Cochlear Implant Electrode Detection (CIED)

Tim Ogi, Lucien Hinderling, Dona Lerena

May 28, 2020

1 Introduction

Nowadays, profound hearing losses are most commonly treated with cochlear implants (CIs), which can restore the hearing by directly stimulating the auditory nerves with electrical impulses. A CI system consists of 1) an external audio processor, 2) the transmission coil, 3) the receiver/stimulator unit and 4) the electrode array, which is inserted into the cochlea. As the cochlea is organized tonotopically, the position of the inserted electrodes is of at most importance to estimate how a particular frequency is perceived by the patient. Therefore, the aim of this project is to determine the angular depth of the 12 inserted electrodes, using pre-and post-operational computer tomography (CT) images from multiple patients.

2 Methods

2.1 Ground truth creation

To make a more precise analysis of the training set, we identified the coordinates of the twelve electrodes by hand. For this we used the online tool *Make Sense* [**?**] .

2.2 Electrode detection and labelling

This chapter describes how the electrodes are detected and labelled from innermost to outermost.

2.2.1 Preprocessing

The images are normalized using contrast stretching. First a lower and upper bound of the input image histogram specified, we chose the 2nd and 98th percentile. Values outside of this range are ignored in the calculation to be robust against outliers. The lower and upper bound are set as the new minimum and maximum brightness, a linear transformation is applied to stretch the intensity values over the whole dynamic range of our data type. This function improves the contrast of low contrast images without distorting the relative intensities, while high contrast images remain mostly unchanged[1,2]. This step is necessary as threshold is applied on the response value of the following blob detection, which is dependent on the absolute image intensities.

[1] `http://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm`

[2] `https://scikit-image.org/docs/0.9.x/auto_examples/plot_equalize.html`

2.2.2 Blob detection

To detect bright spots in the image which could represent electrodes, blob detection Laplacian of Gaussian (LoG) blob detection is applied. It was selected over other implementations of the blob detection, as it provides the most accurate results[3] and computation speed is not a criterion. The second derivatives of Gaussian kernels with different sigma values are convoluted with the input image. For each image location the response value and the sigma value with which the highest response was achieved are stored. The results are filtered a) to only contain Gaussian responses of certain sigma values, representing the size of the electrodes, and b) to only contain responses of a certain intensity, which filters out the response to image noise or other blob-like structures with weaker intensities than the electrodes we are interested in. An implementation[4]. provided by the *scikit-image* [**?**] library was used. See figure 1 as an example of an output. The threshold was intentionally set low, so blobs can also be detected in images where the electrodes are differentiated weaker.

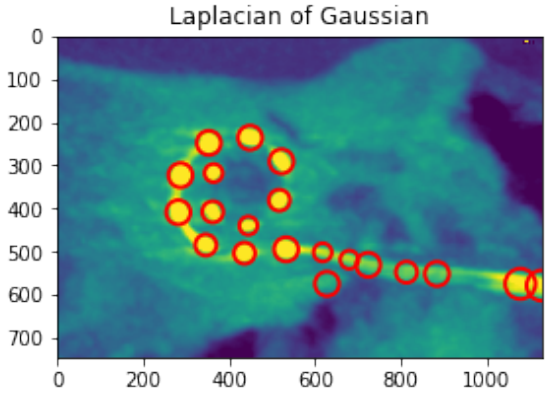


Figure 1: Image after contrast stretching. Red circles indicate locations where blobs have been detected. Radii are approximated using $r = \sqrt{2\sigma}$.

2.2.3 Blob selection and labelling with MLE

The next question is to filter the electrodes from all detected blobs and to find a path that connects them all, so they can be labelled from 1 to 12 accordingly. As the information of the pattern to be recognized is mostly structural, a graph approach comes naturally. Some key information available on the electrodes can be boiled down to these points:

Electrode shape and intensity The electrodes appear in a round shape and are brighter than the surrounding tissue. This information is already encoded in the model for blob detection.

Electrode count There are exactly 12 electrodes to be detected in every image

Equidistance All the distances between two adjacent electrodes are the same, as they are fixed onto the wire inserted in the cochlea in a regular pattern. Except the first and last electrode, all electrodes have exactly two neighbours.

[3] `https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_blob.html`
[4] `https://scikit-image.org/docs/dev/api/skimage.feature.html#skimage.feature.blob_log`

Spiral shape The wire inserted follows the form of the cochlea and has a spiral shape. The cochlea turns in a regular pattern and has no sharp angles. The electrode density is highest in the cochlea where the turns get tighter.

To remove outliers, we calculated the sum of the distances to the other blobs D_i for each blob i according to the formula below. Subsequently. we removed the blob that was furthest away from all others. This reduced the number of blobs to $n - 1$. We repeated this process until the number of blobs was 20.

$$D_i = \sum_{j \neq i}^{n} \sqrt{(b_i x - b_j x)^2 + (b_i y - b_j y)^2}$$

A graph representation is created using the *networkX* library [**?**] by assuming each detected blob to be a node. Two blobs are connected with an edge if the distance between them falls into an empirically determined range (see fig. 2).

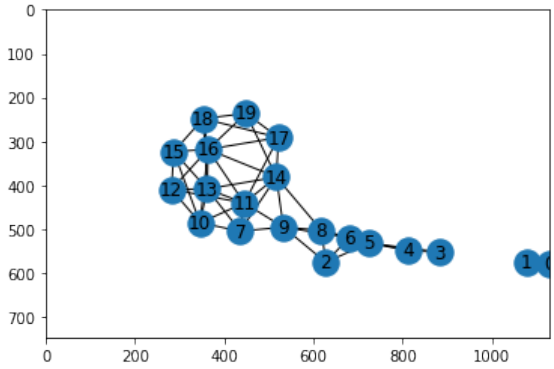


Figure 2: The initial graph constructed from the blobs in figure 1. The unconnected sub graph containing node 1 will be removed in the following step, see figure 4

Disconnected sub graphs containing less than 12 electrodes are removed. A set of valid paths is generated by using the information of the electrode count and spiral shape: Two neighbouring nodes are chosen as starting nodes (this is repeated for all n nodes and all their neighbours). The two starting nodes are used as initialization values of $path_{1...i=len(path)}$ the following recursive function:

Algorithm 1 findpath($graph, path$)

if $len(path)$ = 12 **then**
    We have found 12 electrodes. **return** $path$
end if
For the node added last to the $path$ check if they have neighbours.
for $node = [all\ neighbours\ of\ path_i, \notin path]$ **do**
    Check if we make a plausible turn by choosing this node as continuation of our path:
    α = angle between $\overrightarrow{path_{i-1}, path_i}$ and $\overrightarrow{path_i, node}$
    if $\alpha <$ empirically determined threshold **then**
        A new possible way to continue this graph has been found, the turn is not to sharp.
        $path_{new}$ = Add $node$ to $path$
        Recursively call this function with the new path:
        findpath($graph, path_{new}$)
    end if
end for
if $node = [\,]$ **then**
    There is no way to continue this path **return**
end if

This leads to a set of paths P where each path $p_i \in P$ is valid to the constraints of having exactly 12 nodes and not making impossible turns. Now, we filter the paths by assigning a cost function c_a encoding our knowledge about the equidistance of the path: The cost is the variance of the length of all the edges connecting the 12 electrodes in the path p_i

$$c_a(p_i) = var(edges \in p_i)$$

These values were then normalized with min-max normalization for each image. The path with the lowest cost was accepted as the right solution, giving us not only the 12 right nodes, but also the order in which they are connected. This already led to really good results in most of the images of the test set, except in two images where there was another path which had a lower edge length variance than the expected solution. These paths started further out, by wrongly assuming part of the wire leading to the outermost electrode as an electrode. To penalize these solutions, a second cost function c_b was introduced, that measured the "compactness". It works by summing up all distances of each node to the mean location of all 12 nodes, assigning less cost to solutions that are nicely curled up, and more cost to solutions that are stretched out.

$$c_b(p_i) = dist(nodes \in p_i\ to\ center\ of\ mass)$$

The two cost functions are combined into c_c. One can think of it as the euclidean distance of the two cost functions from the null point in figure 3.

$$c_c = \sqrt{c_a^2 + c_b^2}$$

The distribution of c_a and c_b for all valid paths detected in one image are illustrated in figure 3.

This procedure meant each path was detected twice (forwards and backwards). To calculate the angles it is necessary to start counting at the right node. The starting node was decided by comparing the distance to the electrode center of mass for the first and last node in path.

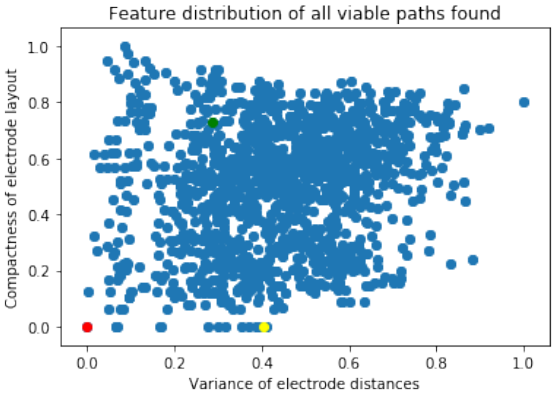
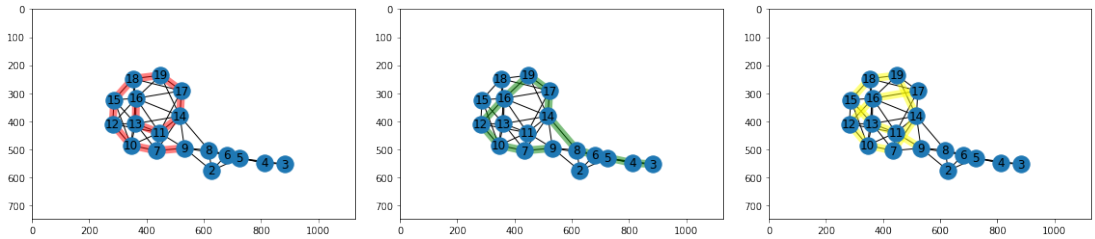


Figure 3: Costs of all paths found in an image. Three paths are highlighted to better illustrate what the dots stand for, and are pictured in the subsequent figures. **Red:** Best path found (lowest cost for c_c), **Yellow, Green:** Randomly selected.



(a) Best path found (b) Randomly selected path (c) Randomly selected path

Figure 4: Paths taken from the feature distribution illustrated in figure 3

2.3 Spectral center

The spectral center has been approximated as the center of the circle built by the three innermost electrodes.

$$\overrightarrow{M1} + s * \overrightarrow{n1} = \overrightarrow{M2} + r * \overrightarrow{n2}$$

,where $\overrightarrow{M1}$=middle point between electrode 1 and 2, $\overrightarrow{M2}$=middle point between electrode 2 and 3, s/r= some numbers, $\overrightarrow{n1}$=normal vector to the vector connecting electrode 1 and 2, $\overrightarrow{n2}$=normal vector to the vector connecting electrode 2 and 3.

3 Results

4 Electrode detection

The automatically detected electrode coordinates were compared with the coordinates from the manually annotated ground truth set. The results are illustrated in figure 5. We were able to detect the electrodes with a high accuracy.

5 Discussion

We have investigated different methods with which the electrodes in the image can be recognized. An exciting discovery was the UNET convolutional neural network, which has recently

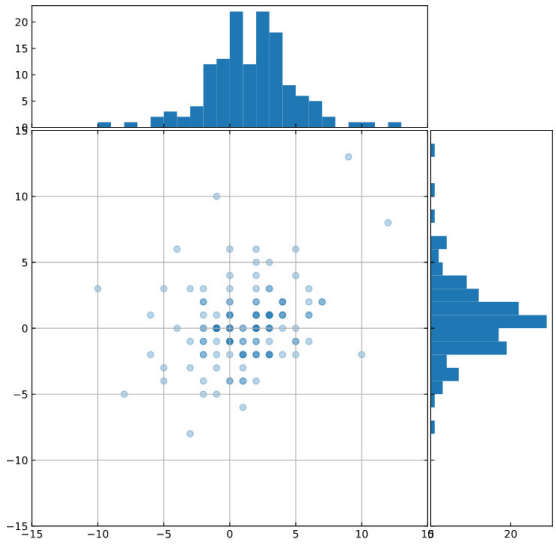


Figure 5: The x and y pixel deviations of all electrodes in the training dataset from the ground truth dataset.

gained notoriety for its reliable, accurate segmentation of medical images and its ability to achieve excellent results even with limited training data [**?**]. The concept of this architecture is not discussed in detail in this report, as it is not included in the finalized version of our code. We created a training set with postoperative images and corresponding masks of only four examples. This was a very counter-intuitive approach, since more than a hundred times the amount of this data is usually recommended for training neural networks [**? ?**]. To counteract overfitting, we used extensive data augmentation and included dropout layers in the architecture [**?**]. After training for 300 epochs, we could show very promising results.

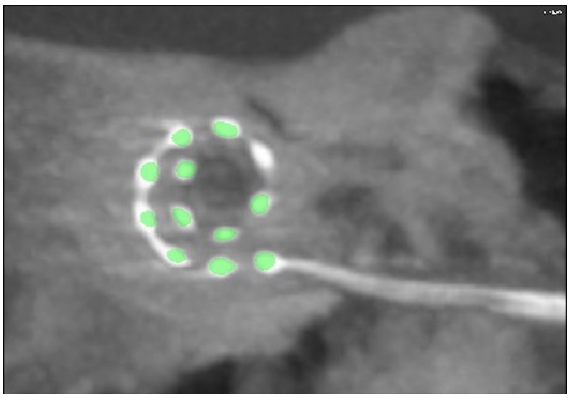


Figure 6: Electrode region prediction by UNET. The green mask indicates the region predicted to include the electrodes with a certainty of more than 95%.

In the example in figure 6, eleven of the twelve electrodes were correctly identified in an image that was not used for training the network. This result is quite astonishing considering the small amount of training data. On the one hand, the quality of the result can be attributed to the impressive properties of the UNET, however, on the other hand it also became clear that this is a rather simple problem. Deep learning approaches have had a difficult time in medicine because of their lack of traceability [**?**]. Thus, we decided to use a less complex algorithm to detect the electrodes, where every step can be followed exactly.

The use of a graph approach allows us to be less strict on the blob filtering thresholds. We considered a multitude of other cost functions (e.g. penalizing direction changes, inferring likelihood of angles between specific electrode numbers using empirically determined angle distributions, penalizing crossing edges) which we did not implement as satisfactory results were already achieved with only two features.

6 Contributions

The authors have worked very closely together in the realisation of this project. Lucien Hinderling and Tim Ogi investigated different methods for preprocessing and the recognition of the electrodes and finally agreed on the process presented here. Lucien Hinderling developed a graph based approach to realize the numbering of the electrodes. The individual components of the pipeline presented here were configured using data extracted by Tim Ogi from the images available to us and statistically evaluated by Dona Lerena. The calculations of the angles and positions of the electrodes and the centre point were carried out by Dona Lerena.

References

[] Foody, G., McCULLOCH, M. B., and Yates, W. B. (1995). The effect of training set size and composition on artificial neural network classification. *International Journal of Remote Sensing*, 16(9):1707–1723.

[] Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

[] Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. (2017). Deep Learning Scaling is Predictable, Empirically. *arXiv:1712.00409 [cs, stat]*. arXiv: 1712.00409.

[] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., and Sánchez, C. I. (2017). A Survey on Deep Learning in Medical Image Analysis. *Medical Image Analysis*, 42:60–88. arXiv: 1702.05747.

[] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

[] Skalski, P. Make sense (alpha version). `https://www.makesense.ai/`. Accessed: 20-05-2020.

[] van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors (2014). scikit-image: image processing in Python. *PeerJ*, 2:e453.

[] Wu, H. and Gu, X. (2015). Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10.