Neuroinform
DOI 10.1007/s12021-014-9242-5

SOFTWARE ORIGINAL ARTICLE

NeuroMorph: A Toolset for the Morphometric Analysis and Visualization of 3D Models Derived from Electron Microscopy Image Stacks

Anne Jorstad · Biagio Nigro · Corrado Cali · Marta Wawrzyniak · Pascal Fua · Graham Knott

Abstract Serial electron microscopy imaging is crucial for exploring the structure of cells and tissues. The development of block face scanning electron microscopy methods and their ability to capture large image stacks, some with near isotropic voxels, is proving particularly useful for the exploration of brain tissue. This has led to the creation of numerous algorithms and software for segmenting out different features from the image stacks. However, there are few tools available to view these results and make detailed morphometric analyses on all, or part, of these 3D models. We have addressed this issue by constructing a collection of software tools, called NeuroMorph, with which users can view the segmentation results, in conjunction with the original image stack, manipulate these objects in 3D, and make measurements of any region. This approach to collecting morphometric data provides a faster means of analysing the geometry of structures, such as dendritic spines and axonal boutons. This bridges the gap that currently exists between rapid reconstruction techniques, offered by computer vision research, and the need to collect measurements of shape and form from segmented structures that is currently done using manual segmentation methods.

Keywords Neuroimaging software · 3D mesh analysis · 3D image segmentation analysis · Serial section electron microscopy · Neuron · Dendrite · Axon · Synapse · Cell morphology

A. Jorstad · P. Fua
Computer Vision Lab, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

A. Jorstad
e-mail: anne.jorstad@epfl.ch

P. Fua
e-mail: pascal.fua@epfl.ch

B. Nigro · C. Cali · G. Knott (✉)
Centre for Electron Microscopy, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
e-mail: graham.knott@epfl.ch

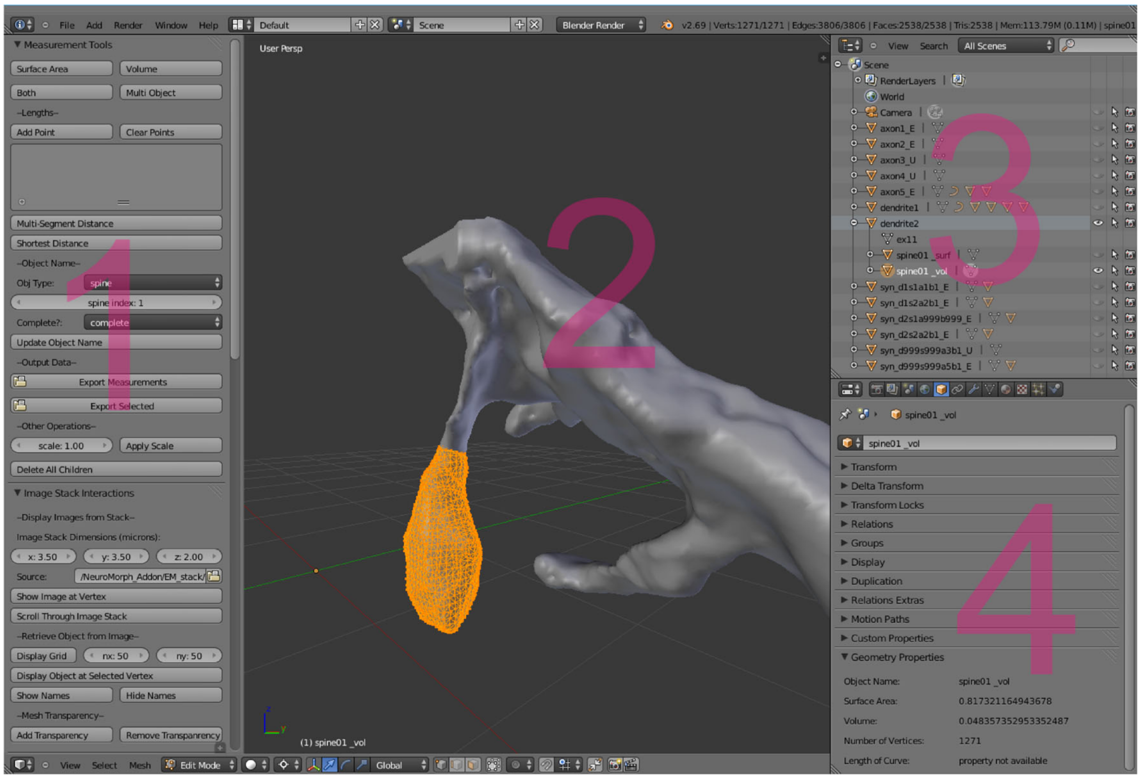
B. Nigro
e-mail: biagio.nigro@epfl.ch

C. Cali
e-mail: corrado.cali@epfl.ch

M. Wawrzyniak
International Institute of Molecular and Cell Biology, Warsaw, Poland
e-mail: marta-wawrzyniak@wp.pl

Introduction

For many years neuroscientists have relied on electron microscopy (EM) to study the morphology of neurons and understand their connectivity within the brain's circuits. This high resolution imaging technique is the only method capable of seeing synaptic connections as well as all the other elements with which they are associated. In recent years, the appearance of block face scanning electron microscopy has led to significant advances in the automated serial imaging through volumes of tissue samples, giving the opportunity to explore the structure of the brain's circuitry in unprecedented detail (Denk and Horstmann 2004; Knott et al. 2008; Briggman and Bock 2012). These 3D imaging methods are able to produce aligned image stacks, with near isotropic voxels, from which algorithms are able to segment features such as neurites, synapses, and mitochondria (Merchán-Pérez et al. 2009; Jain et al. 2010; Kreshuk et al. 2011; Lucchi et al. 2011; Straehle et al. 2011; Vazquez-Reina et al.



```
v2.69 | Verts:1271/1271 | Edges:3806/3806 | Faces:2538/2538 | Tris:2538 | Mem:113.79M (0.11M) | spine01
```

Obj Type: spine
spine index: 1
Complete?: complete
scale: 1.00
Image Stack Dimensions (microns):
x: 3.50 y: 3.50 z: 2.00
Source: /NeuroMorph_Addon/EM_stack/
nx: 50 ny: 50

Object Name:	spine01 _vol
Surface Area:	0.817321164943678
Volume:	0.048357352953352487
Number of Vertices:	1271
Length of Curve:	property not available

Fig. 1 The user interface. 1, The leftmost window shows the geometry tools with which morphological measurements can be made of whole, or parts of models. 2, The 3D view of the model shows all the vertices that comprise a single dendrite segmented from a stack of EM images. An individual dendritic spine protruding from the dendrite has been selected (*yellow vertices*). 3, Use of the volume and surface area measurement functions will create two new children (the meshes *yourname*_**surf** and *yourname*_**vol**), and will be listed in the outliner window. These new meshes will be exact copies of the highlighted region and contain the correct surface area and volume properties, as will be displayed in Geometry Properties field, 4

2011; Ciresan et al. 2012; Becker et al. 2013; Hu et al. 2013). This has significant implications as it allows large-scale segmentations of complex structures, like brain tissue, to be completed in a reasonable time frame. However this creation of 3D models does not directly provide any morphometric data, such as volume, surface area, or length of any of the segmented features. In addition, important subregions, such as synaptic boutons situated on axons, or spines that protrude from dendrites, for which these types of measurements are important, cannot be identified automatically in these models and need to be delineated by hand. This is time consuming, and to date can only be achieved by manually segmenting identified features directly from the image stacks.

To address these issues, and capitalize on the effectiveness with which algorithms can generate mesh models from a stack of EM images, we have developed software with which users can make measurements directly on these segmentation results in a 3D workspace. Our software package, NeuroMorph, integrates into the Blender open source modeling software and allows the user to measure, annotate, and manipulate features. A screen shot of the tool's user interface can be seen in Fig. 1.

Description of NeuroMorph

NeuroMorph is a set of tools to be used in the Blender[1] 3D modeling software for performing morphometric analyses of mesh models. Blender is open source and used extensively in the 3D modeling community. The NeuroMorph toolset is written primarily for the analysis of models of neuronal elements, such as axons and dendrites derived from serial EM image stacks. It is often important to understand the size, shape, and connectivity of these complex structures. Here, the segmentation reconstruction software ilastik (Kreshuk et al. 2011) and TrakEM2 (Cardona et al. 2012) were used to reconstruct some axons, dendrites, and synapses from a single series of images from the cerebral cortex of an adult rat, taken with a focussed ion beam scanning electron microscope. However, mesh models from any source can be used. The typical measurements sought from these types of structures include volumes, surface areas, and lengths, across different parts of the individual models.

[1]www.blender.org

The toolset comprises three parts. A measurement tool allows users to select any region of a model to calculate its volume and surface area, and also measure distances. Objects can also be labeled consistently to create new names for the subregions measured, and these calculated measurements can be exported. A second tool accesses the image stacks from which the models were originally derived. Any point on a model can be selected, and the corresponding image in the stack displayed at this location. With this tool the correspondence between the reconstructed model and the original images are revealed, providing a means by which structures can be identified and labeled. In the case of dense reconstructions, where many models are reconstructed from a single image stack, it is often necessary to search for different structures. By pinpointing these in the original image, the user can then display the corresponding mesh. Also provided is an import tool with which .obj formatted models, obtained from segmentation software, can be opened, ready for analysis. During this import process, the number of points that comprise the mesh can be reduced, decreasing its file size, then scaled to the correct units of measure. In the example shown here the units are microns. The Neuromorph tools assume that the mesh models are accurate representations of the segmented objects, with their surfaces adequately smoothed to produce accurate morphological measurements. This part is to the discretion of the user to ensure that the parameters used in this import phase do not alter significantly the shape of the object.

Morphometric Analysis Using NeuroMorph

When a 3D mesh surface is loaded into Blender and visible in the viewing window (Fig. 1), regions can be selected and measurements of volume, surface area, and length made. The user highlights the part of the mesh, defines the name of this feature either manually or by using the "Object Name" interface, then clicks on the "Surface Area" or "Volume" buttons. These create two new Blender mesh objects as children of the original object, and in the Object Properties, a new "Geometry Properties" field is populated with analytical information about the new objects (Fig. 1 region 4).

The first new object, called *yourname*_**surf**, is an exact copy of the highlighted region of the mesh, and its surface area is exact. The second new object, called *yourname*_**vol**, closes any open holes of the highlighted region, which is necessary for the volume calculation, as explained later. The volume property of this second object is the exact volume of the original highlighted region, and its surface area includes the surface area of any surface faces that were added to close any holes in the highlighted mesh, so may be larger than the surface area of *yourname*_**surf**.

To measure distances, the user selects several points through which a piecewise linear curve is fitted, and the distance of this curve is provided in the 'Geometry Properties". Alternately, a second length-measuring tool is provided which calculates the shortest distance between exactly two points on a mesh via a path through the vertices of the mesh surface itself.

The Surface Area Calculation

A mesh object consists of connected vertices that form its surface. Each vertex is connected to its neighbors by edges, which are then connected to defined polygonal faces. The surface area of the object is the sum of the areas of each of the individual faces. To calculate the surface area, all faces contained in the selected mesh are converted to triangles, which means that any face with $n > 3$ sides is divided into $n - 2$ triangles.

The area of a triangle, given the 3D coordinates of its three vertices $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$, is calculated via the area of the parallelogram defined by two sides of the triangle, which is equal to the norm of the cross product of the two edges. The area of the triangle is half this value, $A = \frac{1}{2}\|(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)\|$.

The Volume Calculation

Calculating a volume is only meaningful for a closed mesh with no holes. Any open region in the mesh structure must therefore be closed prior to the volume calculation. A hole is detected by finding edges that are only connected to a single face. These edges can be grouped together into strings of edges that share consecutive vertices, and every closed string defines a single hole to be closed (Fig. 2). A new vertex located at the center of each hole is added to the mesh. New triangular faces defined by each boundary edge connected to the new vertex are added to close the hole. The normal vector to the face, which is the vector pointing in the direction perpendicular to the face surface, is stored along with the face.

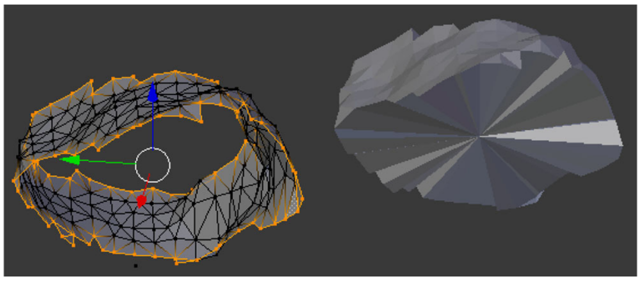


Fig. 2 An open region of a mesh with its open boundaries highlighted, and a copy of the mesh after its holes have been closed for the volume calculation

The orientation of the new faces must be consistent with that of the original mesh for the volume calculation to be correct; that is, the normal vector of each face must be pointing outwards, by convention, rather than towards the inside of the mesh. However, the normal vector may have been added incorrectly when the new faces were added. The following algorithm is used to determine whether the orientation of a new face, f_1, is consistent with that of the face in the original mesh that it connects to, f_0. Take e to be the edge shared by the two faces, defined by vertices v_a and v_b, take vertices v_0 and v_1 to be the third vertex of each of the triangular faces, respectively, and take n_0 and n_1 to be the normal vectors of each of the faces, respectively.

$$s_0 = (v_0 - v_a), \text{ the edge connecting } v_a \text{ to } v_0 \quad (1)$$

$$s_1 = (v_1 - v_a), \text{ the edge connecting } v_a \text{ to } v_1 \quad (2)$$

$$c_0 = s_0 \times e, \qquad c_1 = e \times s_1 \quad (3)$$

$$d_0 = n_0 \cdot c_0, \qquad d_1 = n_1 \cdot c_1 \quad (4)$$

$$d_{\text{indicator}} = d_0 d_1 \quad (5)$$

If $d_{\text{indicator}} < 0$, then the orientation of the normal vector of face f_1 must be flipped.

To calculate the volume of a closed mesh, the signed volumes of the tetrahedra defined by each triangular face connected to the origin are calculated, and these are summed for the total. The signed volume of a tetrahedron connecting vertices v_1, v_2, v_3 with the origin is computed by the standard formula

$$V = \frac{v_1 \cdot (v_2 \times v_3)}{6}. \quad (6)$$

It is positive if the normal vector to the triangle is pointing away from the origin and negative if the normal is pointing towards the origin. The sum of all the signed tetrahedral volumes defined by a mesh surface produces the correct volume of the region contained in the mesh, which can also be understood as an application of the divergence theorem from calculus (Fig. 3).

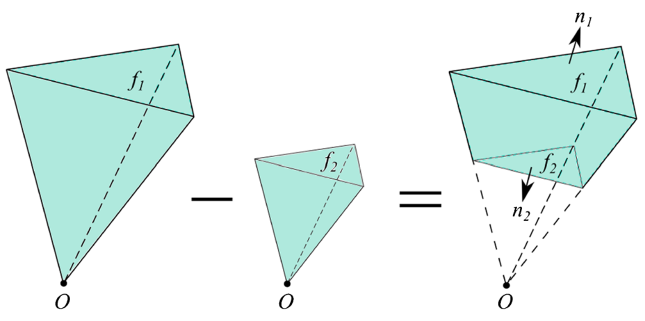


Fig. 3 The volume of the shaded region between faces f_1 and f_2 on the right is equal to the volume of the tetrahedron formed by face f_1 with the origin (labeled O) minus the volume of the tetrahedron formed by face f_2 with the origin. The signed volume of the first tetrahedron is positive because its normal vector points away from the origin, while the signed volume of the second tetrahedron is negative because its normal vector points towards the origin, and so the sum of the two signed volumes produces the desired volume of the region between the two faces

The Length Calculation

In addition to the 2-dimensional and 3-dimensional measurements described above, the software is also able to estimate one-dimensional lengths along surface profiles. The length measurement is based on the interpolation of a user-defined set of vertices by a piecewise linear function.

Two length tools handle various measurement requirements. The first calculates the length of a curve constructed from a set of user-defined points (Fig. 4a-b). The user selects any number of vertices along a mesh, and the software constructs linear segments to connect the points into a continuous curve, creating a new object to store this multi-segment distance called *yourname*_**MS_dist** as a child of the original mesh. This new object contains the curve length in its Geometry Properties panel. The precision of the measurement, compared to the length of the curve that lies exactly along the surface of the selected mesh profile, depends entirely on the user input sampling, which can be tedious for very irregular surfaces. However, this tool allows the user to calculate the distance between any two points in space, regardless of the surface geometry.

A second length measurement tool is provided that calculates the length of the curve lying along the surface of a mesh connecting any two specified vertices (Fig. 4c). Using the “Shortest Distance” button, an internal Blender function is called that finds the shortest path between vertices through successive edges connecting them on the mesh. This tool first temporarily adds supplementary edges to the mesh to connect all vertices on each mesh face. This allows the shortest path to have the option of passing directly between any two vertices across any given face without having to traverse its boundary edges. If the provided remesh function has been used during the import, the faces of the mesh will be near-regular quadrilaterals, as this requires fewer computational resources to maintain, compared to triangles. For quadrilateral faces, which are also encouraged by the developers of Blender, edges are added to connect the two pairs of opposite vertices on the face. For general n-sided faces, a vertex at the center of mass of the face is created, adding edges connecting this vertex to each of the original vertices of the face. After these temporary edges have been added, the shortest path calculated through the edges of the mesh will be the piecewise linear shortest path through the mesh vertices along the surface. The accuracy of this length calculation is increased with increased mesh density. The new object, a child of the original mesh, is called *yourname*_**2pt_dist** and the performed length measurement is stored in the mesh Geometry Properties.

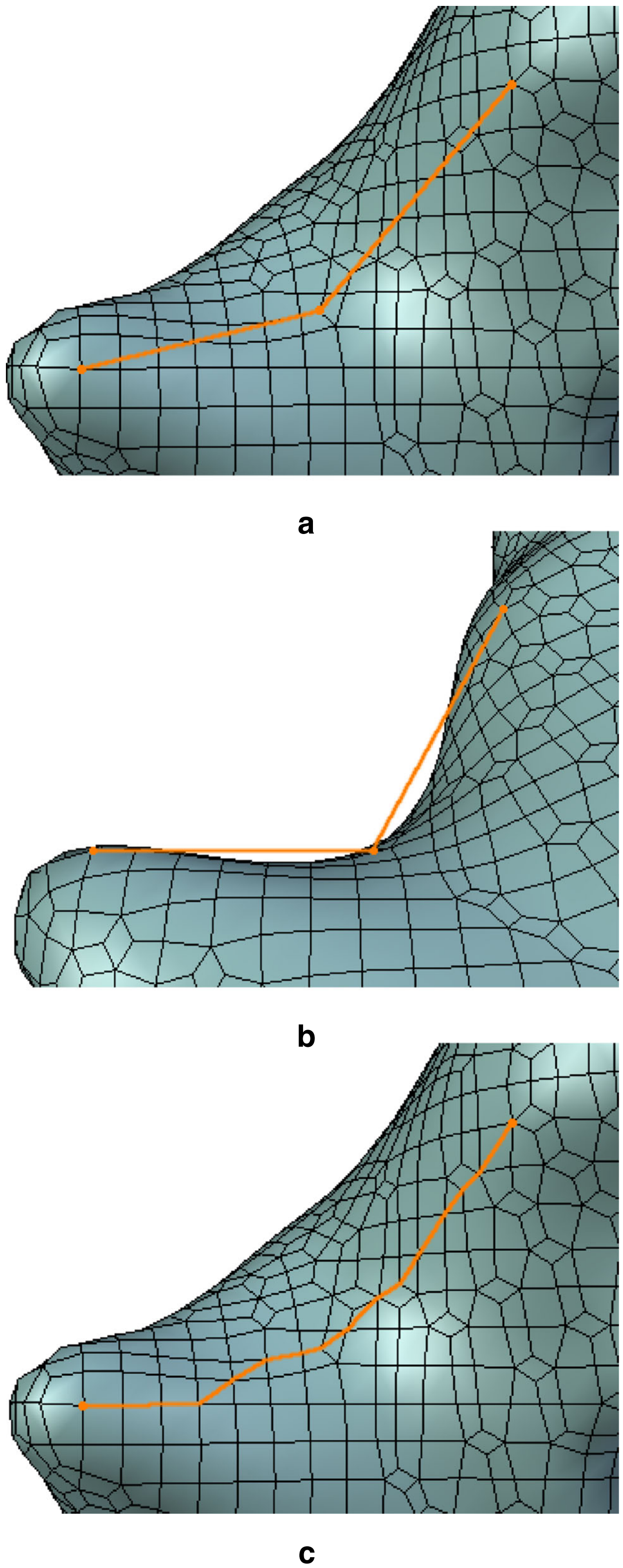


Fig. 4 Examples of curves generated by the length calculation. **a** The multi-segment distance connects any number of user-defined points with linear segments. **b** The multi-segment distance from a different angle. **c** The shortest distance between two user-defined points through vertices on the surface of the mesh

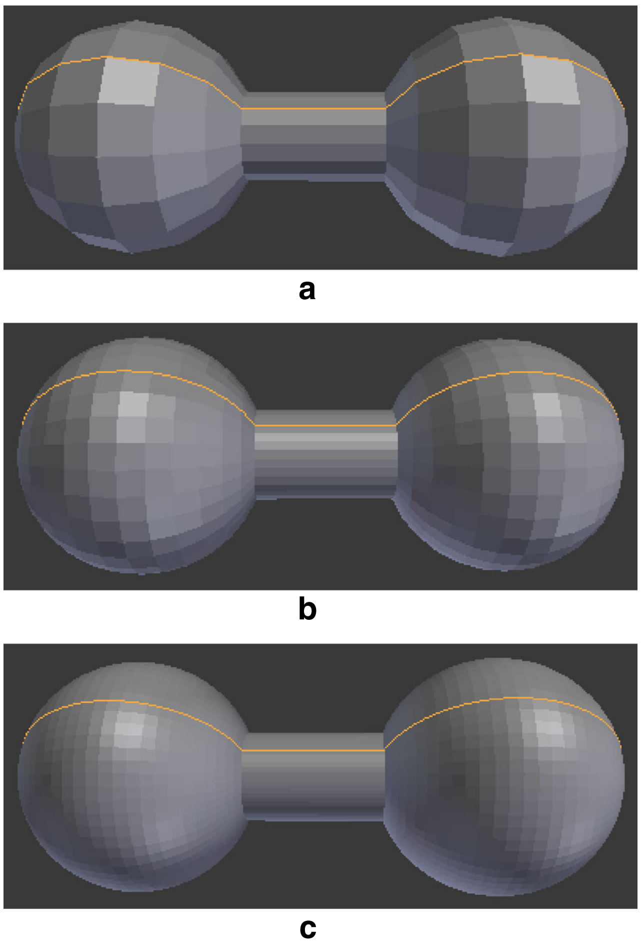


Fig. 5 The dumbbell objects compared in Example 1, constructed using (**a**) 242, (**b**) 930, and (**c**) 3650 vertices

Guiding Examples

This section provides some simple examples to demonstrate the numerical properties of the measurement tools. The tools return exact numerical measurements of the meshes provided, but if a mesh is not sufficiently smooth then the resulting measurements may not adequately describe the structure being studied. In general we will see that as more vertices are used the measurements will become more precise. However, using too many vertices might result in measuring unwanted noise. It is up to the user to ensure that the meshes being measured are sufficiently accurate models of the underlying 3D structures being studied, and this section is included to provide a rough guide as to the types of behavior to expect when using 3D mesh models. We also provide examples of extreme cases where surface area and length measurements might be less precise than expected, so that users are aware of some limitations when working with meshes.

Table 1 Table of measurements of the dumbbell structures of varying mesh refinements. As the mesh more finely approximates the true structure, the measurement becomes more accurate

# Vertices	Surface Area	Volume	Length
242	26.17	8.35	6.61
	(97.1%)	(94.1%)	(99.5%)
930	26.75	8.74	6.64
	(99.3%)	(98.5%)	(99.87%)
3650	26.90	8.84	6.648
	(99.8%)	(99.6%)	(99.97%)
True Surface	26.95	8.87	6.65

Example 1 Numerical Accuracy. Take two spheres with radius 1, located with centers a distance 3 apart, connected by a cylindrical crossbar with radius $\sin(\frac{\pi}{8})$, to form a dumbbell (Fig. 5). Representations of this object are constructed with increasingly finer meshes, and the surface area and volume of each are measured, as well as the distance along each mesh between its two extreme endpoints. The results are provided in Table 1. As more vertices are used, the mesh representation gets closer to the true mathematical surface, and the accuracy of the numerical measurements increases. The errors here are caused only by the fact that the mesh points are discretely sampled from a continuous surface, and so the finer the mesh, the more accurate the model.

Example 2 Mesh Refinement Limitations. Fig. 6 provides an example where using more vertices does not improve accuracy, but instead brings in extra noise that can interfere with the measurements. In this example, the sharp jitters on parts of the surface are probably caused by image noise or errors in the image segmentation process, and are unlikely to be modeling true features of the underlying structure being studied. The measurement most affected by this over-refinement is generally the surface area, see Example 3. In this example, the meshes were imported using octree depths of 11 and 7, respectively, see Section "Importing .obj Files Into Blender" for details. It is up to the user to ensure that the mesh structure being used is an accurate representation of the true object being studied.

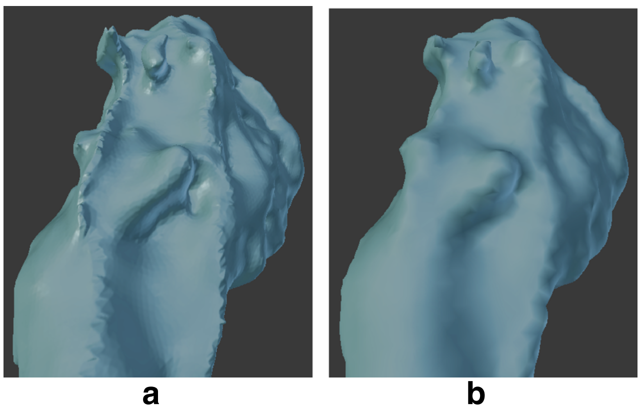


Fig. 6 **a** Example of a mesh with too many vertices, capturing unwanted noise. **b** A better, smoother refinement of this particular mesh uses fewer vertices

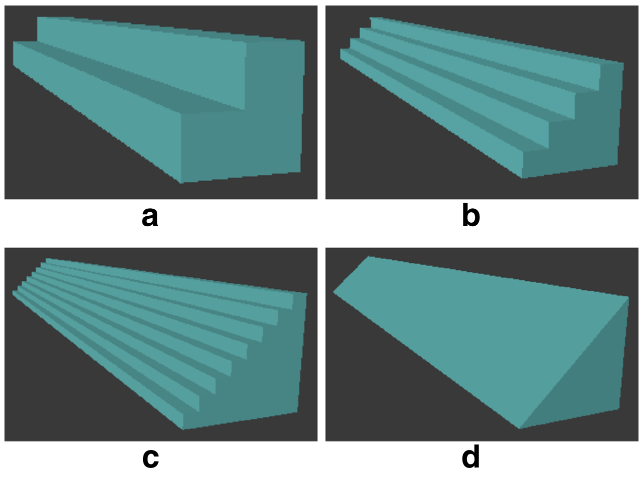


Fig. 7 Four ramps of varying refinement. As the mesh better approximates the ramp, the volume gets closer to the ramp volume, but the surface area is much higher because the surface is locally rough

Example 3 Surface Area Limitations. A mesh that is not sufficiently smooth can result in surface areas that are noticeably larger than the smooth surfaces they represent, while the volume calculation is more stable. Figure 7 shows four ramps of varying refinement, all of width 4, height 4 and length 16. As a higher number of smaller steps are added to better approximate the smooth ramp surface, the volume approaches the true volume, but the surface area remains significantly higher (in fact, the surface area only changes on the two side faces of the ramp). See Table 2 for numbers. This example serves as a warning to ensure that all studied meshes are sufficiently smooth, else small amounts of variation across the surface may cause the measurements of the mesh to not adequately describe the true structure being studied.

Table 2 As the number of steps from Fig. 7 increases and each step gets smaller, to better approximate the ramp, the volumes gets closer to the ramp volume, but the surface areas remains much higher, due to the surface remaining locally rough

# Steps	Surface Area	Volume
2 (a)	134.0	48.0
4 (b)	133.0	40.0
8 (c)	132.5	36.0
True Ramp (d)	113.25	32.0

Example 4 Length Measurement Limitations. An example of the limitations of the length measurement is provided in Fig. 8. Since the paths generated by our Shortest Distance function are required to pass between consecutive mesh vertices, instead of across any part of any face, cases like the left path in Fig. 8 arise. If the user requires a more accurate measurement for specific cases like this, they are encouraged to use the Multi-Segment Distance function, with which the user can click on as many mesh points as needed, and the path connecting these points with linear segments will be returned. While it is possible to construct a path through a mesh surface that is not limited to passing through adjacent vertices, the amount of calculations required results in a function that is currently too slow, on general meshes, to be useful in the setting for which this tool was designed. The tool is limited by the speed at which the Python programming language is able to perform these numerical operations. The tool as provided can return a general vertex-based path on a mesh of several tens of thousands of vertices in under two seconds running Blender on a laptop. Considering that the mesh is an approximation of the true biological structure, and that the accuracy of this approximation can generally be improved by increasing the density of vertices on the mesh, the potential error in the length calculation given its speed is within an acceptable range for practical use.

Integration With EM Image Stacks

The NeuroMorph toolset is aimed at the analysis of 3D models resulting from segmentations of EM image stacks. Performing direct measurement of the 3D structures themselves means that the user does not have to visualize them through a series of 2D planes, and makes the operation faster and more intuitive. However, with this tool the user is able to visualize the original image stack superimposed on the mesh objects during this analysis process, so that any classification of substructures, such as spines on dendrites, can be done correctly. The tool includes functions for the user to refer to an original image in the stack, and to scroll through them, with each image appearing at its correct Z position relative to the mesh objects. Also the ability to search for different meshes, by using each image as a references map onto which the user can select positions with the tool showing the corresponding mesh situated at this position.

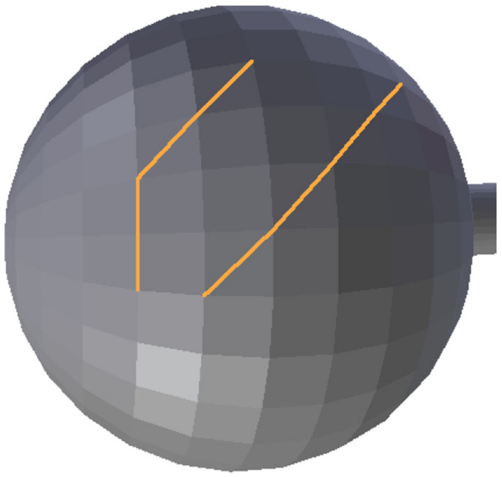


Fig. 8 The “Shortest Distance” length measurement is limited to passing through consecutive vertices

The simultaneous visualization of mesh models and images at their corresponding Z position is important as a means of correctly assigning names to the different objects. This is particularly relevant for models that may be associated with others, but have been reconstructed separately. Synapses, for example, are the site of contact between axons and dendrites, and reconstructed as isolated objects. In the presentation of the final data, these structures must be labelled according to which axon and bouton they are located, as well as to which dendrite they contact. Synapses are labelled, therefore according to the number of the dendrite, spine (‘0’ if synapse on the dendritic shaft), axon, and bouton; and also whether the synapse is excitatory or inhibitory (see Fig. 9 for an example).

Visualization Tool

The visualization tool can rapidly upload the bitmap images from which the models were reconstructed using only limited memory allocation because editing or rendering is not possible. The positioning of a specific image from a stack is achieved by using its position along the z-axis. The correspondence is between the total number of slices of the stack and the total depth of sectioned material. The user is able to define the image dimensions and the mesh objects in microns. The tool then allows the user to scroll through the image stack, visualizing the serial images in rapid succession at their correct z position on the models. This functionality allows the user to check the segmentation and importation process to ensure that the objects correspond precisely with the original images and the segmentation was correct.

Object Retrieval

When large number of models are imported into a single Blender file it can be challenging to locate specific structures within a defined region. For example, to locate a particular dendrite that shares a synaptic contact with a particular axon requires all dendrites to be made visible, one at a time, until the correct one is found. To speed up this

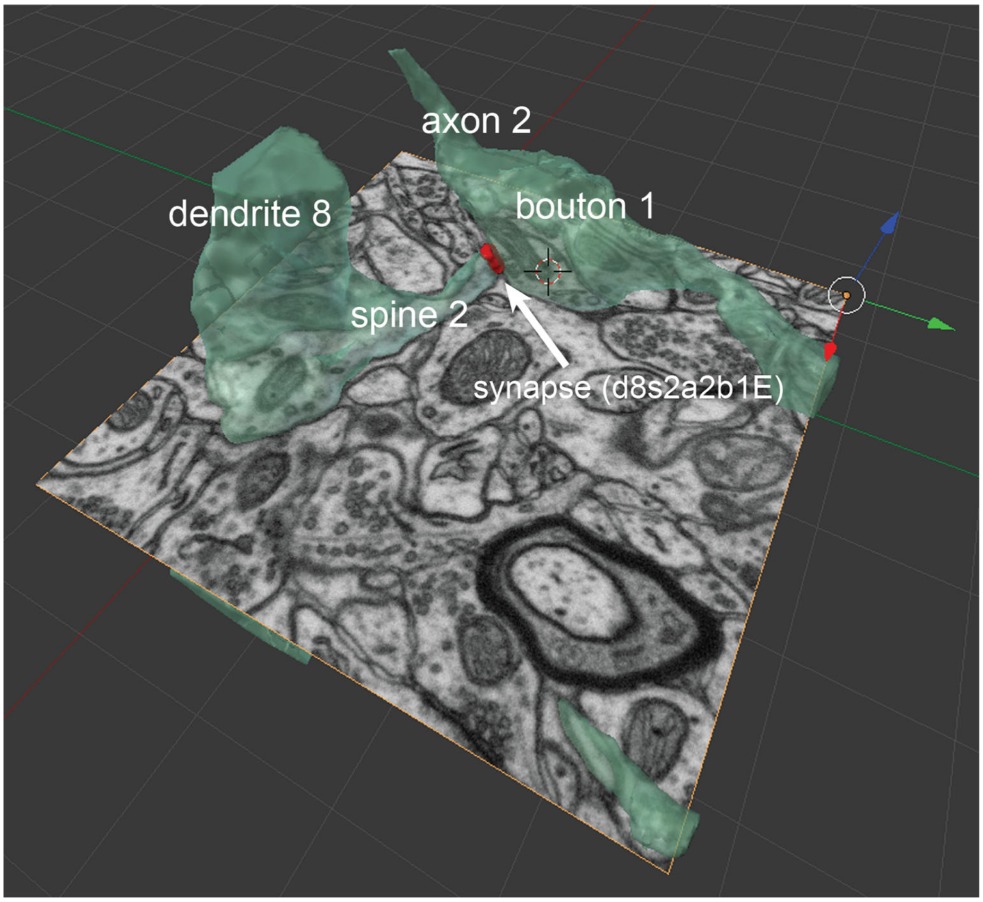


Fig. 9 The models can be viewed simultaneously with the images from which the reconstructions were originally made. They are displayed at their correct position in the volume so that the correspondence between the image and the model can be assessed. Scrolling with the mouse through the image stacks places each image at its corresponding z position. Shown here are dendrite and axon models (*transparent green*) with a connecting synapse (*red*). These models are assigned numbers, shown as white labels in this figure, and these are used to assign a unique identifier to the synapse. In this case the synapse is found on spine 2 of dendrite 8, and bouton 1 of axon 2. This gives in the name **syn_d8s2a2b1E**. The 'E' indicates that the synapse is a putative excitatory synapse

search process, an image from the stack can superimposed on the model (explained above). Once this image is displayed, a grid overlying the image plane is then added. This allows the user to select a specific vertex in this grid that lies inside the object (axon in this case) that is being sought. The retrieval process is then activated to identify the mesh that encloses the selected point. In this multi-step retrieval process, the bounding box of each known mesh is checked to determine which mesh potentially contains the selected point. To further reduce the number of potential meshes considered, if the minimum distance from the selected point to an object's set of vertices is too large, it is excluded. Finally, a ray casting is performed from the selected point along the six cartesian directions, and the number of intersections between the rays and each mesh are counted. If the number of intersections of each ray with a mesh is odd, then the point is on the inside of that closed mesh surface. The mesh object found to contain the point is then made visible.

Importing .obj Files Into Blender

The toolset also provides an import function for opening .obj format mesh models generated from segmentation software. For this phase, information is needed concerning the model's scale so that all measurements in the subsequent analysis correspond to known units. This is based on the arbitrary unit system contained within the Blender software (Blender units). Here, the user can rescale the meshes such that a Blender unit is equivalent to a unit of mest, here microns are used. The scaling is achieved by defining the number of pixels per micron, in the original images and from which the models are made.

In addition, this tool includes a function for reducing the number of points (vertices) in the models. Segmentation software often produces models with high density meshes containing many vertices, resulting in large file sizes. Many of these points do not add a greater level of morphological detail to the structure so are unnecessary. In such meshes, the removal of a significant proportion of the vertices does not change their overall shape. Surfaces of objects that should be smooth are often rough, by way of this high vertex density, and this can interfere with accurate measurements. Many existing software tools are able to perform mesh smoothing and downsampling operations, including freely available functions in Blender, MeshLab,[2] and

[2]meshlab.sourceforge.net

plugins for Fiji,[3] or for-purchase software such as 3DS Max[4] or Maya[5]. The remesh tool we provide is not fundamentally different from any of these options, and it is included only for convenience.

The import function provides a “Remesh” modifier integrated into Blender that generates a new mesh based on the original surface. The algorithm (Ju et al. 2002) is able to produce meshes with fewer vertices, giving different levels of structural detail depending on the settings used. Selection of the most suitable remesh setting (the “Octree Depth” and whether or not to use “Smooth Shading”) is done on the basis of how closely the mesh model fits with the segmented structures, while substantially reducing the vertex count, see Fig. 1. Typical remesh operations during import are able to halve the number of vertices, without disturbing the model’s morphology.

The measurement and visualization tools in NeuroMorph will work with any mesh and image stack, regardless of the method of segmentation. The accuracy of the measurements of the segmented structures depend on how closely the final model represents the imaged structure, and how carefully the vertices selected. The simultaneous imaging of the models and the image stack help to check the accuracy for reconstructions. With these tools the user can then decide how accurately the measurements from the mesh reflect the true biological structure.

Installation of NeuroMorph

NeuroMorph comprises three Python scripts available for download.[6] Modules have been implemented to be cross-platform and do not require any additional installations. They are based on the Blender 2.70 Python API which requires Python 3.x. The installation procedure is straightforward and described in detail within the NeuroMorph documentation site site.[7] It is important to note that although the measurement functions provide a general and comprehensive tool for volume, surface, and length estimates of mesh objects, the use of the import and visualization modules are intimately related to models reconstructed from image segmentations. This means that meshes must be reconstructed and rescaled properly to obtain the correct superposition of image slices according to their size and position in the Blender reference system. Although the toolset has been constructed on the latest Blender version (2.70), it cannot be excluded that during the evolution of Blender’s Python API, the tool could experience errors in future versions. However, it is hoped that any conflicts will be corrected in time.

A sample image stack and object meshes are also provided for download.[8] These include a blender file containing the models of two dendrites, five axons, and their synapses, as well as two .obj files derived from the ilastik software. These were reconstructed from the stack of serial images using the interactive carving feature in ilastik (Straehle et al. 2011). The images were taken using focused ion beam scanning electron microscopy of a resin embedded sample of adult rat brain. The images are 3.5 microns in width and height, and the total stack of images is 3.5 microns in depth.

Discussion

Serial electron microscopy imaging and automated procedures for segmentation and reconstruction promise significant new opportunities for analysing cell structure on an unprecedented scale. However, for the moment at least, few are able to make any distinctions between the types of structures recognized. Synaptic contacts, for example, can be detected automatically from electron microscopy images (Kreshuk et al. 2011; Becker et al. 2013), but not classified according to their morphology as to whether they are symmetric, and presumed inhibitory, or asymmetric, and excitatory. Automatic segmentation of neurites is now achievable, but these cannot be separated as either axons or dendrites. Neither can the automated recognition of features such as boutons, or dendritic spines. Though it seems likely, and crucial for large scale connectomics studies, that future algorithms will be able to integrate many different morphological features to give accurate and unbiased measurements of a range of structures, today this can only be done manually. NeuroMorph provides a convenient means of making these further analyses on the current segmentation results. The classification and analysis process carried out on 3D models is more efficient than manual segmentation across serial 2D images. Features are measured more rapidly by selecting them in 3D rather than having to identify them on each image of a stack. Additionally, complex structures can be classified and labeled by visualizing them in 3D, together with the images from which they were reconstructed. This process of identification is crucial for any analysis in which separately reconstructed features may be related to others in the vicinity. As an example, synapses connect two different neurites, therefore it is necessary to use a consistent naming strategy so that in the final data

[3] fiji.sc/Fiji

[4] www.autodesk.com/products/autodesk-3ds-max

[5] www.autodesk.com/products/autodesk-maya

[6] cvlab.epfl.ch/NeuroMorph

[7] wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Neuro_tool

[8] cvlab.epfl.ch/NeuroMorph

output it is clear which synapse is attached to which dendrite and axon. NeuroMorph is constructed so that the user can identify, label, measure all the connected elements, without leaving the 3D environment. Dense reconstructions from single volumes may contain many hundreds of objects. The Neuromorph tools enable the precise organization of such datasets as well the ability to search for objects. Though the examples given in this paper show only limited numbers of elements within a small volume, it is scalable to any number of meshes across far larger stacks of images, dependent only on the limitations of the computer being used. Constructing the NeuroMorph toolset within Blender has not only enabled the visualization and analysis to be carried out in a 3D environment, but within a well-developed software containing a vast collection of functionalities for the modeling community. This provides the opportunity to exploit these functions and use the meshes of neuronal elements to make further analyses by modeling different physiological processes. The Blender software has previously been used for several biomedical applications, including the integration of another neuron visualization tool called Py3DN (Aguiar et al. 2013) for importing and displaying whole cell reconstructions made with the Neurolucida reconstruction system (Microbrightfield, USA) from light microscopy. This gives users the tools to perform large-scale analyses of parameters such as dendritic tree lengths, and bouton (varicosity) densities. Blender's versatility, therefore, for integrating any number of 3D analysis tools gives it huge potential as an open source platform with which imaging scientists are able to use the results of computer vision research.

Information Sharing Statement

All code for the NeuroMorph Toolkit (RRID:SciRes_000156) is available in open source under the GNU General Public License as published by the Free Software Foundation, and is available for download at cvlab.epfl.ch/NeuroMorph. Example meshes and detailed documentation are also provided at this link. The toolkit functions as an add-on within the Blender open source modeling software (RRID:nif-0000-31943), available at www.blender.org.

Acknowledgments The work was supported by the Novartis Foundation for Medical-Biological Research (G.K.) and EU ERC MicroNano Project (P.F.).

References

Aguiar, P., Sousa, M., Szucs, P. (2013). Versatile morphometric analysis and visualization of the three-dimensional structure of neurons. *Neuroinformatics*, *11*, 393–403.

Becker, C., Ali, K., Knott, G., Fua, P. (2013). Learning Context Cues for Synapse Segmentation. *IEEE Transactions on Medical Imaging*, *32*, 1864–1877.

Briggman, K., & Bock, D. (2012). Volume electron microscopy for neuronal circuit reconstruction. *Current Opinion in Neurobiology*, *22*, 154–161.

Cardona, A., Saalfeld, S., Schindelin, J., Arganda-Carreras, I., Preibisch, S., Longair, M., Tomancak, P., Hartenstein, V., Douglas, R. (2012). TrakEM2 software for neural circuit reconstruction. *PLoS One*, *7*, e38,011.

Ciresan, D., Gambardella, L., Giusti, A., Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *NIPS*, 2852–2860.

Denk, W., & Horstmann, H. (2004). Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biology*, *2*, 1864–1877.

Hu, T., Nunez-Iglesias, J., Vitaladevuni, S., Scheffer, L., Xu, S., Bolorizadeh, M., Hess, H., Fetter, R., Chklovskii, D. (2013). Electron Microscopy Reconstruction of Brain Structure Using Sparse Representations Over Learned Dictionaries. *IEEE Transactions on Medical Imaging*, *32*, 2179–2188.

Jain, V., Bollmann, B., Richardson, M., Berger, D., Helmstaedter, M., Briggman, K., Denk, W., Bowden, J., Mendenhall, J., Abraham, W., Harris, K., Kasthuri, N., Hayworth, K., Schalek, R., Tapia, J., Lichtman, J., Seung, H. (2010). Boundary learning by optimization with topological constraints. *IEEE Xplore*, 2488–2495.

Ju, T., Losasso, F., Schaefer, S., Warren, J. (2002). Dual contouring of hermite data. *ACM Transactions on Graphics (TOG)*, *21*(3), 339–346.

Knott, G., Marchman, H., Wall, D., Lich, B. (2008). Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *The Journal of Neuroscience*, *28*, 2959–2964.

Kreshuk, A., Straehle, C., Sommer, C., Koethe, U., Cantoni, M., Knott, G., Hamprecht, F. (2011). Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images. *PLoS One*, *6*, e24,899.

Lucchi, A., Smith, K., Achanta, R., Knott, G., Fua, P. (2011). Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features. *IEEE Transactions on Medical Imaging*, 30.

Merchán-Pérez, A., Rodriguez, J., Alonso-Nanclares, L., Schertel, A. (2009). Counting Synapses Using FIB/SEM Microscopy: A True Revolution for Ultrastructural Volume Reconstruction. *Frontiers in Neuroanatomy*, 3.

Straehle, C., Köthe, U., Knott, G., Hamprecht, F. (2011). Carving: scalable interactive segmentation of neural volume electron microscopy images. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011*, *14*, 653–660.

Vazquez-Reina, A., Gelbart, M., Huang, D., Lichtman, J., Miller, E., Pfister, H. (2011). Segmentation fusion for connectomics. *IEEE Xplore*, 177–184.