

## Root Tracer

Root tracer is a plugin of the iSEG software designed for the purpose of doing a manual segmentation to train a deep learning algorithm for (semi-)automatic image processing for the development of targeted epidural electrical stimulation paradigms through personalized computational modelling.

### Installation

After installation and compilation with CMAKE

Using the terminal, enter the designated folder chosen with CMAKE ()

And run the following command:

**For mac**

```
./bin/Debug/iSeg.app/Contents/MacOs/iSeg -D <path>/bin/Debug/
```

**Note: In addition to the necessary libraries needed to run iSEG, this plugin also requires the installation of the C++ library EIGEN.**

### How it works

The root tracer takes either a whole image or a masked subset and thresholds it using the same algorithm as iSEG's Auto -Tube Tracer plugin in order to find roots. In each algorithm step the current slice is transformed into a label map which contains all the potentially found roots. Each root is represented as a label object with a distinct label (numbered from 1 and up). The plugin finds many false positives, yet allows the user to manually remove them and to easily select the correct ones. The user may add more roots to the given results making it easier if some roots cannot be found with certain settings but only with other settings. The plugin predicts a given root centroid's position in the consecutive slice by considering:

- The distance between the given root and all roots in the previous slice
- The difference in parameters between the given root and all roots in the previous slice. Each root is characterized by parameters such as: number of pixels, number of pixels on border, diameter, perimeter, physical size, roundness and more (for more information see ITK's documentation on `itk::ShapeLabelObject`).
- A unique Kalman filter for each root that uses the position of all occurrences of the given root in the previous slices to predict its next position.

### Detailed Explanation

As an example, let's take a slice that we have already filtered all the false positives and are left with 2 distinct roots (represented by two differently colored circles *a* and *b* below). The algorithm will first find all the potential roots in the next slice and then will infer the connection between the found potential root to the confirmed roots in the previous slice (the rectangle *x* represents one of these potential roots).



Each time a potential root is found ( $x$ ) the algorithm calculates: the difference in all parameters between the potential root ( $x$ ) and all of the confirmed roots in the previous slice ( $a$  &  $b$  in this example), the distance from the potential root to all the confirmed roots in the previous slice and the difference between the potential root's centroid position and its predicted centroid position given by its Kalman filter (each time a Kalman filter is updated it predicts its next position). The algorithm takes all these results and calculates probabilities of the potential roots being each of the confirmed roots of the previous slice using a softmax function.

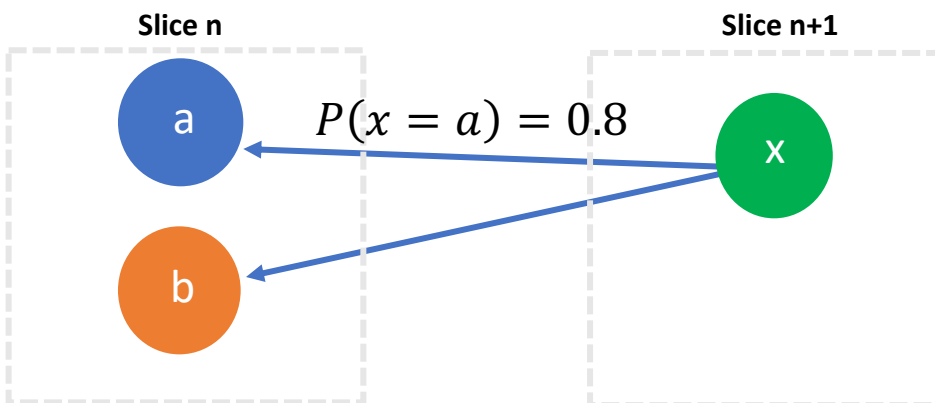
Mathematically the probability of root  $x$  from the current slice being root  $a$  from the previous slice is defined as:

$$P(x = a) = \frac{e^{-(w_d \times D_{x \rightarrow a} + w_p \times \Delta P_{x \rightarrow a} + w_k \times \Delta K_{x \rightarrow a})}}{\sum_i^N e^{-(w_d \times D_{x \rightarrow i} + w_p \times \Delta P_{x \rightarrow i} + w_k \times \Delta K_{x \rightarrow i})}}$$

Where  $w_d$ ,  $w_p$  and  $w_k$  are weights given for the distance between the potential and a given confirmed root, difference in parameters between the potential and a given confirmed root and the difference between the Kalman filter's prediction of a given confirmed root and the position of the potential root respectively and  $D_{x \rightarrow a}$ ,  $\Delta P_{x \rightarrow a}$ ,  $\Delta K_{x \rightarrow a}$  are the values respectively.

If the highest probability is larger than the threshold probability the potential root will be classified as the corresponding root from the previous slice that generated the said probability.

If we take the previous example the algorithm may result in:



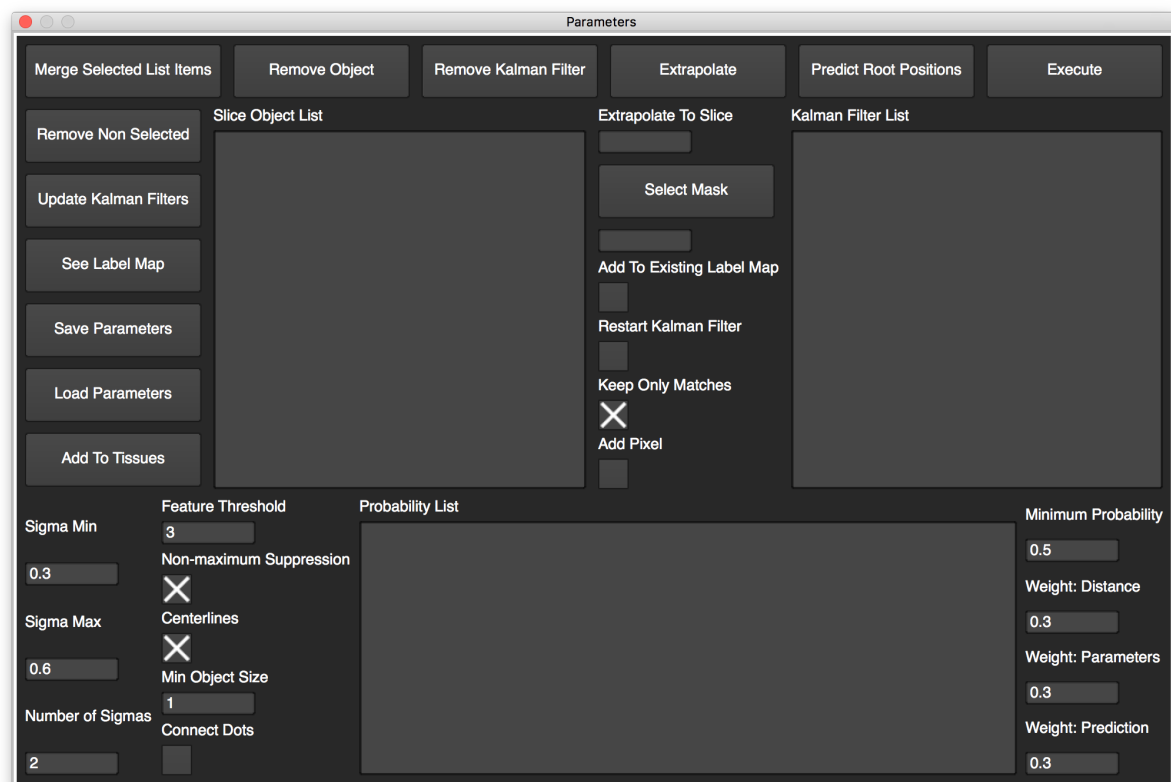
$$p(x = b) = 0.2$$

In this case algorithm will label x as *a* (assuming the threshold probability is lower than 0.8).

## User Guide

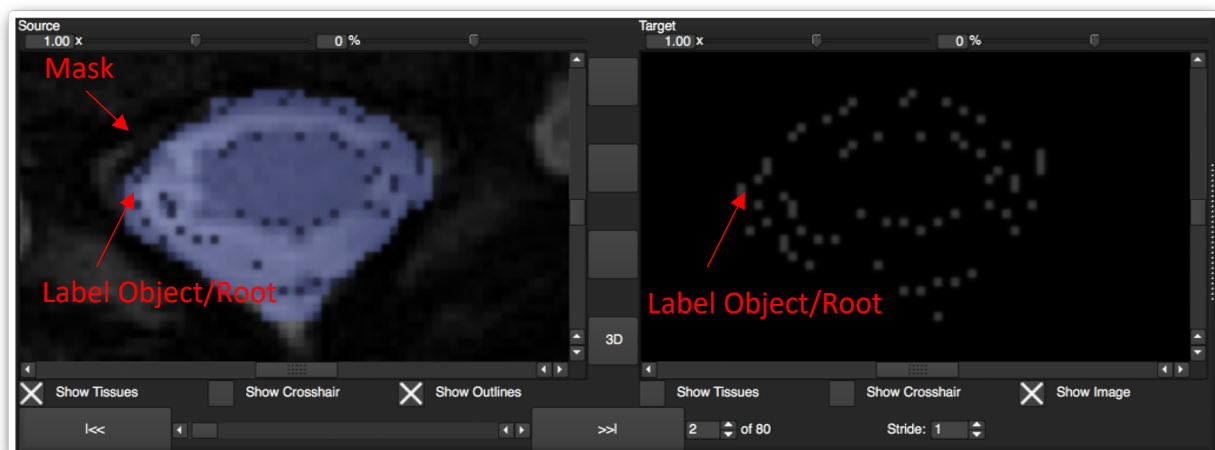
The main panel of the plugin is shown below

Main Panel



MRI Image

Label Map



## Lists

Slice Object List	Probability List	Kalman Filter List
root b root a	Probability of 1 -> root b = 0.999997 Probability of 1 -> root a = 0.000003 Probability of 2 -> root b = 0.000004 Probability of 2 -> root a = 0.999996  <a href="#">Sandbox Tool</a>	root b root a

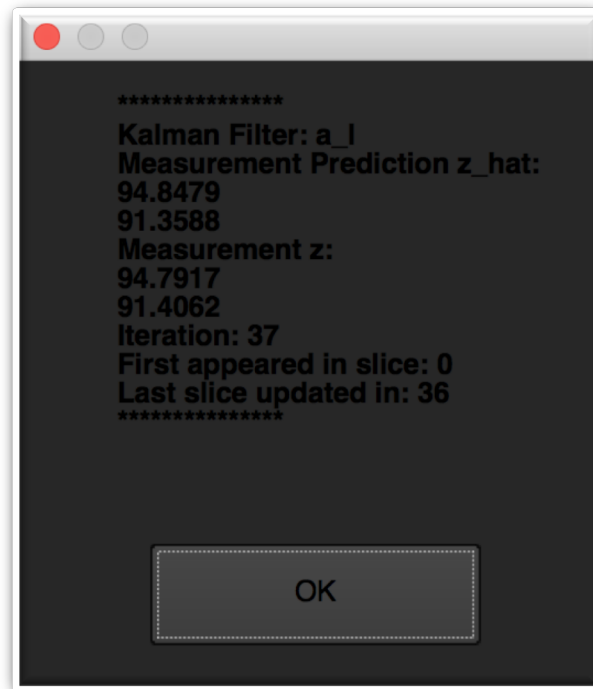
1. **Slice Object List** – names of potential roots found in the current slice's label map.

### Interactions:

- a. Double clicking on an object in the list allows to edit its name. **NOTE: Editing an objects name into a name that didn't exist before in ANY slice will create a new Kalman filter with this name** (see Update Kalman Filters button).
  - b. Clicking on an object in the list highlights it in the label map.
2. **Probability List** – probability of each potential root in the current slice to be a root found in the previous slice. Each potential root in the **Probability** List is labeled via its index in the **Slice Object List** and **NOT** via its name.
  3. **Kalman Filter List** – a list of **ALL** the Kalman filters

### Interactions:

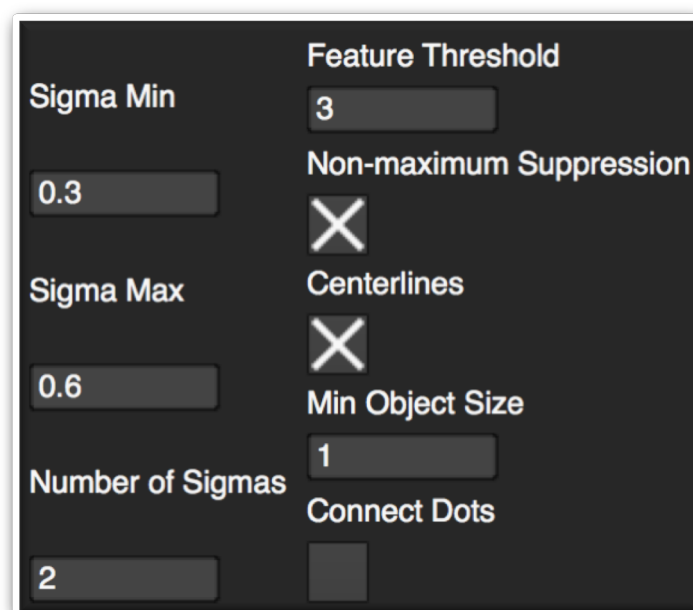
- a. Double clicking on a Kalman filter will show all the information on it (see image below for an example of a Kalman Filter named: *a\_1*)  
**Note: the coordinates are ordered as following: x on top, y below**



- b. If the **Restart Kalman Filter** checkbox is checked -> double clicking on a Kalman filter will reset it (see Update Kalman Filters button explanation to understand why it works this way).

## Buttons

1. **Execute** – Finds the potential roots in the current slice. Roots are filtered using the following editable image processing parameters:



- The roots are calculated using a multiscale hessian filter which uses a user defined number of sigmas between a user defined range.
- **Non-max suppression** and **centerline** extraction may be used as techniques for finding roots.
- **Feature threshold** – a threshold for a feature to be identified as a root
- **Min Object Size** – minimum number of pixels in a feature for it to be classified as a root.
- **Connect Dots** – if checked connected features will be merged
- **Select Mask** – select a mask to use for the region in which to conduct the image processing operation.

The image is processed using the above image processing parameters. This is followed by transforming the output image into a label map where each feature is a label object labeled from 1 up to the number of features found.

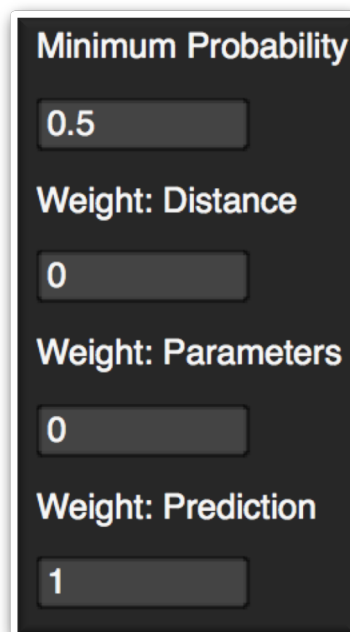
If a label map already exists for the current slice (has objects in the Slice Object List) Pressing **Execute** erases the current label objects (and items from the Slice Object List) and re-calculates from scratch unless **Add To Existing Label Map** is checked which will merge new found objects with the existing label map.

2. **Add To Tissues** – Adds all found roots to **existing** tissues. This step is **essential** for the 3D reconstruction of roots! **Before Clicking make sure that:**
  - **All slices have been worked on**
  - **All roots have a tissue created with their name**
  - **All roots appear in the Kalman filter list**
3. **Remove Kalman Filter** – Removes selected Kalman Filter (needs to be selected from the Kalman Filter List)
4. **Remove Object** – Removes selected objects from the current slice's Slice Object List and from its label map.
5. **Merge Selected List Item** – merges selected objects from the slice object list into 1 object. This also merges them in the label map.
6. **Remove Non-Selected** – The user may click on an object (root/feature) on the current slice's label map directly (highlights the object in the Slice Object List as well). Every object that was clicked in the current slice will be stored in memory and when the user clicks on the **Remove Non-Selected** button it will remove all non-selected objects from the label map and from the Slice Object List (similarly as to the **Remove Object** button). **NOTE: The memory is retained so long as the user is on the current slice, if one had made a mistake and would like to reset this memory (clicked a wrong object for example) the user just needs to move to a different slice and come back.** This method makes it very fast to filter false positives in a given slice (just click on all the correct objects and then this button to filter).

7. **Add Pixel (Check Box)** – When checked, any pixel clicked on the image will be added to the label map and become an object in the object list. (Useful for manually adding pixels to an object).
8. **Predict Root Positions** – Each root predicted position (via its Kalman filter) is added to the label map and the root is added to the object list.  
**Note:** can be used on empty label maps -> may replace the need of the **Execute** or **Extrapolate** buttons if filters have converged.
9. **See Label Map** – shows the label map of the current slice (useful as when the user highlighted a specific root and now wants to see all roots).
10. **Save Parameters** – saves parameters into a file. **IMPORTANT! When saving a project in iSEG it does not save the work of this filter! That is why the user needs to save all work done with this filter into a separate file.**
11. **Load Parameters** – loads plugin parameters from a file
12. **Extrapolate** – Automatically finds roots in next slices based on the current slice (continues until the last slice or until the slice specified via **Extrapolate To Slice**).

How does it work – **Extrapolate** calls the **Execute** button (same image processing parameters are used as would be used if the user had clicked on **Execute**) on the next slice and then calculates the probability for each found object to be the continuation of the object from the previous slice using the softmax function. Object that have a probability higher than the number specified in **Minimum Probability** are given the same name as the objects they are linked to from the previous slice. Objects that do not pass the criteria to be linked to an object from the previous slice are named via their index in the Slice Object List unless **Keep Only Matches** which only keeps the objects that have passed the criteria.

The parameters used for the softmax function are shown in the image below:



Minimum Probability

0.5

Weight: Distance

0

Weight: Parameters

0

Weight: Prediction

1

Where  $w_d$ ,  $w_p$  and  $w_k$  are Distance, Parameters and Prediction respectively.

**13. Update Kalman Filter** – Updates the measurements of **ALL** Kalman Filter and predicts their next position.

Explanation – For each Kalman Filter and for each slice up to the current slice, the Slice Object List is searched for the given Kalman filter's name. If present, the label object's centroids position is given as measurement to the Kalman Filter which it uses to predict its next position, this is called a Kalman Filter iteration. **This means that updating after each sliced worked on, is equivalent to updating after working on multiple slices.** For each Kalman Filter an iteration is performed **only** if the slice number is **greater** than the slice number of when the last iteration was performed.

Usefulness of restarting a Kalman Filter – Each Kalman Filter is initialized before the first update from random numbers which in some cases may hamper convergence. Usually after about 10 iterations the Kalman Filter has converged and manages to predict very accurately its object's next position. If after many iterations the Kalman Filter has not converged restarting it and re-updating the Kalman filter may fix the problem. If the user has made a mistake the user may fix the mistake reset and re-update the Kalman filter. This can happen for example in the case where a user labeled a root updated the Kalman filter and then regretted and deleted the root from the given slice resulting in the addition of a false measurement to the Kalman filter.

### Important Notes

1. Label Objects in the label map are clickable directly on either the original image or the label map (useful!).
2. Any work done with the plugin is not saved using iSEG's normal saving buttons and need to be saved separately. All label maps, lists and Kalman filters can only be saved and loaded via their respective buttons in the plugin, whereas masks and all work done not using the plugin should be saved using the normal iSEG's saving methods.
3. **MAC – WINDOWS** – iSEG projects saved on windows can only be loaded on mac if they were saved as **.h5** files (not **.prj** files)
4. **It is recommended to save often!**

### Typical Order of Usage

#### Part 1

1. Load Existing Project or MRI images and create masks for all the slices wanted (if working with masks)
2. Load Parameters (if project was worked on before)
3. Select Mask (if working with mask)
4. Define Image Processing Parameters
5. Define Probability calculation parameters
6. Press on **Execute**



7. Click on all the label object one desires to keep by clicking them directly on the label map.
8. Click on **Remove Non Selected**
9. Edit the names of the remaining objects by double clicking on them in the **Slice Object List**
10. Press on **Update Kalman Filter**
11. (optional) **SAVE**

## Part 2

1. (optional) Set the next slice in **Extrapolate To Slice** (working slice by slice for as it may be faster)
2. Press on **Extrapolate**
3. Click on all the label object one desires to keep by clicking them directly on the label map.
4. Click on **Remove Non Selected**
5. Edit the names of the wrongly named objects by double clicking on them in the **Slice Object List**
6. Press on **Update Kalman Filter**
7. (optional) **SAVE**
8. Repeat Part 2

**Note:** As the Kalman's filter iterations increase changing the probability calculation parameters is recommended as the default values do not represent optimal values and are **purely random!** Playing with these parameters will reduce the number of wrongly named objects (in some slices to 0) and will reduce the number of false positives found.