Team project

Tobias Blatter, Lionel Rohner, and Thibault Schowing

Introduction to Signal and Image Processing

June 11, 2020

Introduction

When the cochlea, the spiral like part of the inner ear, is no longer able to stimulate the auditory nerve thus provoking severe-to-profound deafness, an implanted electric system is able to take over. This implant is built in two parts: an external part, consisting of a microphone, batteries, signal processing units and transmission coil, and an implantable part consisting of a receiving coil, a receiver-stimulator unit, the reference electrode and, what is the most interesting for this project, the electrode array.

With the angular insertion depth θ of the electrode it is possible to estimate the perceived frequencies for a particular patient. To achieve this, a preoperative and a postoperative Computed Tomography (CT) scans like shown in Figure 1 are used. By detecting the cochlear center and the electrodes array we can assign an angle to each electrode around the center, from the outermost to the innermost electrode. The higher the insertion angle of the last electrode, the lower frequencies the patient will be able to ear.

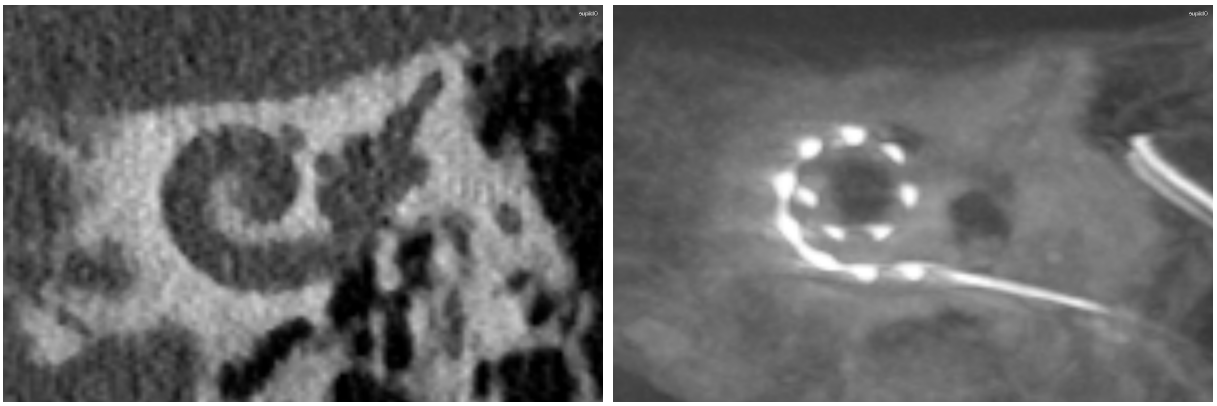

Figure 1: Preoperative and postoperative Computed-Tomography (CT) of the inner ear before and after insertion of an electrode array.

1 Methods and intermediary results

An object oriented approach has been chosen for the program structure as it allows an easier teamwork on a same class. The program is divided in a relatively small number of files:

- **config.py** Configuration file of the program, allows to tune the workflow or diverse function parameters from one place.
- **utils.py** Contains usefull functions not specifically related to a class e.g. debug functions, string processing functions,... etc.
- **Collection.py** Class collection, create all the CTscanPair when given the corresponding path to the images. Contains getters to retrieve the data in custom forms.
- **CTscanPair.py** Class containing the pair of pre-surgical and post-surgical images and doing all the detection work on them when initialized by the class Collection.
- **main.py** Main script, create a *Collection* of *CTscanPair* and use the function provided by these two class to extract the calculated information.

A *pattern* folder contains the pattern used to detect the cochlea, the *GEN_IMG* folder contain images generated by the program (usually a parameter *save_img* can be set to True in interesting functions) and the *DATA* folder contains the CT images.

Note: The code is rich in comments and annotations. Do not hesitate to read functions of interest. Functions header have been made in order to be able to generate an automatic documentation, functionality unfortunately not available in the Educational edition of PyCharm, which was used in this project.

Note: The program was mainly coded and tested on **Windows** and might produce unexpected results when executed on **Mac-OS** or **Linux**, but only when displaying images is set to `True`.

Part I: Cochlea detection

To detect the cochleas' center from the pre-surgical CT images, it is first necessary to process the images in order to get only the liquid regions, corresponding to a gray area, in contrast to brighter region matching with denser tissues like bones, and darker region matching with the density of air. After an exploration of the images composition, the ideal *blurring* and *thresholding* parameters are applied to the images in order to get a black and white binary image like in Figure 2.

Once the image arrays with good contrast are created, they can be used to detect the center of the cochlea. For this task, two methods were tried: **Hough Circle Transform** and **template matching**. Hough Transform has immediatly shown poor and unstable results, giving many or no circles, when applied to different images. However, Pattern Matching gave fairly good results.

Pattern matching uses a circle as pattern and searches the image for a best match. In order to be adaptable to eventual differences between images, the image is resized with different percentages to allow size difference of cochlea. A simple circle has been chosen here and gave good results, but a more complex pattern can always be used. In our case, to use different patterns, for instance spiral-like patterns, it would be needed to test for both orientations of the cochlea. Here, the center of the cochlea is considered to be the center of the circle with the best match. Note, the size of the pattern approximately matches our actual cochlea, thus images with too small cochlea would probably not give optimal results.

Figure 2: Presurgical images after processing in order to have a good contrast between the area matching with liquid and the others. Three images from the data set are shown.

Results for center detection are shown in Figure 3 where the cochlea is colored in blue. Now that we can extract the center and cochlea area, we can use this to create a mask allowing to select the specific zone of interest in the post-surgical images as shown in Figure 4. The goal of this was to avoid the white pixels of the scan number on the images' corners that were making the contrasting more difficult.

Improvements: To increase the precision to calculate the electrodes' insertion depth, it would be interesting to get a "center" based on the average of the deepest electrodes after their detection. Indeed, the actual center is hard to define, not only mathematically but also when looking at the pictures without medical expertise. Furthermore, a center more "centered" according to the electrode array's position (in the spiral) would make the ordering, which is the most sensitive step (see Part III: Electrodes ordering), stronger.

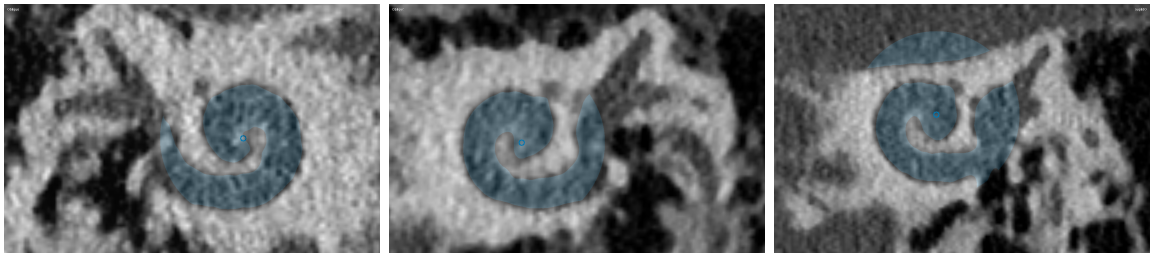

Figure 3: The cochlea is here colored in blue, with a blue circle as the center. As you can see, the blue part correspond to the overlap of the mask in Figure 2 and a circle with an arbitrary diameter around the cochlea, big enough to include the cochleas' basal turn but too big so in also includes non desired areas. As it is a binary mask, some algorithm (Island detection, propagation from the center,...) could be used to remove the smaller blue areas that are not part of the cochlea, but this was not considered a priority for this short project.

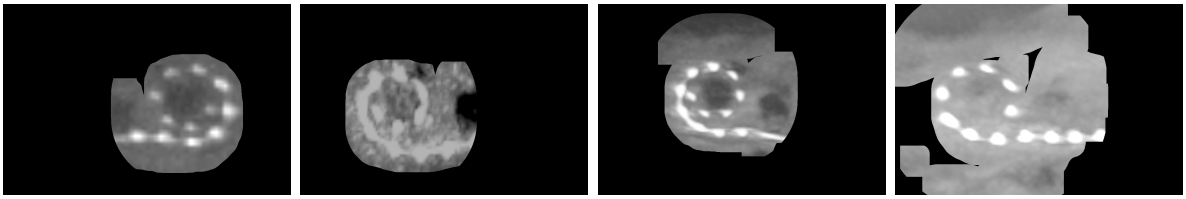

Figure 4: Area of interest extracted from the post-surgical images with the cochlea center and a configurable circle radius. For some images, like *ID06* (most right picture), the electrodes are not inserted deep enough so all of them appear on the image. In this case, a bigger radius can be set to expand this area.

Part II: Electrode array detection

The next key step of this project was to detect individual electrodes that have been implanted in the cochlea. The electrode array consists of 12 electrodes that should ideally be evenly distributed along the spiral shape of the cochlea. Since the electrode array is metallic, it appears as bright spots connected by a wire in the postoperative CT image. The algorithm that was implemented to detect the electrodes and their corresponding center points involves normalization and thresholding of the images followed by connected-component labeling (CCL) and contour recognition (Figure 5). The algorithm can be found as a class method (`setElectrodesCoordinates(self)`) in the file CTscanPair.py

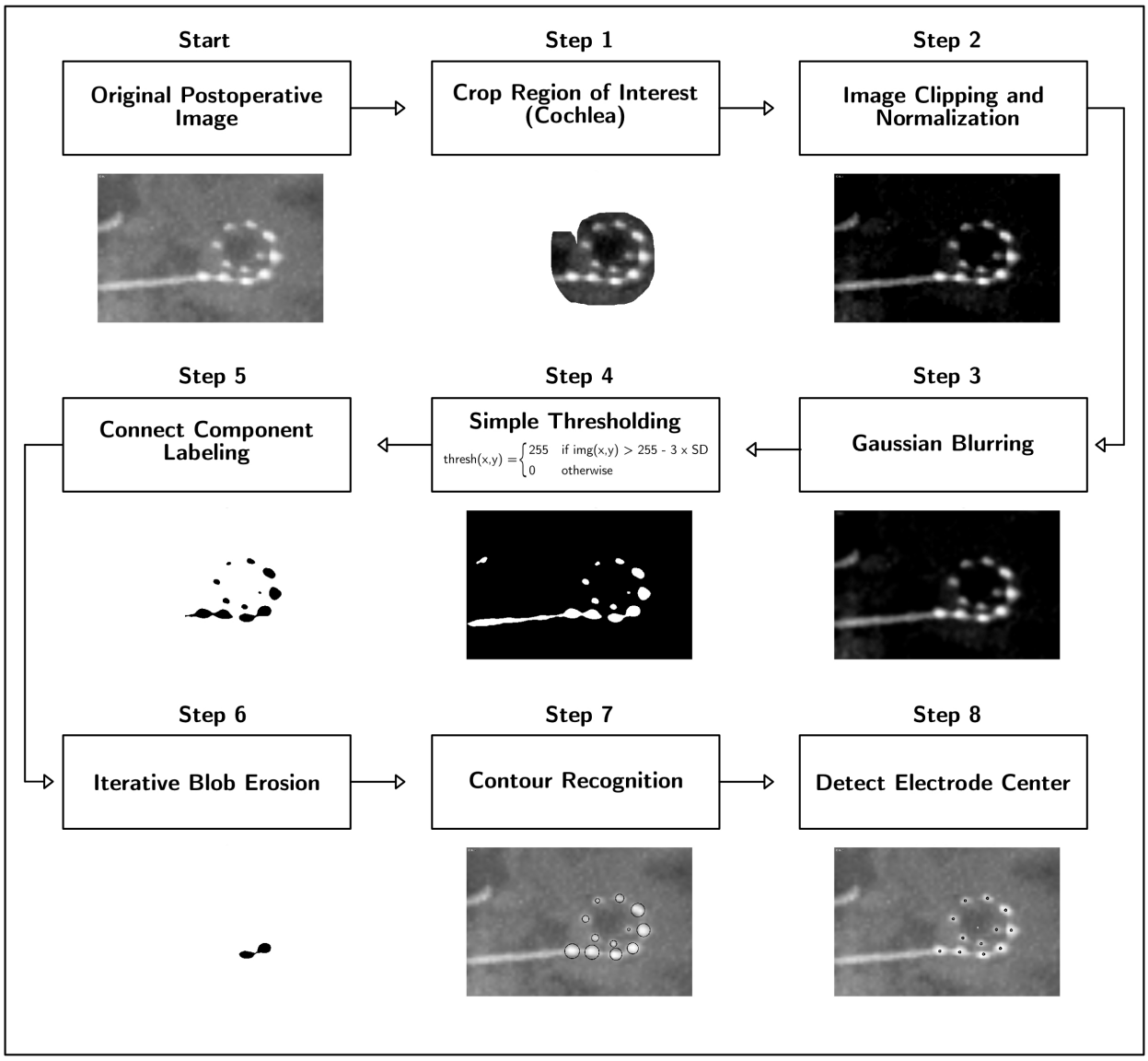

Figure 5: Flow chart of the main steps of electrode array detection. The algorithm is shown as a series of 8 consecutive steps. For each step, a representative image is depicted to demonstrate how the postoperative image array is processed to detect the electrodes.

Since the pre- and postoperative images are registered, meaning that both have the same coordinate system, we defined an area with a certain radius around the previously detected cochlea center to individually pre-process the image for electrode detection. Most of the important information for image normalization is found within a certain area around the cochlea, namely the electrodes and the background surrounding them. In order to enhance the intensity of the electrodes and to minimize the noise around them, we clipped the gray-scale intensities of the postoperative image array according to the mean and standard deviation (SD) of the area of interest (AOI), which is the cochlea (Step 1). Array clipping leads to an image that has minimal and maximal intensity values that are within 1 standard deviation (SD) of the mean, i.e. in the interval [mean(AOI) - SD, mean(AOI) + SD]. Clipping is followed by normalization using OpenCV2 with normType `cv2.NORM_MINMAX` (Step 2).

The normalization procedure re-scales the intensity, such that the maximum value is 255. The clipping and normalization procedure based on an AOI within the image allows individual normalization without the need of a hard coded threshold. This is preferable for the provided data set. Subsequently, Gaussian blurring is applied to decrease high-frequency noise and allow smoother thresholding (Step 3). We continued with the segmentation of the image with the application of a simple thresholding method (Step 4). The following criteria were selected for thresholding:

$$\text{thresh}(x, y) = \begin{cases} 255, & \text{if img(x,y)} > 255 - 3 \cdot \text{SD(img)} \\ 0, & \text{otherwise} \end{cases}$$

After thresholding, we apply the morphology operator `cv2.erode` and `cv2.dilate` from OpenCV and also set every pixel to 0 that are not within the cochlea, in order to exclude any pixel blobs that are not electrodes. To detect coherent blobs of pixels, we perform CCL using SciPy's *measure* module (Step 5). CCL is a graph theory-based approach that consists of identifying and labeling connected components. The blobs are identified based on the pixel neighborhood and their similarities, meaning that pixels belong to the same blob if they are adjacent and if they have the same value. In our case, every pixel has either a value of 0 or 255, which corresponds to the background or ideally an electrode, respectively. The blobs enter a loop that we have coined iterative blob erosion (Step 6).

This procedure consists of adding a blob to a binary mask if its size is smaller than 50 pixels and not bigger than 3000 pixels (Figure 6A). These values have been estimated empirically based on the available data set. By enabling *verbose* mode, one can see that the average blob size for the provided data set is 1775 pixels with a SD of 268. The lower boundary has been set rather low as some blobs get very small due to low signal intensity and erosion filtering. Moreover, we do not expect that small false-positive blobs are included as pre-processing and erosion filtering minimizes their occurrence. If a blob is bigger than 3000 pixels, we assume that the object is an aggregate of blobs and erosion filtering is applied iteratively until the blobs are separated (Figure 6B).

The last steps of the algorithm involve recognition of the contours by OpenCV2 (Step 7) and calculation of the centers of the electrodes (Step 8). The coordinates of the electrode centers

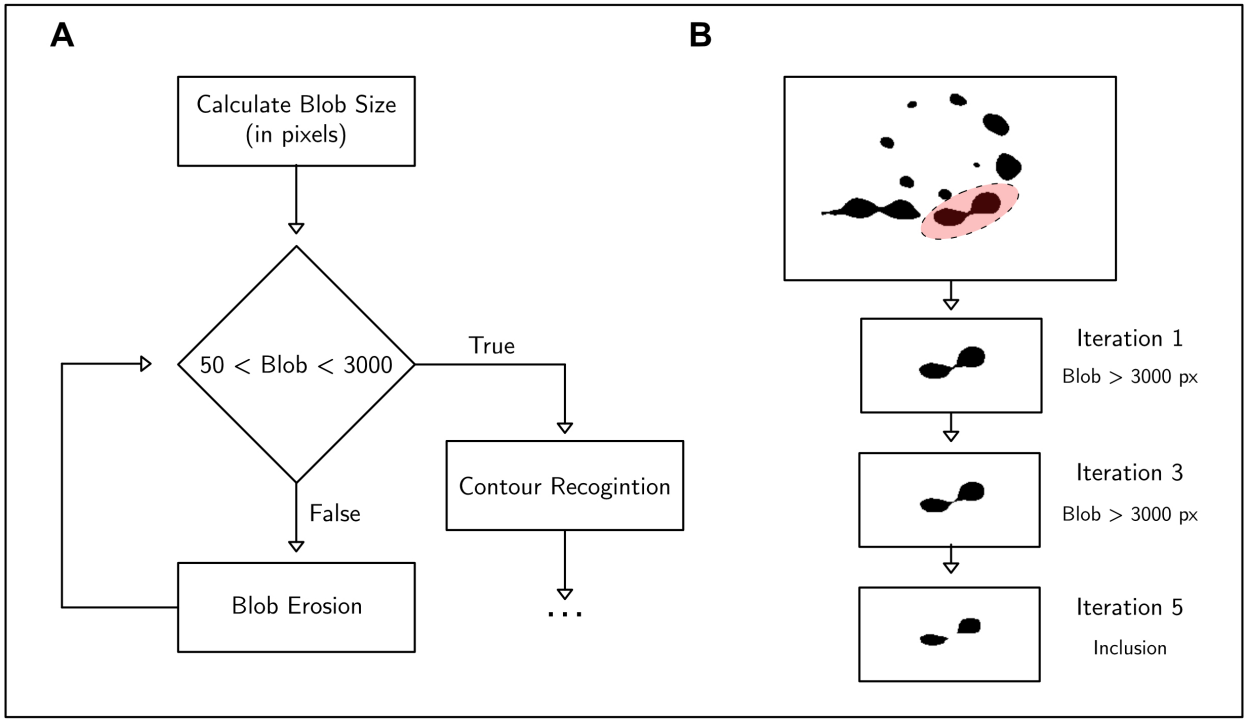

Figure 6: Simplified decision structure of step 6 with an example. (A) Paths in the decision tree illustrate how blobs are handled and filtered. **(B)** An example of the erosion filtering of a blob aggregate from image ID03 is depicted. After 5 iterations, the blobs are separated and pass the size inclusion criteria, which allows to detect them as 2 individual blobs based on the contour recognition.

are stored in a list, which is used to compute the order and the angles of the electrodes to the 0-degree reference line: The line between the electrode #12 and the center of the cochlea.

Results: In general, our electrode detection algorithm works quite well. However, it all depends on the quality of the postoperative images. We were able to correctly identify all 12 electrodes in 7 of 12 images (see Results). However, some issues could not be circumvented. The algorithm performs poorly if the signal of multiple electrodes overlaps to a great extent and/or the signal of the wire distorts the round shape of electrodes. The problem here is that erosion fails to separate the electrode blobs and rather treats the aggregate as a single object (Figure 7), therefore the center is placed in between two blobs (Figure 12 and 15). Another problem that occurred, was, that weak signals were not detected at all (Figure 17 and 18).

Improvement: There are several steps of the algorithm that require improvement. First, the normalization procedure is rather conservative. There is a clear trade-off between finding all electrodes and the introduction of noise and favoring only strong signals but missing weak electrodes. To optimize this other normalization procedures could be explored. Alternatively, working with a larger data set could be very useful as well. With the inclusion of more postoperative images, some features and reoccurring nuisances might be selected for

or avoided based on statistical information of the images.

Using a larger data set might also be helpful for the optimization of the iterative blob erosion (Step 6) as the estimates of the mean and variance of the blob size could be calculated more accurately. This step could also benefit from performing a second CCL within the innermost loop that keeps on eroding the blobs until they fit within the range of accepted values. In this part of the code, the size of the pixel blob is calculated for all the black pixels in the binary image, regardless of whether the blobs are still connected or not. Performing a CCL after each erosion would allow us to correctly calculate the size of individual blobs. At the moment we overestimate the size of the blobs because we cannot distinguish between blob aggregates and unconnected blobs on the same image.

We used OpenCV's `cv2.SimpleBlobDetector` to implement another electrode detection algorithm, which allows detection of blobs based on area and shape parameters, such as circularity, convexity, and inertia, all of which may be very useful if tweaked appropriately (see `new_version_setElectrodesCoordinates` in CTscanPair.py). Although we were able to find the innermost electrodes in image ID16 and ID18 (Figures 16 and 17) with this approach, the results of the angle calculation were less convincing (not included in this report).

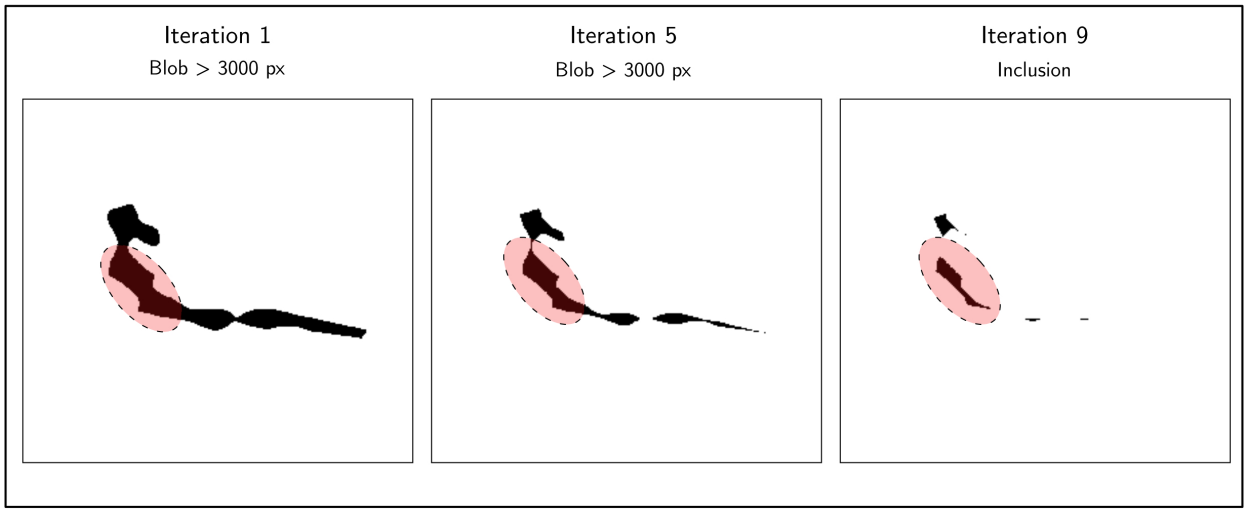

Figure 7: An example of failed erosion. The example was taken from postoperative image ID05 (see Figure 12). The region that fails to be separated into two blobs is highlighted in red. The highlighted blob aggregate fails to be split into two unconnected blobs, because the round shape information of the electrodes is hidden by the strong signal of the wires. If the concavity between two electrodes is too weak, the kernel erodes along almost parallel lines on opposing borders (compare iteration 5 and 9) until the aggregate is thin enough to be included (iteration 9).

Part III: Electrodes ordering

When the cochlea is imaged by CT scanning, the shape resembles a logarithmic spiral. This shape is reflected in the pattern of the inserted electrode array displays. The array wounds around the cochlea center in around 1.5 turns. Logarithmic spirals can be easily described in polar coordinates r and θ as $r = a \cdot e^{k\theta}$. However, with this basic formula, natural spirals can only be approximated roughly. Often times, and most certainly in our case, these spirals do not contain the crucial property of "self-similarity" of true logarithmic spirals. This would allow for the validity of the simple mathematical formula. Additionally, the outermost electrodes often do not lie on the proposed path of the spiral. With this, algorithms for non-linear line detection are not a good choice to annotate electrodes efficiently. Other approaches, like snake line finding proved too complicated to implement. But, with the limited amount of electrodes (only 12 per array), a simple sorting algorithm might just do the trick.

The idea was to convert the spacial information of the detected cochlea center (see Part I: Cochlea detection) and of the corroded centers of the twelve detected electrodes (see Part II: Electrode array detection) from the euclidean space into the polar space. With a simple conversion function, the radius and angle of each electrode is calculated in relation to the center of the cochlea (see `utils.polar`). This allows a very important and initial detection of the outermost electrode (the one with the biggest r-value) and this information can already be used for determining the way the spiral is wound. If the outermost electrode is left and below the center, the spiral is wound *clockwise* (CW) and if right and below the center, it is wound *counter-clockwise* (CCW). It turned out, that this way a very efficient way to detect cochlea orientation.

The algorithm for correctly enumerating the electrodes following a basic spiral shape inwards works by iteration through a sorted list of electrodes (biggest to smallest radius). As previously stated, the outermost electrode is easily detectable. The next electrode in the sequence almost always has a smaller radius and the angle θ increases either strictly monotonically (CW) or decreases strictly monotonically (CCW). By iteration, if a set of conditions are met, the electrode in question is appended to the correct sequence and the conditions for the next testing are changed according to the appended electrode. If the conditions are not met, the electrode in question is appended to a buffer (also a list). After each iteration, the buffer is checked to see if an electrode there would be a good fit as a next electrode in sequence (see the function `utils.electrode_sequence`). After this, the total angular insertion depth is calculated using a simple calculator function (see `utils.calculate_angular_insertion_depth`).

Proof of concept is checked by analysing a strictly synthetic picture (see Figure 8). In this picture, both the radius and the angle change monotonically when the sequence of electrodes is followed from the outermost *#12* to the inner-most electrode *#1*. It can be seen, that the algorithm works for both CW and CCW spirals.

Testing on dataset: When adapting to the provided data set, this algorithm was modified

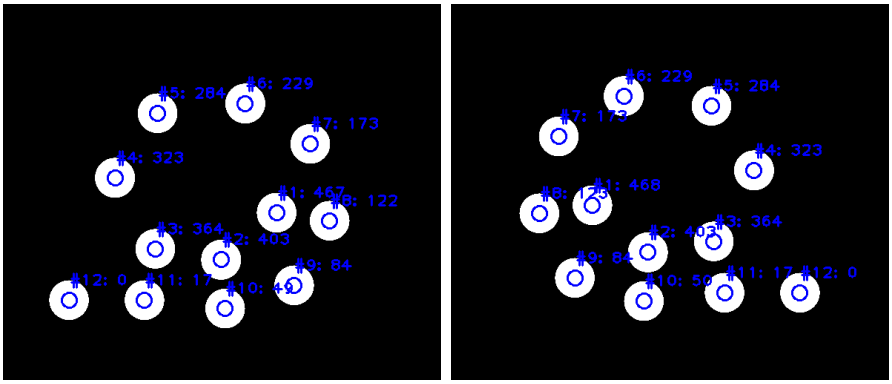

Figure 8: Proof of concept: For both clockwise (left) and counter-clockwise (right) electrode arrays, the correct order and the same angular insertion depth angle is calculated.

to accept non-monotonic changes when looking for the next electrode. For instance, the angle between neighboring electrodes was allowed to be as much as 75° and the radius was accepted in a range of $[0.59, 1.5]$ of the radius before. This worked perfectly fine for the best results, here both center and electrode detection was superb (see Results), but overall with this algorithm, only 90% of all electrodes detected were annotated. Due to the way this function is set up, it always checks electrodes furthest away from the center. After adding an electrode to the sequence, the function does not "look back". The initial sequence given to the function is a sorted list by radius length. If the detected center is more than slightly off the actual center, this results in missing annotation of already detected electrodes (see Figure 9 for details).

Improvements could be done by calculating the approximate angle between the set electrode and the next in sequence. This would remove the need to set a fixed `max. angle`, which can result in poor judgment criteria (as seen in Figure 9). The provided information in the handout, that all electrodes on the array are equally spaced, could help determine the angle between the electrodes. By calculating the angle (θ) of a given arc length (s) with the radius (r): $\theta = \frac{s}{r} \cdot \frac{180}{\pi}$. The arc length s (the space between electrodes) would have to be determined additionally by a function or put as a parameter in the `config` file. Also, if the next electrode on the array is not detected by the applied erosion technique (from Part II: Electrode array detection), it would be nice to have the option to skip this electrode and still be able to annotate the rest of the detected electrodes. With these two improvements already this algorithm would improve by a mile!

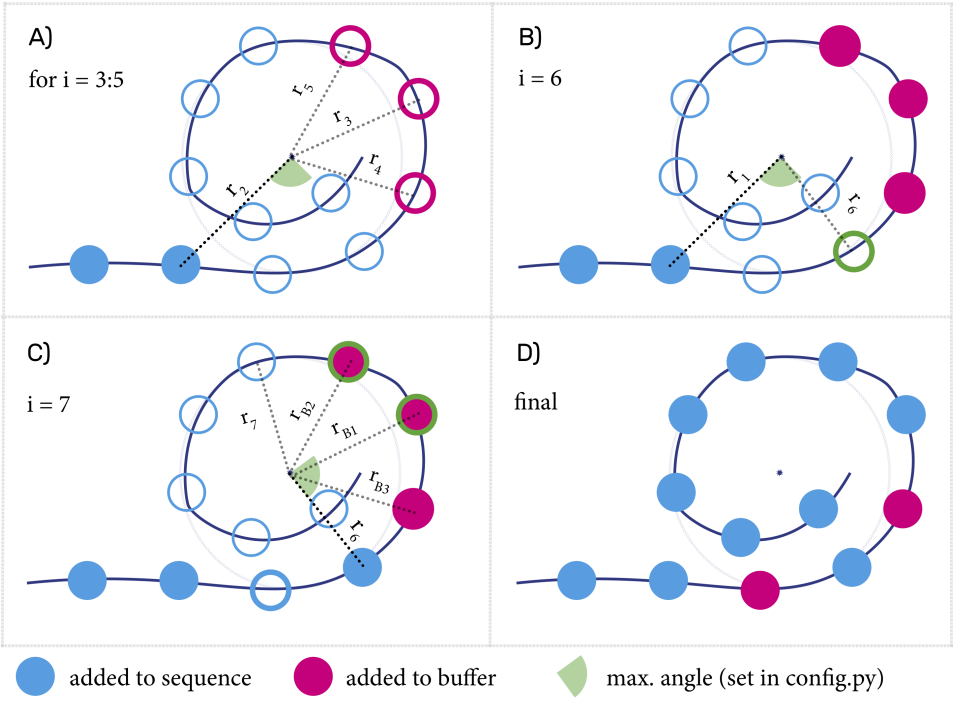

Figure 9: Missing electrodes: A) After setting electrode 1 and 2 (filled blue circles) and during the iteration steps 3 to 5, the electrodes in question (red circles) do not fit the acceptable angle (green, here 90°). B) But at iteration step 6, the electrode 4 (green outline) is next and is added to the sequence, because it lies within the maximal angle. With this, a detected electrode is skipped and not annotated. C) This can also happen during buffer checking: r_{B1} and then r_{B2} fit in the max. angle and are added sequentially to the sequence. With this, r_{B3} is skipped. The algorithm doesn't look back. D) After all iterations, the sequence with two missing electrodes is returned. The missing electrodes remain in the buffer and thus get lost.

2 Results

The results for the three different steps, center detection, electrode detection and annotation are shown below. The center appears as a dark point and the electrodes were marked with a white circle. Assigned numbers (1-12) and angles (in degree) were written next to the circles. A short comment is provided as well. Note that the black and white colors are reversed due to manipulation during the process or because of the image interpreter. This does not change the difference of intensity.

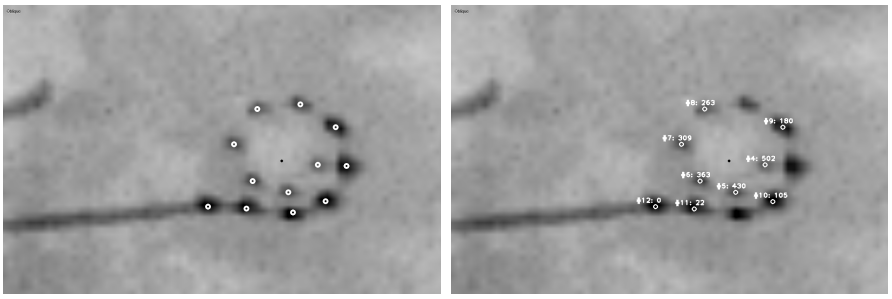

Figure 10: ID03 - **Detection: Excellent**, **Labeling: Good**.
We can observe here that three electrodes are not labeled. However, the final angular insertion depth of the deepest electrode is still relevant.

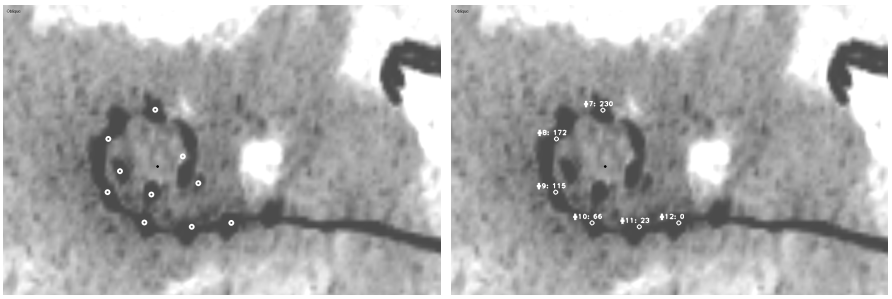

Figure 11: ID04 - **Detection: Poor**, **Labeling: Poor**.
We clearly observe that the quality of the image is much poorer than the previous picture, blobs are harder to detect. The annotation stops because of the missing electrode #6.

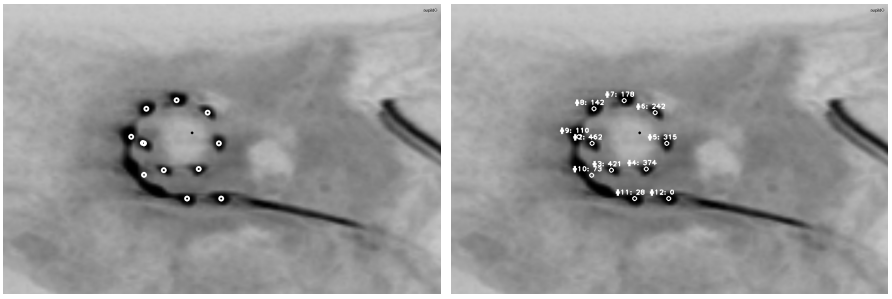

Figure 12: ID05 - **Detection: Very good**, **Labeling: Excellent**.
One electrode is between two blobs and another has been detected twice but the labeling ignores the doubled one and the measures are still very accurate as the first and last electrode are correctly labeled.

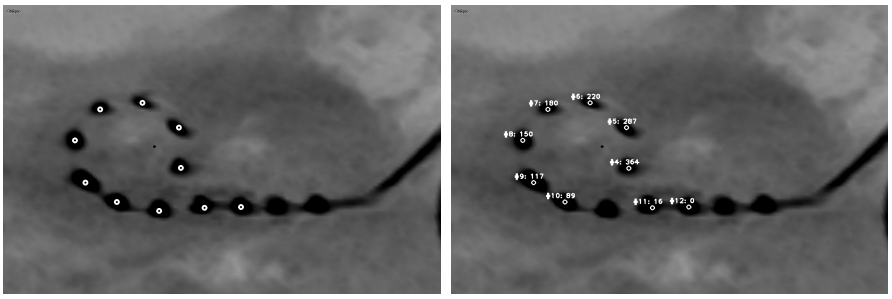

Figure 13: ID06 - **Detection: Excellent**, **Labeling: Very good**.
The two missing rightmost electrodes are due to the too small mask size set for the area of interest. However, the angle does not change much when taking into account the two last ones.

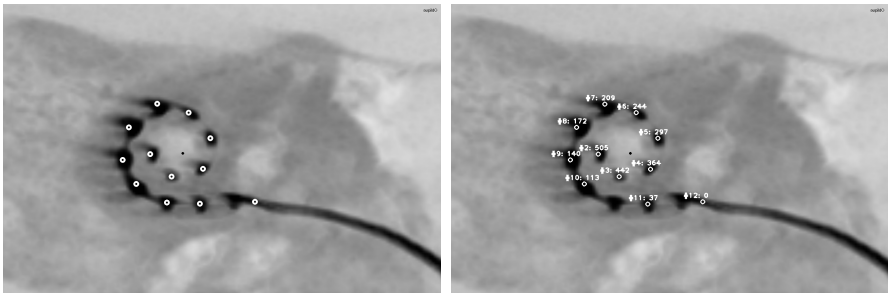

Figure 14: ID07 - **Detection: Very good**, **Labeling: Very good**.
All detected electrodes but one are correctly labeled, including the first and last ones which makes the results accurate.

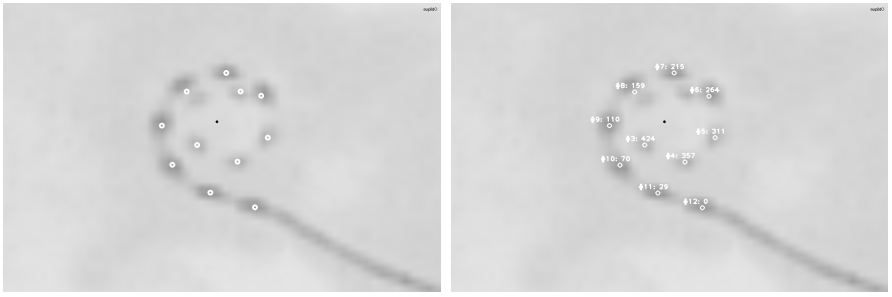

Figure 15: ID14 - **Detection: Very good**, **Labeling: Fair**.
All electrodes except one are detected correctly, unfortunately the last electrode is ignored when labeling thus making the measurement less accurate.

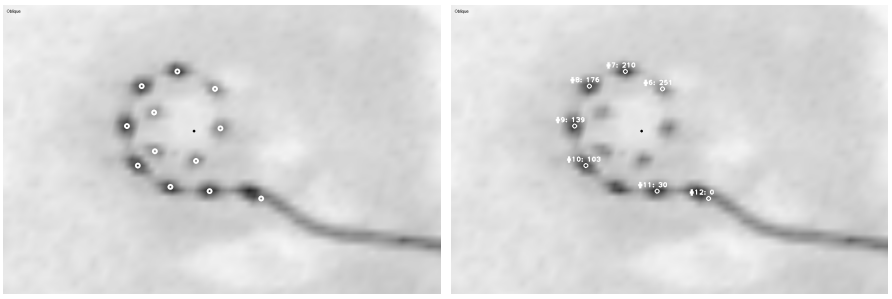

Figure 16: ID15 - **Detection: Excellent**, **Labeling: Fair**.
Innermost electrodes are neither detected nor labeled, which will give results not very accurate for the lower frequencies as in figure 15

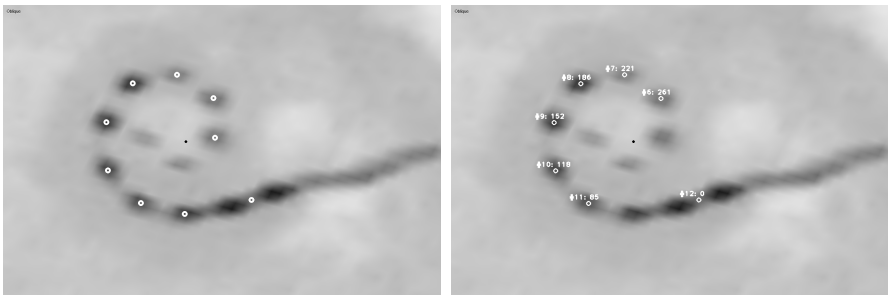

Figure 17: ID17 - **Detection: Fair**, **Labeling: Very good**.
Innermost electrodes are not detected however, the annotation misses only two of them.

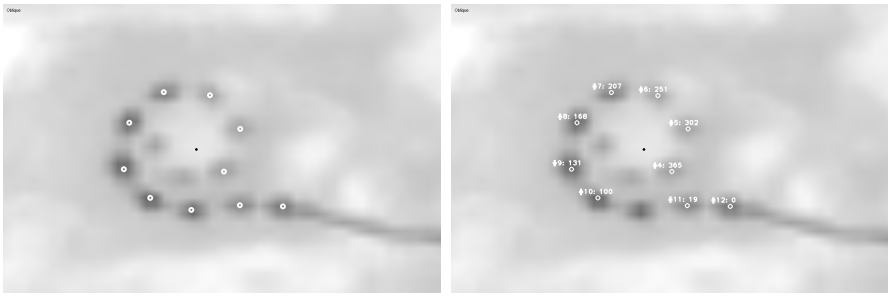

Figure 18: ID18 - **Detection: Good**, **Labeling: Very good**.
As often, the lighter electrodes are not detected but the annotation only misses one electrode.

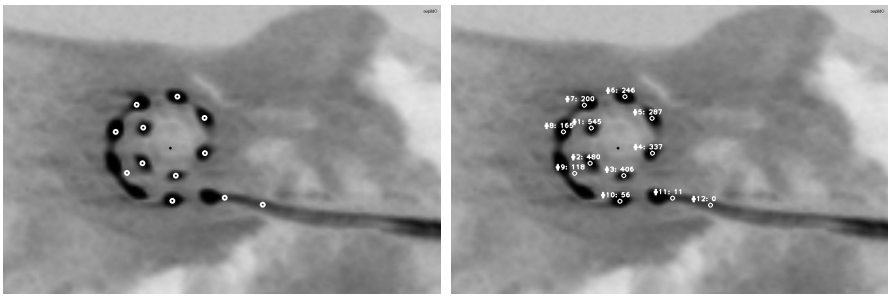

Figure 19: ID37 - **Detection: Very good**, **Labeling: Excellent**.
Despite a slight shift in the electrode array detection and a detected electrode between two electrodes, all found electrodes are annotated properly, including the deepest one.

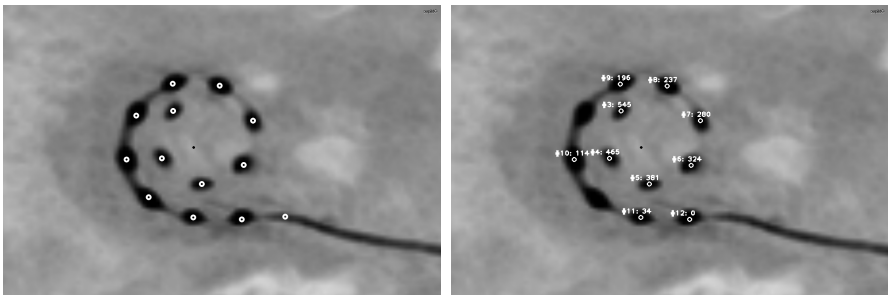

Figure 20: ID38 - **Detection: Excellent**, **Labeling: Very good**.
All electrodes plus one extra are detected and three of them including the "imaginary" one are not labeled. The outermost and innermost are present, thus the angular insertion depth is accurate.

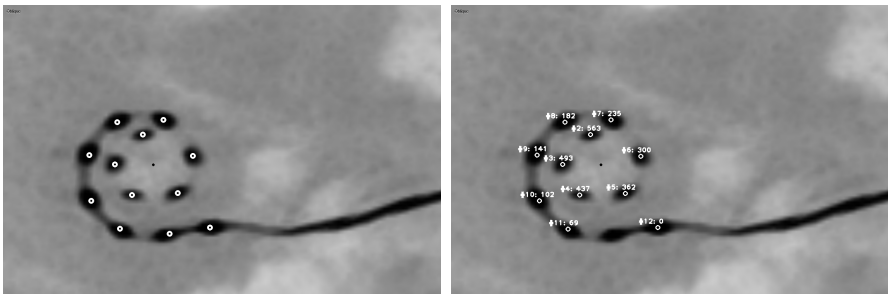

Figure 21: ID55 - **Detection: Excellent**, **Labeling: Very good**.
Only one electrode is missed inside the array. Results are accurate.

3 Discussion

This project's workflow presents three bottlenecks: the center detection, the electrode array detection and the correct annotation and sorting of each electrode. Many ideas emerged during the conception and realization processes to make the features more robust. With the provided data set, having consistent and robust normalisation and masking during the workflow, proved detrimental. While both center and electrode detection were both improved to fit (almost) all of the provided data, it would be interesting to set how they perform on a bigger data set. With the proposed improvements for the sorting algorithm above (see **Improvements**, p. 10), this algorithm could be more robust. In combination, the established algorithm here is really fast, has a high accuracy in electrode detection and can be set to be very verbose (good for further analysis).

Since the complexity of the provided images is reduced to a mere list of twelve electrode coordinates as tuples and a tuple with the coordinate to the detected cochlea center, one reduces the information to a linear space, where a simple sorting algorithm is able to efficiently compute the angular insertion depths of detected electrodes. While this allows fast scanning of a even bigger database, clinical use for this tool might be limited. Not only are the parameters for the many steps of image filtering and blob detection hard coded, but also the final output lack distinct information about the way each electrode is actually positioned in 3D space. For clinical use, it would be nice to have a GUI to get user input to check whether the automatic detected electrodes are all detected. This would then be taken into consideration before the angular insertion depth is calculated. With this, even poor quality pictures could be annotated correctly. In an even further reach for this approach, an implementation of a machine learning based algorithm, which would learn the characteristics of undetected electrodes by the said user input, could be done efficiently with a python based pipeline, in a similar structure that we provided here.

Our approach, while initially having many hurdles and lots of parameter optimisation, proved quite robust and in the end provides really adequate results, which we are quite happy with.

Contributions

Name	Contributions
Thibault Schowing	Program structure, cochlea center detection
Lionel Rohner	Electrode array detection
Tobias Blatter	Electrode annotation