3DSomaMS User's Guide

Sergio Luengo-Sanchez[1], Concha Bielza[1], Ruth Benavides-Piccione[2,3], Isabel Fernaud-Espinosa[2], Javier DeFelipe[2,3,4] and Pedro Larrañaga[1]

[1]Computational Intelligence Group, Escuela Técnica Superior de Ingenieros Informáticos, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Campus Montegancedo, Boadilla del Monte, Madrid, 28660, Spain
[2]Laboratorio Cajal de Circuitos Corticales, Centro de Tecnología Biomédica, Universidad Politécnica de Madrid, Campus Montegancedo, Pozuelo de Alarcón, Madrid, 28223, Spain
[3]Instituto Cajal (CSIC), Ave. Dr. Arce, 37, Madrid, 28002, Spain
[4]Centro de Investigación Biomédica en Red sobre Enfermedades Neurodegenerativas (CIBERNED), ISCII, Madrid, 28031, Spain

Correspondence: `sluengo@fi.upm.es`

Abstract

The definition of the soma is fuzzy, as there is no clear line demarcating the soma of the labeled neurons and the origin of the dendrites and axon. Thus, the morphometric analysis of the neuronal soma is highly subjective. This software provides a mathematical definition and an automatic segmentation method to delimit the neuronal soma. We applied this method to the characterization of pyramidal cells, which are the most abundant neurons in the cerebral cortex. Thus, this software is a means of characterizing pyramidal neurons in order to objectively compare the morphometry of the somata of these neurons in different cortical areas and species.

1 Prerequirements

This software has been developed as an R package. Consequently it is needed an R environment and internet connectivity to download additional package dependencies. R software can be downloaded from `http://cran.rstudio.com/index.html`. During R installation "64-bit Files" must be ticked.

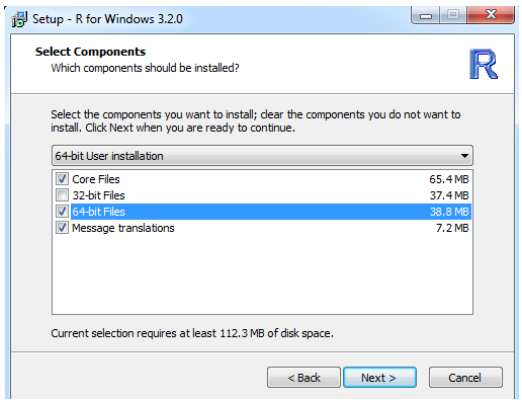


Figure 1: *R installation*: 64-bit option must be ticked.

To process the meshes our package uses MeshLab. Since the latest version of MeshLab is only available for Microsoft Windows 64-bits and we found some problems running automatic scripts in previous versions of MeshLab, a Microsoft Windows 64-bits OS is needed to use our package. Latest version of MeshLab software is available at `http://sourceforge.net/projects/meshlab/files/latest/download?source=files`.

2 3DSomaMS installation

Once R and MeshLab are installed, next step is the installation and configuration of the package. First of all, MeshLab path must be included in the environment variable PATH. The steps to follow are detailed below. An image of each step is shown in AppendixA:

1. Click *Start* symbol of Windows → Right click on *Computer* → Select *Properties*.
2. Select *Advanced system settings* at the top left corner.
3. Select the *Advanced* tab and click *Environment Variables...*
4. Search *Path* in *System variables* box → Select *Path* → Click *Edit...*
5. Go to the end of the *Variable value* text field → Do not close the window.
6. Click *Start* symbol of Windows → Search *meshlabserver* → Right click on *meshlabserver* → Select *Properties*.
7. Select *Shortcut* tab → Copy the text in *Start in* text field without the quotes.
8. Go to the window that you left opened → Add a *semicolon* at the end of the *Variable value* text field → Paste the text you copied previously → Select OK in all windows to close them.

Now, to install the package it should be copied the next sentences in the R console:

```
setRepositories(addURLs = cigrepo = "http://vps136.cesvima.upm.es/R"))
install.packages("SomaMS")
```

During installation it is possible that some warning messages appear. It is due to R has not got writing permission. Clicking OK you create a folder in *My Documents* to save the package dependencies. Then, when R asks you to install from sources you must answer Yes. One the package is installed and uncompressed its size will be more than 300MB due to the meshes included as examples. An example is showed in Fig.2. Finally you load the package with the command:

```
library("SomaMS")
```

After that, you have loaded 3DSomaMS in the R workspace so you can start to repair and segment somas. The somas that we show in some images of the article are included as mesh examples. In the next section we show how to execute these examples and how to repair and segment new somas. It should be noted that the original generated triangular meshes necessary for the development of the present method, were created to obtain a single coarse solid surface of a particular threshold, which included both the soma and proximal dendrites of labeled neurons. Thus, they are available only for reproducibility purposes of the present algorithm.

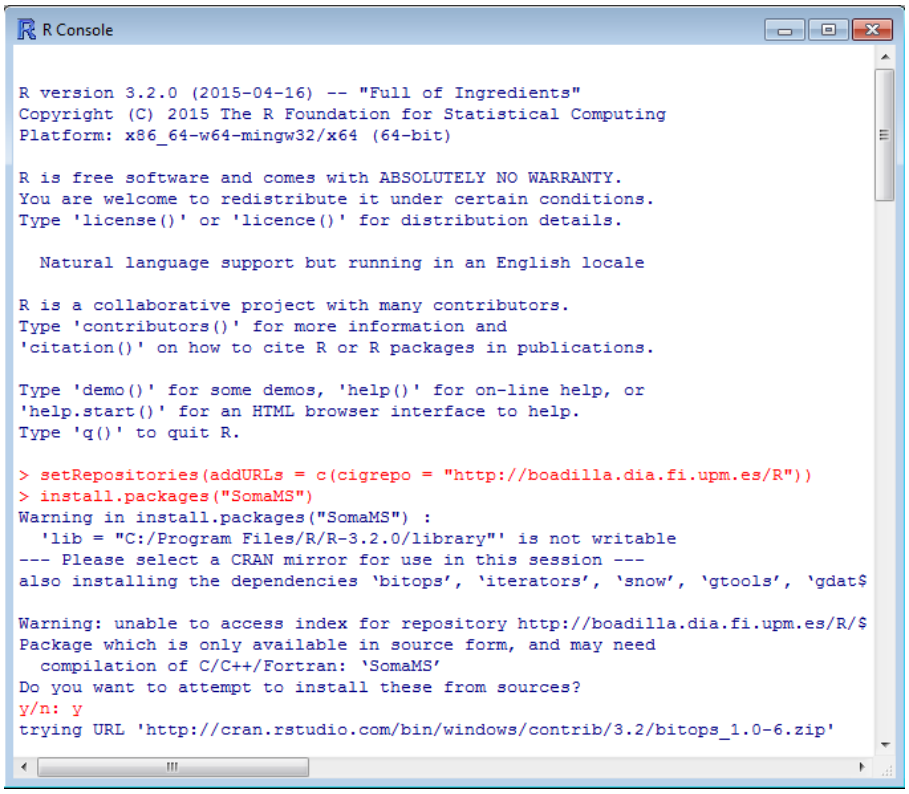
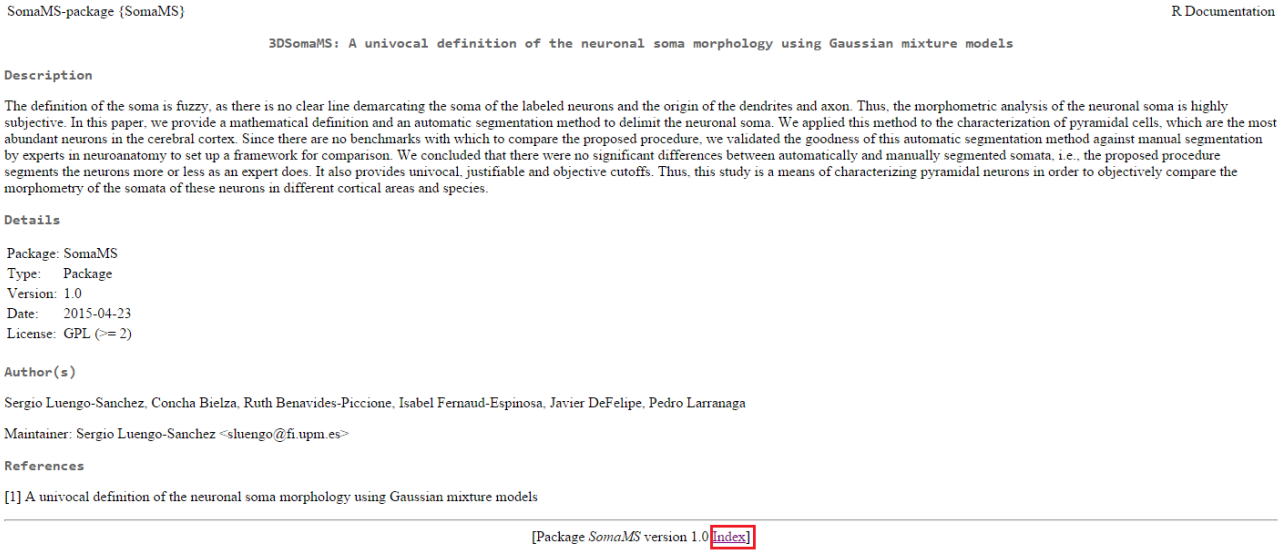


Figure 2: *3DSomaMS command installation*


SomaMS-package {SomaMS} R Documentation

3DSomaMS: A univocal definition of the neuronal soma morphology using Gaussian mixture models

Description

The definition of the soma is fuzzy, as there is no clear line demarcating the soma of the labeled neurons and the origin of the dendrites and axon. Thus, the morphometric analysis of the neuronal soma is highly subjective. In this paper, we provide a mathematical definition and an automatic segmentation method to delimit the neuronal soma. We applied this method to the characterization of pyramidal cells, which are the most abundant neurons in the cerebral cortex. Since there are no benchmarks with which to compare the proposed procedure, we validated the goodness of this automatic segmentation method against manual segmentation by experts in neuroanatomy to set up a framework for comparison. We concluded that there were no significant differences between automatically and manually segmented somata, i.e., the proposed procedure segments the neurons more or less as an expert does. It also provides univocal, justifiable and objective cutoffs. Thus, this study is a means of characterizing pyramidal neurons in order to objectively compare the morphometry of the somata of these neurons in different cortical areas and species.

Details

Package: SomaMS
Type: Package
Version: 1.0
Date: 2015-04-23
License: GPL (>= 2)

Author(s)

Sergio Luengo-Sanchez, Concha Bielza, Ruth Benavides-Piccione, Isabel Fernaud-Espinosa, Javier DeFelipe, Pedro Larranaga

Maintainer: Sergio Luengo-Sanchez <sluengo@fi.upm.es>

References

[1] A univocal definition of the neuronal soma morphology using Gaussian mixture models

[Package *SomaMS* version 1.0 Index]

Figure 3: *Help page of the package*

3 Using 3DSomaMS

Lets start writing ?SomaMS in the R console. The page that appears in front of you is the help page (see Fig.3). Some general information about the package is shown. Next, if you click the hyperlink *Index* at the end of the page you will be redirected to the index page where you can see all the functions of the software and a short description of each one of them (see Fig.4).

Depending on the characteristics of your meshes the flow of the execution could change. In the

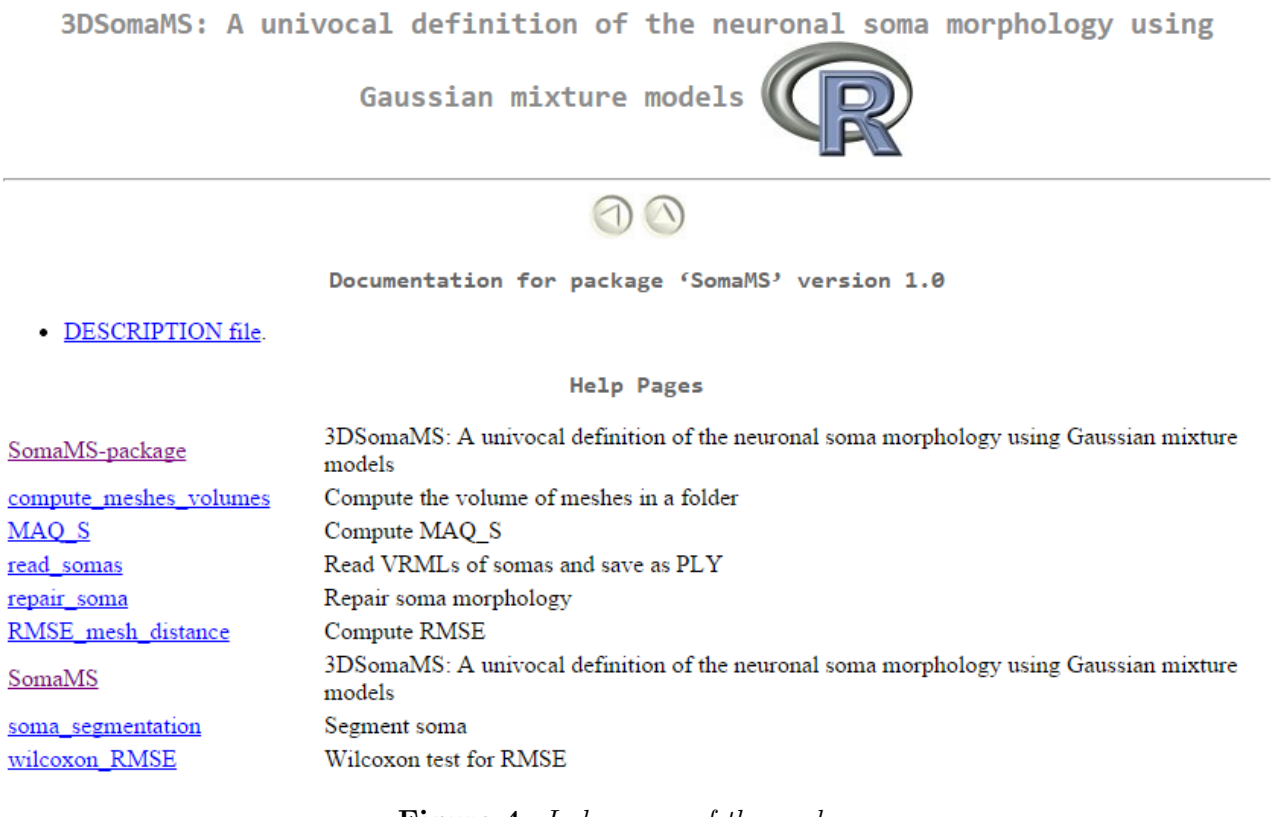


Figure 4: *Index page of the package*

examples that we provide, we first convert the meshes from VRML format to PLY format. After that, somas are repaired and finally segmented. However, when your meshes are initially in PLY format or they do not need the repairing process you can avoid some steps. Next, the main functions are explained.

3.1 convert_somas_to_PLY

Some scientific software like Imaris allow visualization of 3D and 4D microscopy data. Usually, they export that information in meshes which are in VRML format. However, to work with MeshLab is preferable meshes in PLY format. This is due to PLY format manages more information than only vertices and the faces as, for example, colors, normals, quality, etc. To convert meshes from VRML format to PLY format you must run the **convert_somas_to_PLY** function.

Clicking hyperlink **convert_somas_to_PLY** in the index page or writing

```
?convert_somas_to_PLY
```

in R console opens the help page of the function.

This page shows the following example that can be executed if it is directly copied to the R console:

```
convert_somas_to_PLY(read_directory = system.file("test/VRMLs", package = "SomaMS"),
write_directory = file.path(tempdir(), "PLYs"), parallel = TRUE)
```

The example reads the VRMLs saved in a folder and creates a temporal directory to save the meshes in PLY format. You can change the path of *read_directory* to read your own VRMLs. Also, you can change *write_directory* to save the PLY files in other folder. The output meshes of some functions are the input of other functions so if you change the path of *write_directory* you also have to change the input path of the next function. *write_directory* usually will be the first input of the **repair_somas** or **segment_somas** functions.

3.2 repair_somas

The 3D generated somata sometimes showed distortions due to the hole produced by the micropipette used to inject the dye. Thus, the labeled cell bodies are not suitable for an automated morphological analysis because the measurements on a damaged surface are incorrect. Therefore, once the meshes are in PLY format, both if you have applied the previous step to convert your somas from VRML or your somas are initially in PLY format, the injected somas should be repaired.

To repair your somas you should run the **repair_somas** function. It also have the following example that you can execute copying and pasting it to the R console:

```
repair_somas(directory = file.path(tempdir(), "PLYs"), output_ambient_occlusion =
file.path(tempdir(), "ambient_occlusion"), broken_mesh = file.path(tempdir(),
"broken_mesh_ao"), output_poisson_reconstruction = file.path(tempdir(),
"poisson_reconstruction_ao"))
```

This function has four inputs. The first one is the folder where damaged somas in PLY format are saved. Hence, when you have executed previously *convert_somas_to_PLY*, the writing directory path of *convert_somas_to_PLY* must match with *directory*. *output_ambient_occlusion* and *broken_mesh* are the paths where meshes will be saved after the ambient occlusion and Gaussian mixture model computation. It is recommended not to change these paths unless you want to see the three-dimensional output of each step. In that case, you can open a these meshes with MeshLab user's interface (see Appendix B). *output_poisson_reconstruction* is the path where repaired somas will be saved. Usually they will be the first input of the *segment_somas* function.

3.3 segment_somas

Once the surface of the somas is appropriate, next step is the definition of soma. To achieve it you must run the **segment_somas** function. As in **repair_somas** case, we provide an example:

```
segment_somas(directory = file.path(tempdir(), "poisson_reconstruction_ao"),
output_shape_diameter = file.path(tempdir(), "sdf"), broken_mesh = file.path(tempdir(),
"broken_mesh_sdf"), output_poisson_reconstruction = file.path(tempdir(),
"poisson_reconstruction_sdf"), final_result = file.path(tempdir(), "final_result"))
```

The first input, *directory*, have to be a path to the folder where meshes in PLY format are saved so:

- They are the output of the **repair_somas** function.
- They are the output of the **convert_somas_to_PLY** function if the somas were in VRML format and are not damaged by the injection.
- They are the path to a folder where a set of somas that are in PLY format and were not damaged by the injection are saved.

The *output_shape_diameter*, *broken_mesh* and *output_poisson_reconstruction* inputs are the paths where meshes will be saved after the shape diameter function, Gaussian mixture model and poisson reconstruction computation. It is recommended not to change these paths unless you want to see the three-dimensional output of each step. Last input, *final_result* is the final result of all process, that is, the definition of soma. The validation process is applied to these somas.

3.4 Validation

In the paper the somas are validated in two different ways. We compute $RMSE$ between the surface of the meshes and MAQ_S between the volume of the meshes.

3.4.1 *RMSE*

You can carried out the first validation method using *RMSE_mesh_distance* function. The interexpert and intraexpert validation are included in the examples of this function.

Interexpert

```
path_somas_algorithm <- file.path(tempdir(), "final_result")
path_somas_experts <- system.file("test/pre_repaired", package = "SomaMS")
experts_paths <- list.dirs(path_somas_experts, recursive = F)
pre_repaired_RMSE <- RMSE_mesh_distance(path_somas_algorithm, experts_paths[1],
   experts_paths[2], TRUE)

path_somas_algorithm <- file.path(tempdir(), "final_result")
path_somas_experts <- system.file("test/post_repaired", package = "SomaMS")
experts_paths <- list.dirs(path_somas_experts, recursive = F)
post_repaired_RMSE <- RMSE_mesh_distance(path_somas_algorithm, experts_paths[1],
   experts_paths[2], TRUE)

X11(width = 18, height = 10.37)
par(mfrow = c(1, 2))
values_barplot <- t(pre_repaired_RMSE)
colors <- c(rainbow(3))
mp <- barplot2(values_barplot, main = "RMSE before repairing experts' somas",
ylab = "RMSE", beside = TRUE, col = colors, ylim = c(0, 1.7), cex.names = 1.5,
   cex.lab = 1.5, cex.axis = 1.5)
legend("top", legend = c("Procedure  Vs  Expert 1", "Procedure  Vs  Expert 2",
   "Expert 1     Vs  Expert 2"), fill = colors, box.col = "transparent",
   x.intersp = 0.8, cex = 1.5)
mtext(1, at = mp[2,], text = c("Neuron 1", "Neuron 2", "Neuron 3", "Neuron 4",
   "Neuron 5", "Neuron 6", "Neuron 7", "Neuron 8", "Neuron 9"), line = 0.5, cex = 1)
legend("topleft", legend = "", title = "A", box.col = "transparent", cex = 2)

values_barplot <- t(post_repaired_RMSE)
colors <- c(rainbow(3))
mp <- barplot2(values_barplot, main = "RMSE after repairing experts' somas",
   ylab = "RMSE", beside = TRUE, col = colors, ylim = c(0, 1.7), cex.names = 1.5,
   cex.lab = 1.5, cex.axis = 1.5)
legend("top", legend = c("Procedure  Vs  Expert 1", "Procedure  Vs  Expert 2",
   "Expert 1     Vs  Expert 2"), fill = colors, box.col = "transparent",
   x.intersp = 0.8, cex = 1.5)
mtext(1, at = mp[2,], text = c("Neuron 1", "Neuron 2", "Neuron 3", "Neuron 4",
   "Neuron 5", "Neuron 6", "Neuron 7", "Neuron 8", "Neuron 9"), line = 0.5, cex = 1)
legend("topleft", legend = "", title = "B", box.col = "transparent", cex = 2)

print(paste0("Mean interexpert RMSE: ", mean(post_repaired_RMSE[,3])))
```

Intraexpert

```
path_somas_experts <- system.file("test/intraexpert", package = "SomaMS")
experts_paths <- list.dirs(path_somas_experts, recursive = F)
```

```
expert_1_days <- list.dirs(experts_paths[1], recursive = F)
expert_2_days <- list.dirs(experts_paths[2], recursive = F)

intraexpert_expert_1 <- RMSE_mesh_distance(expert_1_days[1], expert_1_days[2],
expert_1_days[3], TRUE)
intraexpert_expert_2 <- RMSE_mesh_distance(expert_2_days[1], expert_2_days[2],
expert_2_days[3], TRUE)

X11(width = 18, height = 10.37)
par(mfrow = c(1, 2))

valuesBarplot <- t(intraexpert_expert_1)
colors <- c(rainbow(3))
mp <- barplot2(valuesBarplot, main = "Intra-expert variability of the first expert ",
   ylab = "RMSE", beside = TRUE, col = colors, ylim = c(0,1.6), cex.names = 1.5,
   cex.lab = 1.5, cex.axis = 1.5)
legend("top", legend = c("Day 1 Vs Day 2","Day 1 Vs Day 3","Day 2 Vs Day 3"),
   fill = colors, box.col = "transparent", x.intersp = 0.8, cex = 1.5)

mtext(1, at = mp[2,], text = c("Neuron 1", "Neuron 2", "Neuron 3", "Neuron 4",
   "Neuron 5","Neuron 6"),line = 0.5, cex = 1.3)
legend("topleft", legend = "", title = "A", box.col = "transparent", cex = 2)


values_barplot <- t(intraexpert_expert_2)
colors <- c(rainbow(3))
mp <- barplot2(values_barplot, main = "Intra-expert variability of the second expert ",
   ylab = "RMSE", beside = TRUE, col = colors, ylim = c(0, 1.6), cex.names = 1.5,
   cex.lab = 1.5, cex.axis = 1.5)
legend("top",legend = c("Day 1 Vs Day 2","Day 1 Vs Day 3","Day 2 Vs Day 3"),
   fill = colors, box.col = "transparent", x.intersp = 0.8, cex = 1.5)

mtext(1, at = mp[2,], text = c("Neuron 1", "Neuron 2", "Neuron 3", "Neuron 4",
   "Neuron 5", "Neuron 6"), line = 0.5, cex = 1.3)
legend("topleft", legend = "", title = "B", box.col = "transparent", cex = 2)

print(paste0("Mean RMSE for the first expert: ", mean(c(intraexpert_expert_1))))
print(paste0("Mean RMSE for the second expert: ", mean(c(intraexpert_expert_2))))
```

Wilcoxon rank test

In addition, the significance of the output of *RMSE_mesh_distance* can be tested with *wilcoxon_RMSE* which also have an example.

```
path_somas_algorithm <- file.path(tempdir(), "final_result")
path_somas_experts <- system.file("test/pre_repaired",package="SomaMS")
experts_paths <- list.dirs(path_somas_experts,recursive=F)
pre_repaired_RMSE <- RMSE_mesh_distance(path_somas_algorithm, experts_paths[1],
   experts_paths[2], TRUE)

path_somas_algorithm <- file.path(tempdir(), "final_result")
```

```
path_somas_experts <- system.file("test/post_repaired", package = "SomaMS")
experts_paths <- list.dirs(path_somas_experts,recursive=F)
post_repaired_RMSE <- RMSE_mesh_distance(path_somas_algorithm, experts_paths[1],
   experts_paths[2], TRUE)

pvalue_before_repairing <- wilcoxon_RMSE(pre_repaired_RMSE)
print(paste0("p-value between algorithm and first expert: ", pvalue_before_repairing[1]))
print(paste0("p-value between algorithm and second expert: ", pvalue_before_repairing[2]))

pvalue_after_repairing <- wilcoxon_RMSE(post_repaired_RMSE)
print(paste0("p-value between algorithm and first expert: ", pvalue_after_repairing[1]))
print(paste0("p-value between algorithm and second expert: ", pvalue_after_repairing[2]))
```

3.4.2 MAQ_S

We also provide an example of the second validation method, the MAQ_S function. Besides, we include the a function that computes the volume of all the meshes in PLY format saved in a folder, that is, the *compute_meshes_volumes* function.

An example of how to compute the volumes of a set of meshes that are placed in a folder is:

```
compute_meshes_volumes(file.path(tempdir(), "final_result"))
```

We applied **compute_meshes_volumes** in the MAQ_S computation. An example of how to calculate the MAQ_S measure between two sets of meshes is:

```
path_somas_algorithm <- file.path(tempdir(), "final_result")
path_somas_experts <- system.file("test/post_repaired", package = "SomaMS")
experts_paths <- list.dirs(path_somas_experts, recursive = F)
path_somas_expert_1 <- experts_paths[1]
path_somas_expert_2 <- experts_paths[2]
MAQ_S_result <- MAQ_S(path_somas_algorithm,path_somas_expert_1, path_somas_expert_2)
print(paste("MAQ_S_12 is:", MAQ_S_result[1] * 100, "%"))
print(paste("MAQ_S_13 is:", MAQ_S_result[2] * 100, "%"))
print(paste("MAQ_S_23 is:", MAQ_S_result[3] * 100, "%"))
print(paste("Mean MAQ_S between algorithm and experts is:", mean(c(MAQ_S_result[1],
   MAQ_S_result[2])) * 100, "%"))
print(paste("Difference between experts MAQ_S and mean MAQ_S of algorithm is:",
   abs(MAQ_S_result[3] - mean(c(MAQ_S_result[1], MAQ_S_result[2]))) * 100, "%"))
```

A Configure MeshLab Path

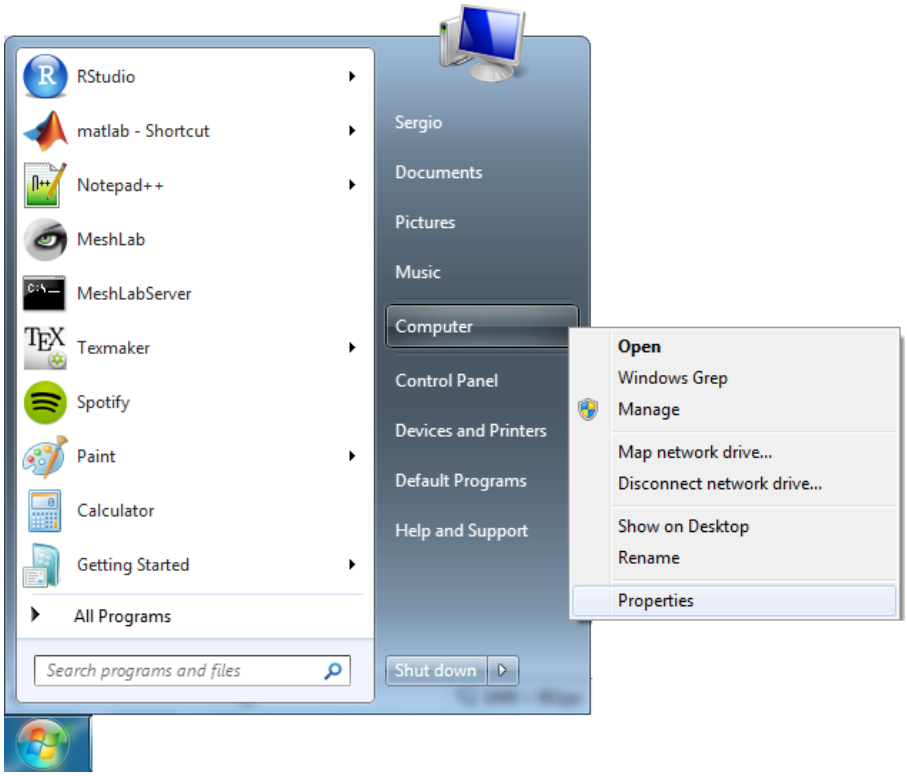


Figure 5: Step 1

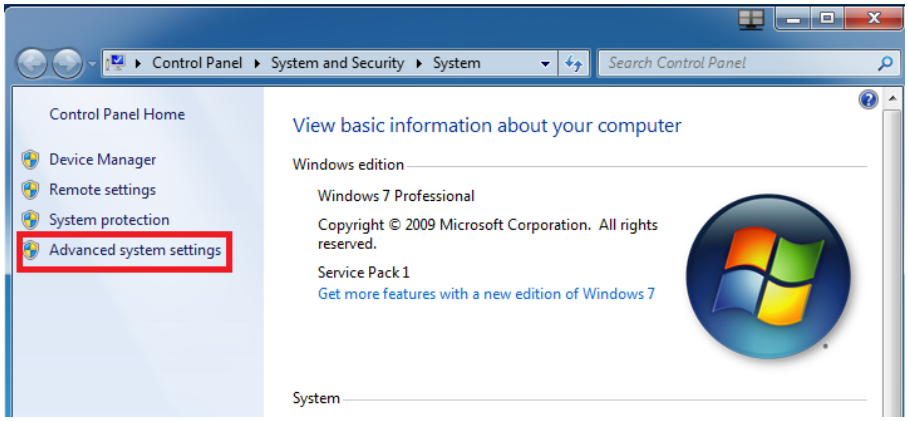


Figure 6: Step 2

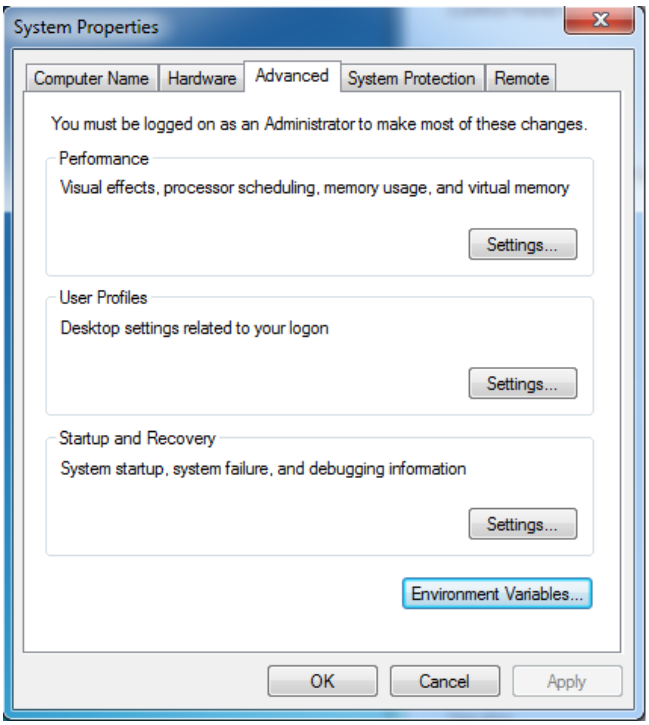


Figure 7: Step 3

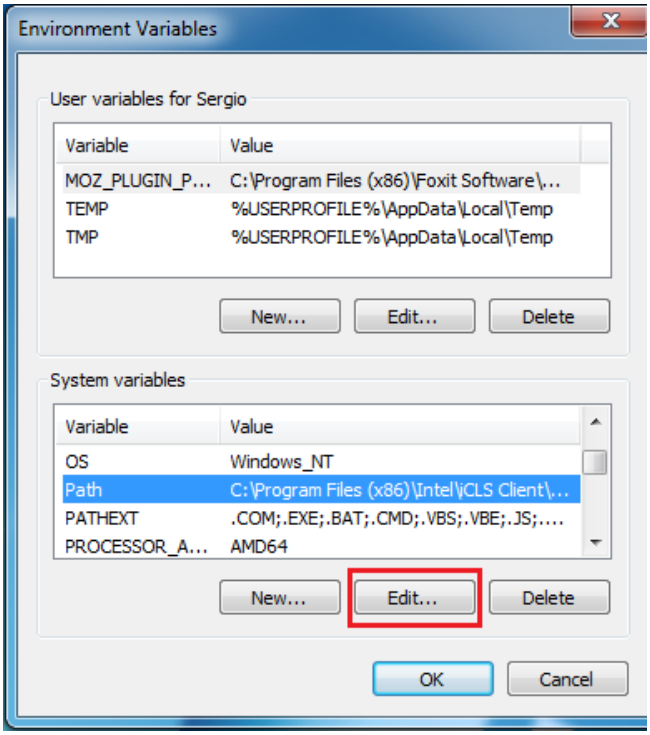


Figure 8: Step 4

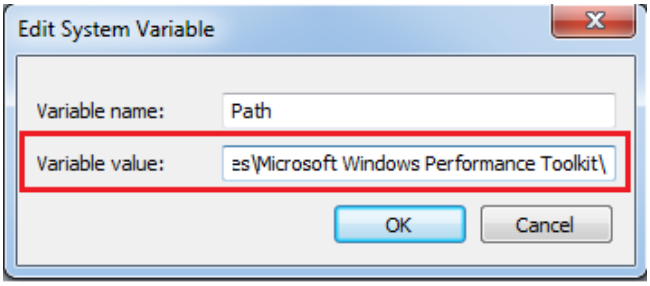


Figure 9: Step 5

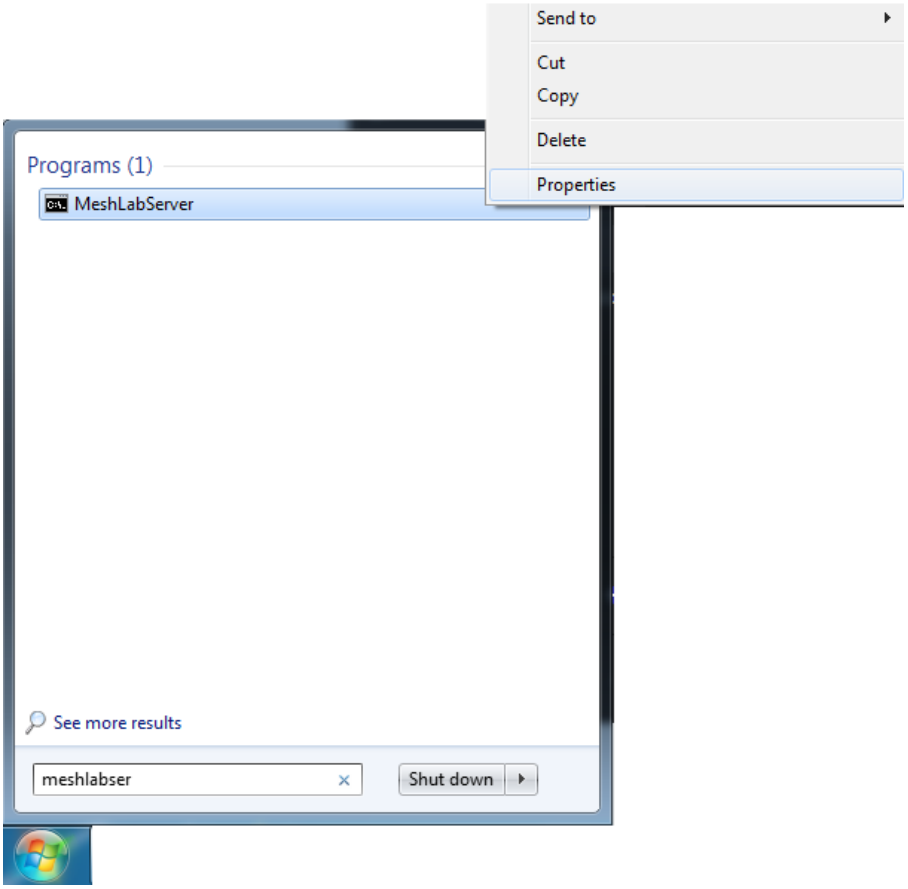


Figure 10: Step 6

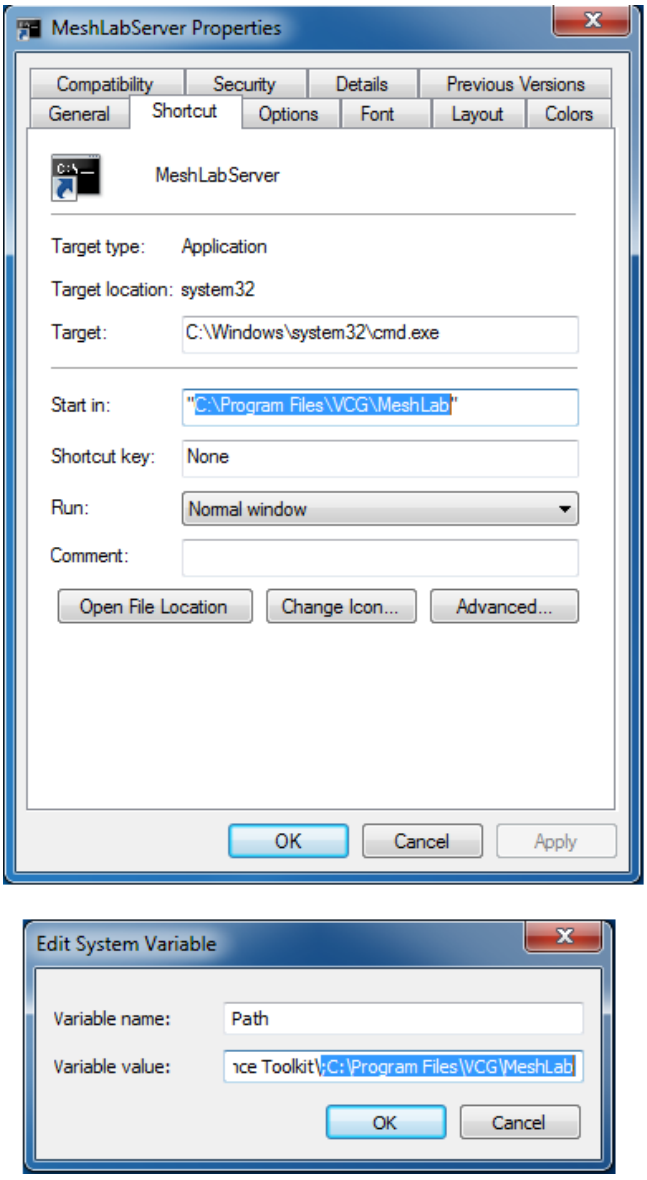


Figure 11: Step 7 and 8

B See output in Meshlab

First you have to open MeshLab application so you see the initial window of MeshLab (see Fig. 12). Then click the Import mesh icon that we denote in red in the previous image and search the mesh you want to analyzed.

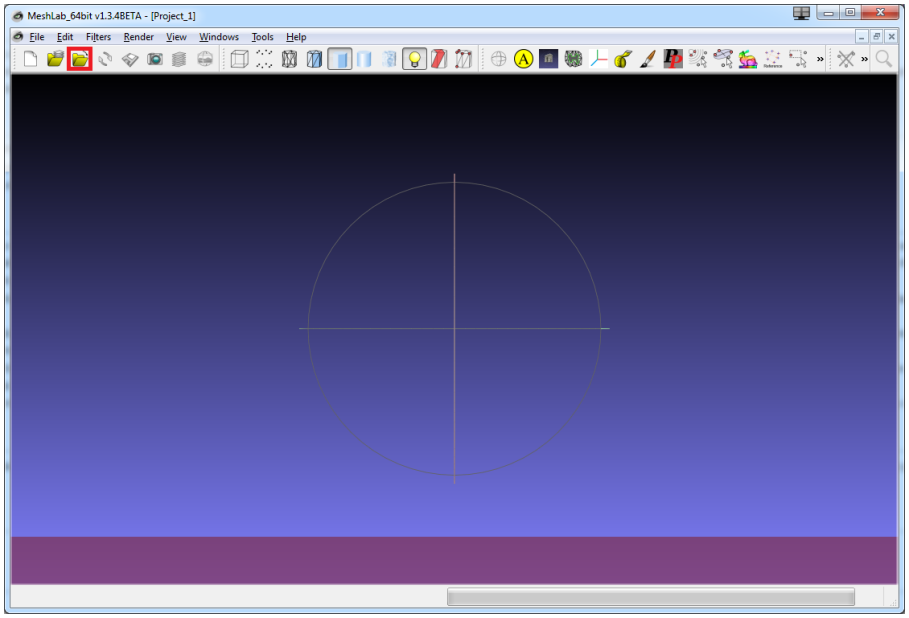


Figure 12: Initial MeshLab window

As a result, you obtain a three-dimensional representation of your mesh (see Fig. 13).

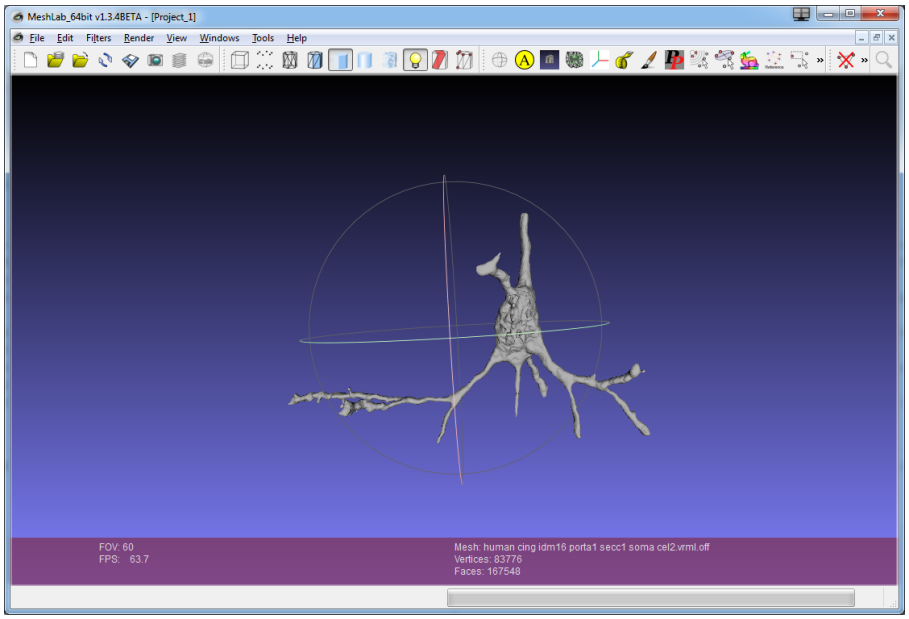


Figure 13: Output example