# Cochlear Implant Electrode Detection (CIED)

Tim Ogi, Lucien Hinderling, Dona Lerena

May 25, 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 atmost 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 postoperational computer tomography (CT) images from multiple patients.

## 2 Methods

### 2.1 Ground truth creation

### 2.2 Electrode detection and labeling

This chapter describes how the electrodes are detected and labeled 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 datatype. 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 treshold is applied on the response value of the following blob detection, which is dependent on the absolute image intensities.

#### 2.2.2 Blob detection

To detect bright spots in the image which could represent electrodes, blob detecion laplacian of gaussian (LoG) blob detection is applied. It was selected over other implementations of

[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`

the blob detection, as it provides the most accurate results[3] and computation speed is not a criterion. The second derrivatives 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. A implementation[4]. provided by the *scikit-image* [2] library was used. See figure 1 as an example of an output. The treshold was intentionaly set low, so blobs can also be detected in images where the electrodes are differentiated weaker.

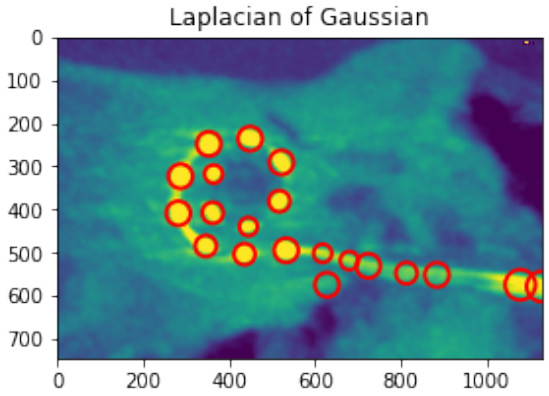



Figure 1: Image after contrast stretching. Red circles indicate locations where blobs have been detected. Radii are aproximated using $r = \sqrt{2\sigma}$.

### 2.2.3 Blob selection and labeling 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 labeled 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 apper 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 exaclty 12 electrodes to be detected in every image

**Equidistance** The all 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 exaclty two neighbors.

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

[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

Blobs are removed sequentially until only a certain amount of blobs $n$ remain. A graph representation is created using the *networkX* library [1] 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).

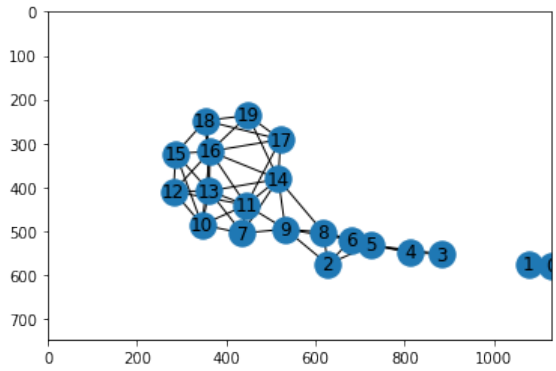



Figure 2: The initial graph constructed from the blobs in figure 1. The unconnected subgraph containing node 1 will be removed in the following step, see figure 4

Disconnected subgraphs containing less than 12 electrodes are removed. A set of valid paths is generated by using the information of the electrode cound and spiral shape: Two neighbouring nodes are chosen as starting nodes (this is repeated for all $n$ nodes and all their neighbors). 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 neighbors.
**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:
    $\alpha$ = angle between $\overrightarrow{path_i - 1, path_i}$ and $\overrightarrow{path_i, node}$
    **if** $\alpha <$ empirically determined treshold **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 exaclty 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 loest 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.

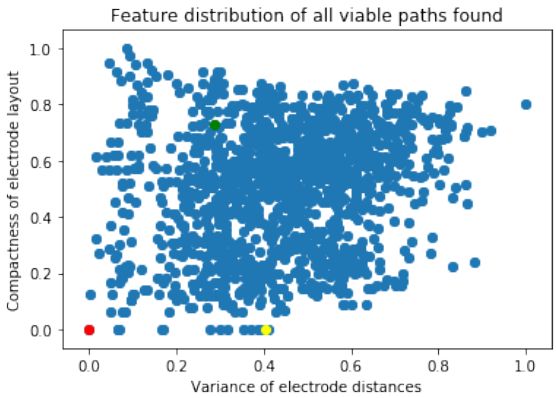
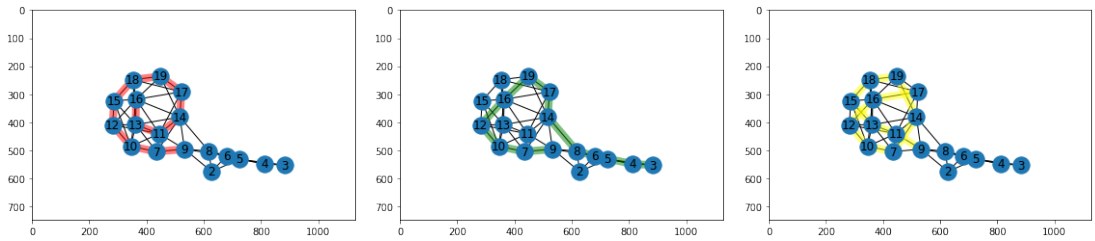



Figure 3: Costs of all paths found in an image. Three paths are highlighted to better illustrate what the dots stand for, and are pictured in the subsequent figures. **Red:** Best path found (lowest cost for $c_c$), **Yellow, Green:** Randomly selected.



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

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

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.

### 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.

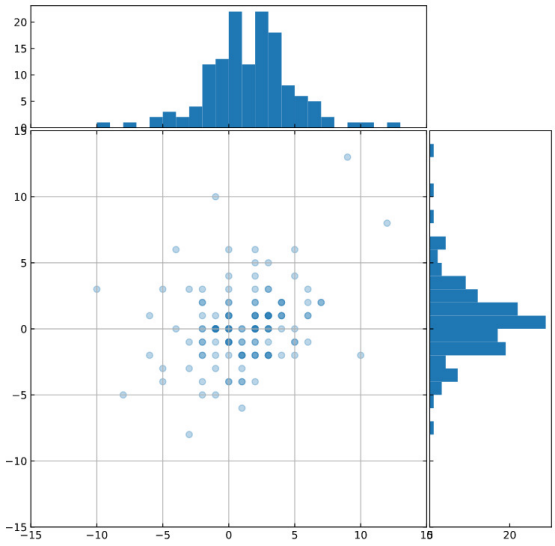



Figure 5: The x and y pixel deviations of all electrodes in the training dataset from the ground truth dataset.

## 5 Discussion

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

## 6 Contributions

## References

[1] 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.

[2] 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.