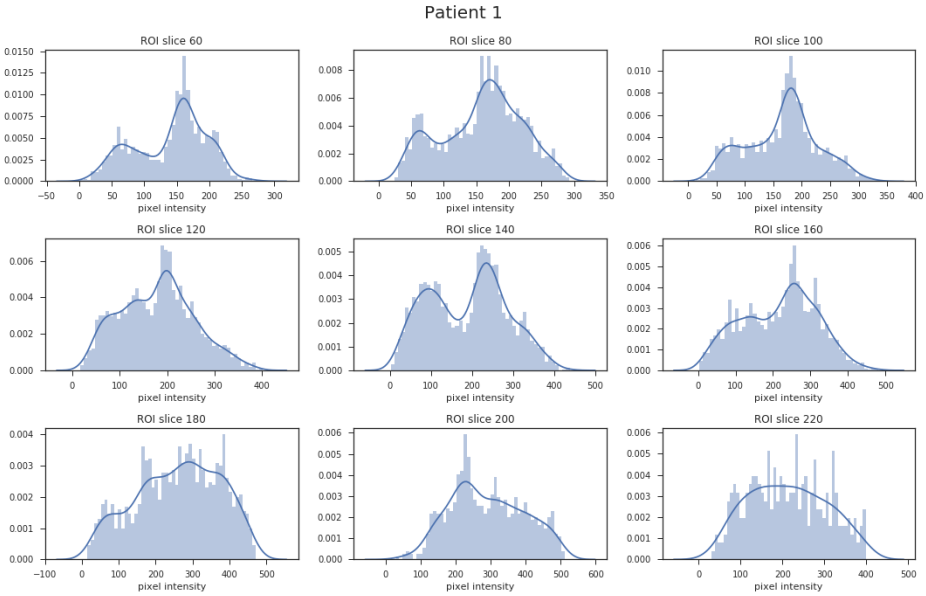

Fig 10. *ROI pixel intensity densities*

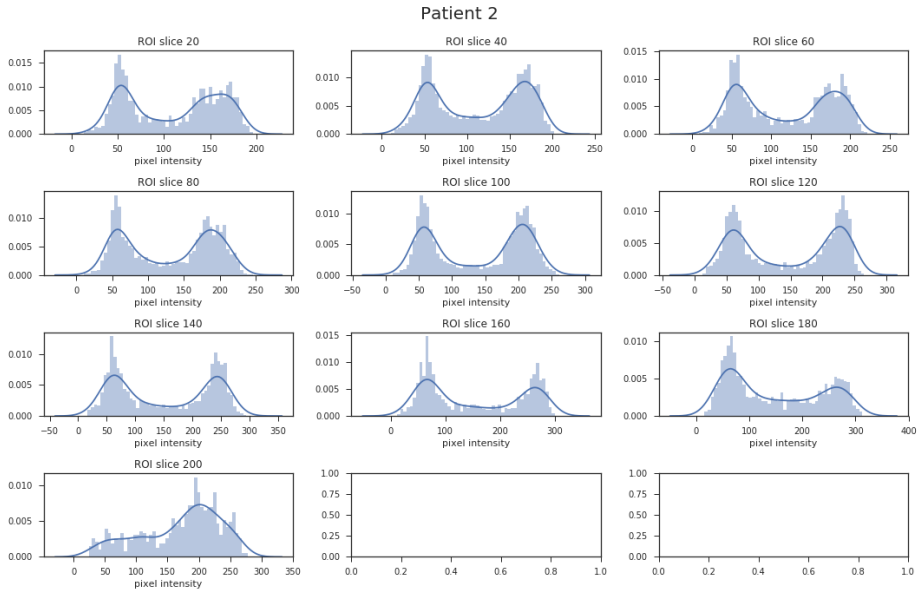

Fig 11. *ROI pixel intensity densities*

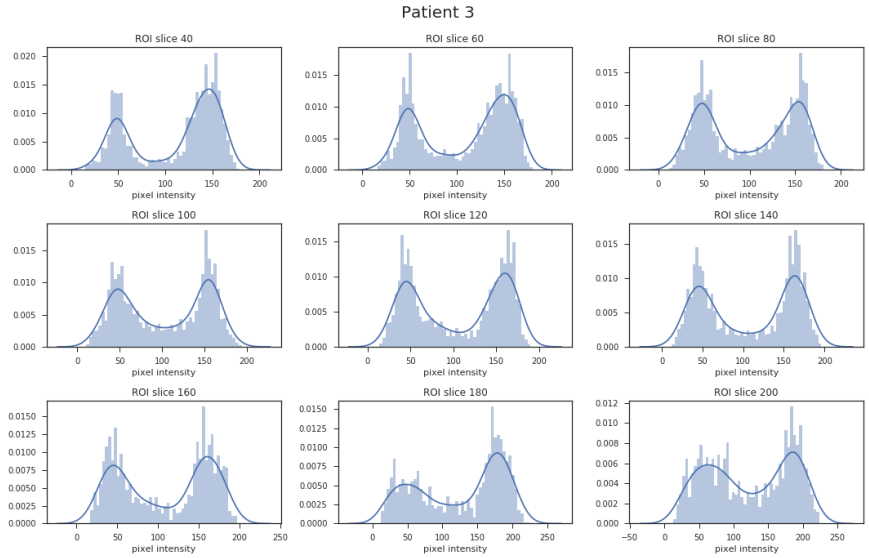

Fig 12. *ROI pixel intensity densities*

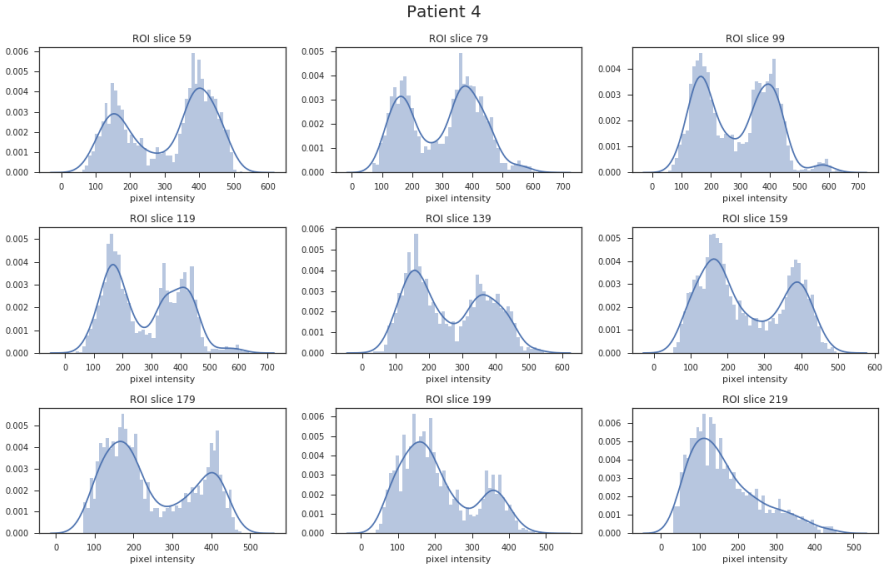

Fig 13. *ROI pixel intensity densities*

Here is an example of the threshold we are trying to find for each slice.

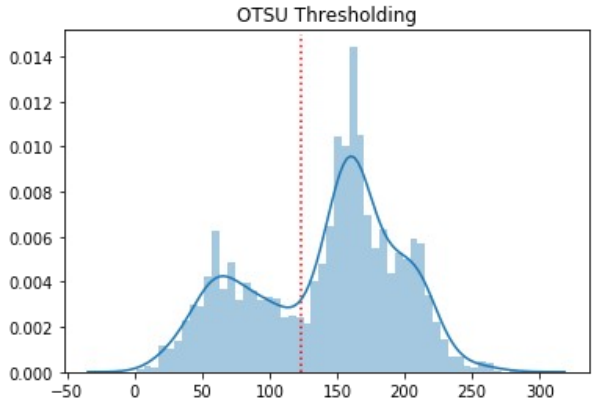

Fig 13. *An example of Otsu thresholding on patient 1 slice 60.*

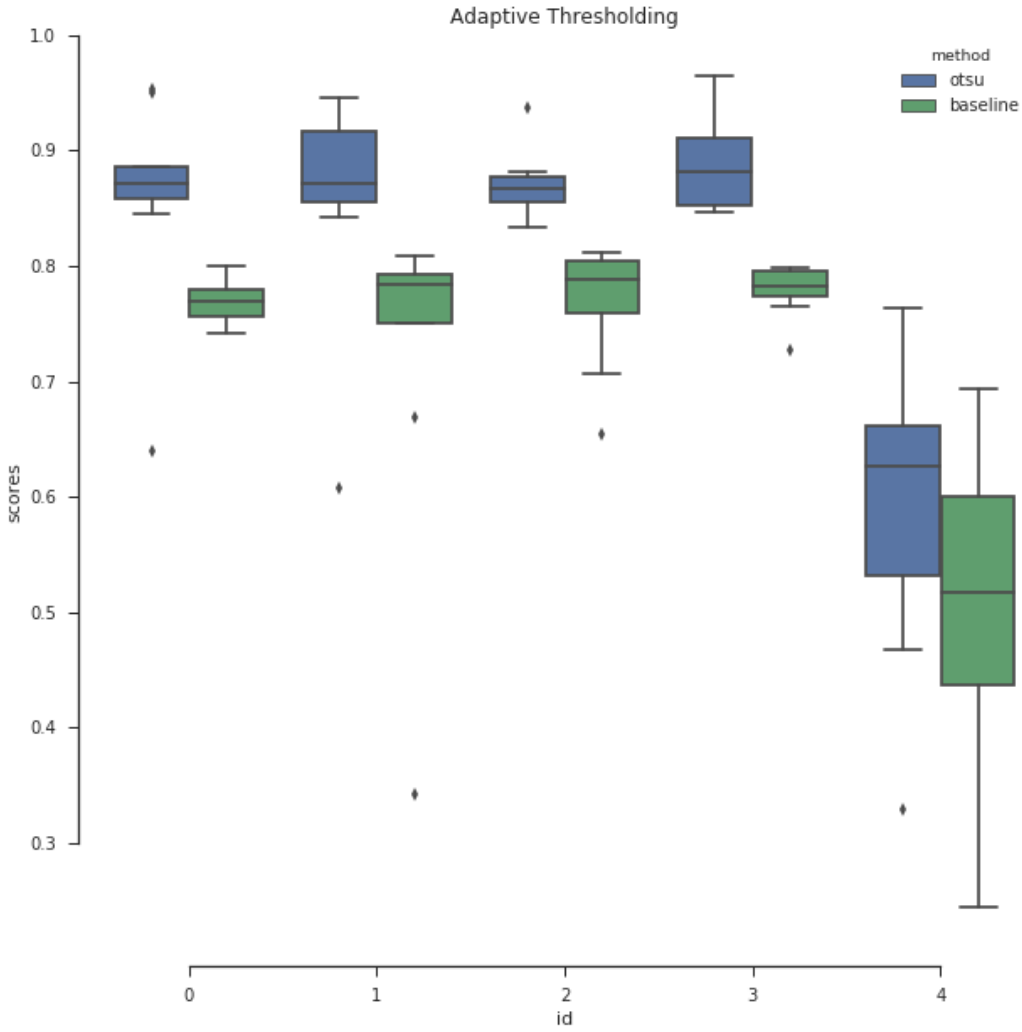

Fig 14. *Dice scores for each patient/study compared with their baseline with Otsu threshold.*

Otsu increases our mean dice score by 2.6% and decreases standard deviation by 2.7 points. Still results are not good enough since we don't have bimodal distribution at each slice as seen in ROI pixel density plots.

method	mean	std
baseline	0.707848	0.144136
heuristic	0.790543	0.167922
baseline	0.707848	0.144136
otsu	0.816391	0.140014

Fig 15 *Summary statistics for different methods.*

In conclusion, we've seen that thresholding in general is working well for this particular case but results are still not satisfactory since for this task human level accuracy can be obtained. Apart from ML based approaches even though I couldn't manage to get it work I still believe edge based methods should do a good job. The idea is to detect the edges from the heart muscle which is obviously seen just by looking at the images. A neural net based approach, a shallow U-Net, would be even better since it will not require a lot of manual post-processing as we would have in edge-based approach and would capture image characteristics like edges, shapes as well as pixel intensities.

You may find my attempts of trying out edge-based segmentation, watershed algorithm and region growing in analysis_development-part2 notebook.

Edge-Based: Might be promising if we can capture edges of heart muscle but would require a lot of post-processing.

Watershed: Seems to work for disconnecting overlapping objects, not very suitable for this case. But wouldn’t know without trying.

Region Growing: Makes sense and suitable for this case but very slow algorithm and still depends on a parameter – threshold for differences of neighboring pixels.