Documentation

2025.02.17. – 2025.03.03.

SNU

Dr. Hatvani, Janka | Csatári, Dominik | Fehér, Márton | Várhidy, Ágoston

Our aim for this 2 week was to complete the following tasks:

1. Create scan conversion of ultrasound images, then also create the reverse function
2. Convert curved image to flat image
3. Find highest intensities on reverse scanned images
4. Create a robust pipeline

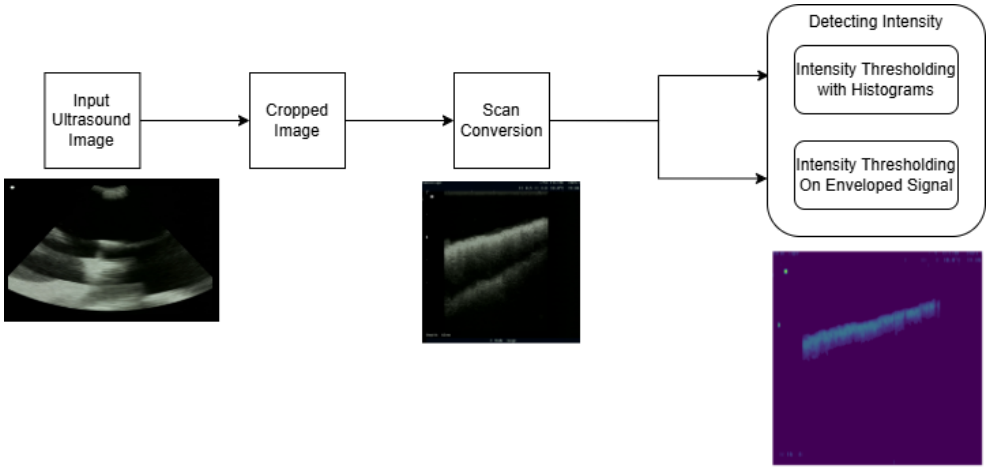

Figure 1 Shematic pipeline of our progress

1. Scan conversion

For creating the conversion, we needed to crop the image and find the angle of which the ultrasound image was created.

For cropping the image, we have created a kernel, which detects a structure ont he upper left corner of the image. This detectation procedure was done with template matching. After this, as the size fo the region of interest is fixed, we can crop the region. This procedure was tested on multiple images to ensure persistence.

For finding the angle, multiple ideas came to mind. To find the most robust solution, we have tested all the ideas. The two most promising ones were done by Monte Carlo simulation and Hough Transformation.

a) Monte Carlo Simulation

The algorithm first defines a region of interest of the rows it might use. This region is fixed. Then, it randomly samples two rows and selects the first two points in those rows that exceed the threshold. After this we define a line by these points and we calculate the angle of it. We repeat this procedure *n* times. After this, we can create a histogram of the most frequent angles and choose the most frequent degree.

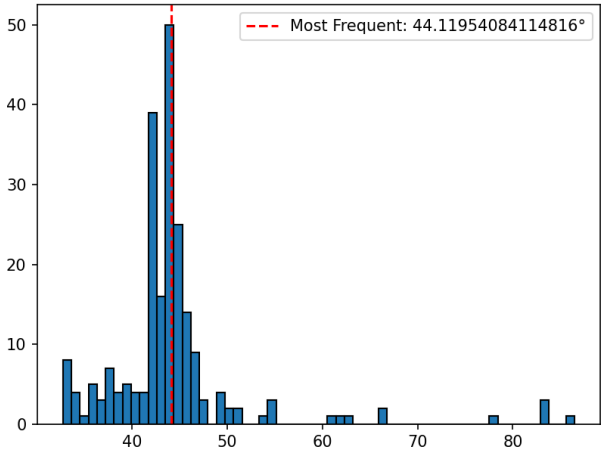

Figure 2 Estimating the angle with simulation

b) Hough Transformation

For this method first we have detected edges on the original image with Canny edge detection. After this, on the edge image, we did the Hough transformation and defined a minimum length for the detected lines. If none was found, we have decreased the threshold. We repeated this procedure till there was a line to detect. On *Figure 3* the detected angle was 44°.

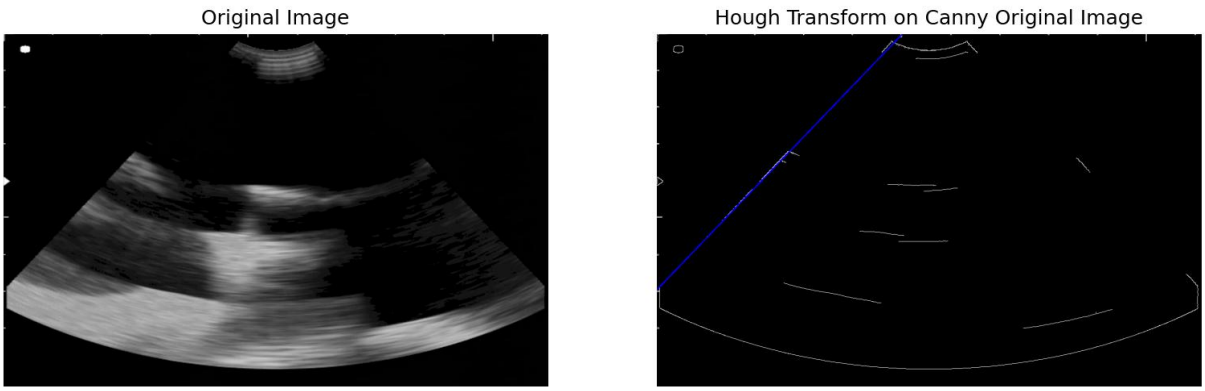

Figure 3 Original Image and the detected line on the Canny edge detected image.

As *Figure 2* and *Fiugre 3* shows, the detected angles are close. However, the second procedure seemed to be more robust, so we decided to use that.

2. Convert curved image to flat image

After identifying the cropping area and estimating the scan angle, we transformed the curved ultrasound image into a flat, Cartesian format. This transformation was necessary to reduce geometric distortion and allow further analysis on a uniformly scaled image.

We began by locating the centimeter calibration marks both horizontally and vertically on the grayscale image. These marks, visible as white ticks, allowed us to determine the pixel-to-distance ratios in each direction. Based on this information, we calculated the actual height and width of the scanned area and identified the offset from the transducer to the start of the usable image.

Using a fixed angle of the ultrasound device (84°), we computed the corresponding angular span in radians and generated a polar coordinate grid covering the relevant radius range and angular field. This grid was then mapped to Cartesian coordinates. To retrieve the corresponding intensity values from the original image, we applied bilinear interpolation, estimating each pixel value from its nearest neighbors.

As seen in Figure 4, the curved scan is effectively transformed into a flat image, preserving the spatial structure and intensity information of the original data in a more usable form.

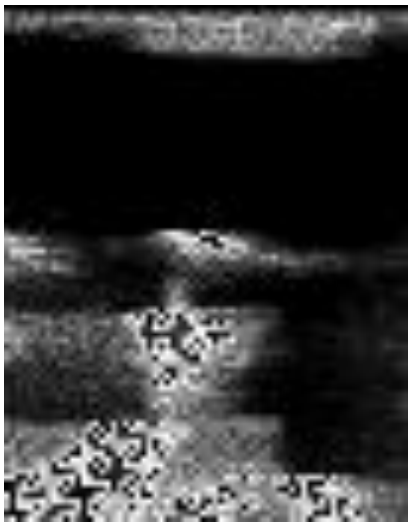

Figure 4 Flattened image after polar-to-Cartesian transformation

This transformation was implemented in Python (transformation_curved_to_flat.py) and is available in the project GitHub repository.

3. Find Highest Intensities

After converting the image, we can identify the highest intensities, which correspond to the scatters of interest—where the ultrasound interacted with an object rather than producing artifacts. For this, we have tried two solutions: Intensity thresholding with histograms and thresholding on smoothed signals. The results for the first solution can be seen in *Figure 5*. The original image can be seen in *Figure 1* under Scan Conversion. This solution produces artifacts. This can be improved by implementing clustering solutions to keep the biggest regions. Instead, we have tried out different solutions.

Figure 5 Thresholding Histogram

We have also tried to find high intensity regions by smoothing the signals and finding the highest intensity point in the given column. After that we also kept the surrounding area of it. One of these column can be seen on *Figure 6*.

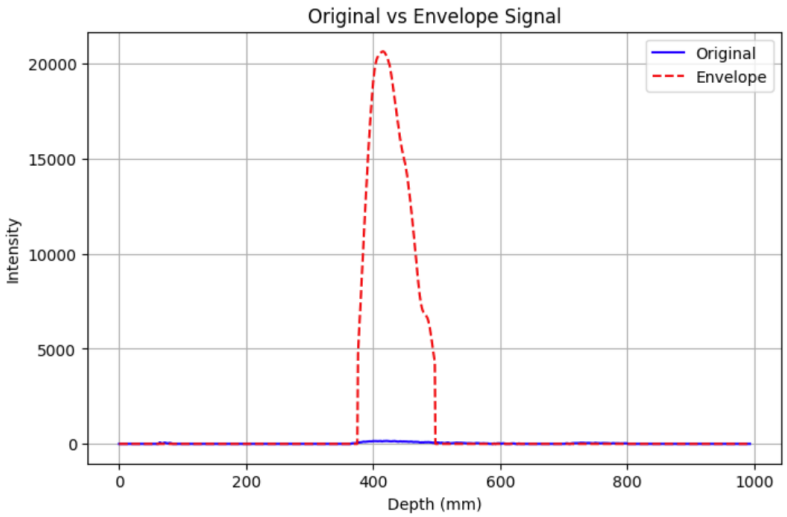

Figure 6 Original vs Envelope Signal

Repeating this procedure for all the columns resulted with the image shown in *Figure 7*. The *Output Image* has artifacts as the procedure was tested on an iamge which was not cropped. However, this does not impact the evaluation, demonstrating that this technique can be applied to all other images as well.

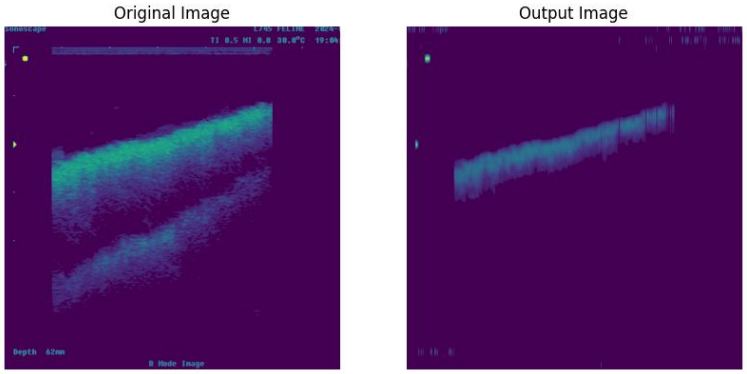

Figure 7 Thresholding the smoothed signal

4. Creating the pipeline

We have worked on these tasks in Python and focused on creating functions for all subtasks so later on these can be combined creating a pipeline as shown in *Figure 1*. The codes we have created and the images we have used for testing can be found on GitHub(https://github.com/csataridominik/SouthKorea2025).