The data for each patient consists of multiple paper templates, which are filled in with pen by both their treated and untreated hand before undergoing FUS treatment, and then again at various stages after undergoing FUS treatment. These papers are then scanned in as PNG files.

This is a typical example of a patient’s data, which more often than not includes missing files.

For demonstration purposes going forward, we will be focusing on one patient in particular. Specifically the data captured with their treated hand both before and after receiving FUS treatment . It is quite clear that this patient shows great improvement.

The data pre-processing took place in four different stages.

First, after talking to our supervisor, we learnt of Kelvin da Silva – a past student of Wits who developed code involving a similar dataset and problem statement for his master’s research project in 2020.

Kelvin graciously allowed us to use one of his scripts that extracts the two spiral images from each of the patient’s templates.

Text detection and other deep learning based methods are not within the scope of this project and so it was decided to implement and improve Kelvin’s code instead of trying to write our own from scratch. All copyrights and required credit is given in each of our scripts.

Unfortunately this code needed to be improved to meet our requirements. It is also a very complicated method used to perform quite a simple task.

This code detects the position of the three known text elements on each image. From here it can crop both spirals using their relative positioning.

The code was then expanded to include the extraction of the line-block C as shown.

Further improvement included removing extra detected text by checking each general position. And only keeping the “Drawing A”, “Drawing B” and “Drawing C” text as shown in the second image.

These erroneous text blocks detection are caused by the many blurs and markings from the poor quality of scanned documents.

The original code relied on the position of “Drawing A” and “Drawing B” exclusively to crop out spirals A, B and line-drawing C. It would not work without these two positions.

However in some instances, “Drawing A” was were not detected. Thus, to try compensate for this, the position of “Drawing B” and “Drawing C” are used to crop out the required images instead.

The same can be said for when “Drawing B” text is not detected…

And when “Drawing C” text is not detected.

The final and most complicated part of the data pre-processing involved figuring out how to compensate for rotated or warped scanned documents.

This issue mainly effected the methods that delt with the line drawings, as the spirals are circular and so rotation does not have as much consequence. As seen, rotation without correction results in a poorly cropped line drawing.

To improve this, OpenCV shape detection was used to detect the position of the two top-most rectangles, as shown in blue. It was then simple to crop the line drawing more accurately.

Finally, to compensate for any rotation or warping, the 𝑔𝑒𝑡𝑃𝑒𝑟𝑠𝑝𝑒𝑐𝑡𝑖𝑣𝑒𝑇𝑟𝑎𝑛𝑠𝑓𝑜𝑟𝑚 and warpPerspective functions from OpenCV are used to correct the final cropped image.

It was decided that another method would be implemented to compare final results and better determine the accuracy of each method.

This method analyses the line images. First, each image is rescaled to be 600 pixels in width. Then basic noise reduction is implemented by converting each image to greyscale and mapping each drawing pixel to black.

The positional index of each pixel is saved and extracted from a multidimensional array. This results in two arrays, one with all x-coordinates and the other with all y-coordinates. These y-values are shifted down so that the line is centered around the x-axis as shown in the final image.