SoftwareX 30 (2025) 102081

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

MitoSeg: Mitochondria segmentation tool

Faris Serdar Taşel [a,*], Efe Çiftci [b]

[a] Department of Computer Engineering, Çankaya University, Ankara, Turkey
[b] Computer Programming Program, Çankaya University, Ankara, Turkey

ARTICLE INFO

Keywords:
Electron tomography
Mitochondrion
Segmentation

ABSTRACT

Recent studies suggest a potential link between the physical structure of mitochondria and neurodegenerative diseases. With advances in Electron Microscopy techniques, it has become possible to visualize the boundary and cristae structures of mitochondria in detail. Segmenting mitochondria from microscopy images remains challenging due to image quality and complex morphology of mitochondria, including cristae and the other subcellular structures. It is crucial to automatically segment mitochondria from images exhibiting different mitochondrial boundary and crista characteristics to investigate the relationship between mitochondria and diseases. In this paper, we present a software solution for mitochondrial segmentation using an automatic validation scheme based on the general physical properties of mitochondria, highlighting boundaries in electron microscopy tomography images and generating corresponding 3D meshes. These capabilities help researchers conduct further investigations into mitochondrial morphology and explore its role in the mechanisms of neurodegenerative diseases.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00483
Permanent link to Reproducible Capsule	https://github.com/fstasel/mitoseg/tree/master/docker
Legal Code License	GPLv3
Code versioning system used	git
Software code languages, tools, and services used	C++, Python, CMake, Docker
Compilation requirements, operating environments & dependencies	A modern GNU/Linux system with development files for OpenCV, Boost, and yaml-cpp libraries installed. python3-tk is optional for GUI.
If available Link to developer documentation/manual	https://github.com/fstasel/mitoseg#readme
Support email for questions	fst@cankaya.edu.tr

1. Motivation and significance

Mitochondria are organelles responsible for producing the chemical energy required for various biochemical reactions in the cell. The relationship between mitochondria and neurodegenerative diseases such as Alzheimer's and Parkinson's has become an area of increasing interest, as understanding the causes of these diseases is of great importance [1–7]. For this reason, it is essential to study the physical structure of mitochondria.

Advances in electron microscopy imaging techniques have significantly impacted the investigation of subcellular structures. Among these techniques, Serial Block-Face Scanning Electron Microscopy (SBF-SEM), Transmission Electron Microscopy (TEM), Focused Ion Beam Scanning Electron Microscopy (FIB-SEM) and Automated Tape-collecting Ultramicrotome SEM (ATUM-SEM) are frequently used, offering detailed imaging down to a few nanometers [8–10]. Such high-resolution imaging allows for the observation of mitochondrial membrane structures, including the boundary and internal regions.

Mitochondria can exist in distinct structural states, including the condensed state, where the internal matrix is tightly packed with proteins, resulting in a dense, blob-like appearance in electron microscopy

* Corresponding author.
E-mail addresses: fst@cankaya.edu.tr (Faris Serdar Taşel), efeciftci@cankaya.edu.tr (Efe Çiftci).

[1] https://library.ucsd.edu/dc/object/bb03438047
[2] https://library.ucsd.edu/dc/object/bb82620133

https://doi.org/10.1016/j.softx.2025.102081
Received 9 September 2024; Received in revised form 19 January 2025; Accepted 30 January 2025
Available online 9 February 2025

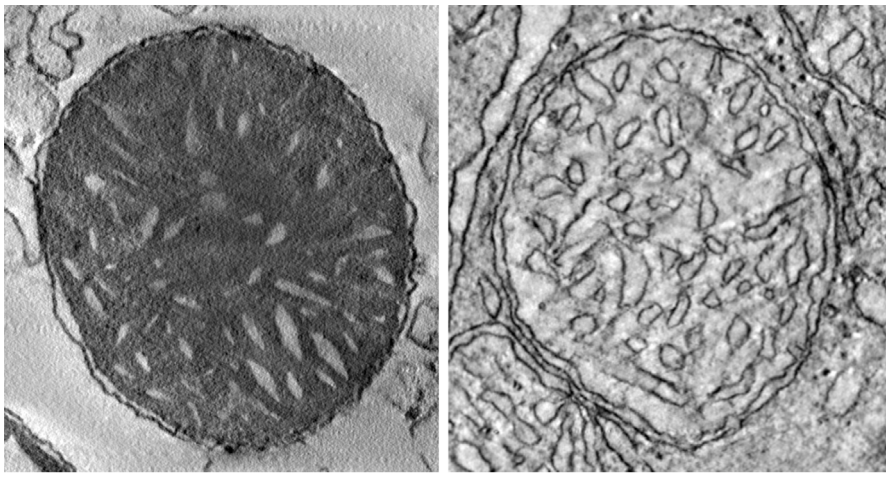

Fig. 1. Sample mitochondria images taken from Cell Centered Database, (a) Accession number: 3864 and (b) Accession number: 54.

images[1] (Fig. 1(a)). This state contrasts with the orthodox state, in which the mitochondria exhibit a more relaxed configuration with visible cristae structures. In order to enhance the visibility of the crista arrangement, a preprocessing step known as heavy metal staining is applied. This technique highlights the fine details of mitochondrial structures, allowing for detailed examination and investigation [11]. Following these preparation steps, the crista structure must be scanned at a resolution high enough to reveal its details[2] (Fig. 1(b)).

In existing literature, various mitochondria segmentation methods employing different modalities have been proposed [12–27]. These methods have led to the development of tools for segmenting mitochondria and other subcellular structures. *MitoSegNet*[3] specializes in segmenting 2D fluorescence microscopy images, offering precise delineation of boundaries. *Empanada*[4] is designed for 2D and 3D electron microscopy datasets, enabling accurate segmentation within volumetric datasets. *Mitometer*,[5] a MATLAB-based tool, is capable of both segmentation and tracking in fluorescence microscopy from time-lapse images, providing valuable insights into the dynamic behavior of mitochondria. For CryoET, another mitochondria segmentation tool[6] that utilizes deep learning for detailed mitochondrial segmentation has been developed. Additionally, some other tools have been also developed for the segmentation of subcellular structures. *Micro-sam*[7] provides semi-automatic segmentation of organelles through deep learning. *ASEM*[8] focuses on segmenting subcellular structures in FIB-SEM datasets. These tools generally utilize deep learning approaches and CNNs and require ground truth data for training purposes.

MitoSeg is a tool developed for mitochondria detection and segmentation based on the algorithm proposed in [9], which is designed to work on datasets exhibiting clear cristae structures. This method enables the segmentation of mitochondria from the Electron Microscopy Tomography (EMT) images using preprocessed specimens mentioned above by leveraging the general physical characteristics of mitochondria without the need for a training phase. For MitoSeg to produce results, a high-resolution intracellular image dataset composed of a set of slices and the corresponding metadata (e.g., slice range and pixel size) is required.

[3] https://github.com/MitoSegNet/MitoS-segmentation-tool
[4] https://empanada.readthedocs.io/en/latest/
[5] https://github.com/aelefebv/Mitometer/
[6] https://github.com/sanketx/mitochondria_segmentation
[7] https://github.com/computational-cell-analytics/micro-sam
[8] https://github.com/kirchhausenlab/incasem

2. Software description

MitoSeg is a command line utility that works with EMT images. It reads through a set of EMT images and produces 2D image and 3D mesh outputs in which the detected mitochondria boundaries are highlighted (Fig. 4). It is developed in C++ and uses the following libraries to operate:

- **OpenCV4:** OpenCV handles the fundamental image processing tasks.
- **Boost:** The Boost library handles string manipulation and command line options.
- **yaml-cpp:** MitoSeg is developed with pre-tuned internal segmentation settings, but it is also designed to allow the users to override these settings via external sources without recompiling MitoSeg. The YAML file format is chosen for its simple syntax among many existing standard formats. The yaml-cpp library provides easy-to-use programming capabilities for loading the user-defined segmentation settings from these external files.

The libraries above are used for building the main CLI utility. Additionally, a graphical user interface has been developed with Python's Tkinter to provide a more user-friendly experience (Fig. 2).

2.1. Software architecture

The software runs in three separate phases, each containing multiple substeps, as illustrated in Fig. 3. The following sections summarize each phase.

2.1.1. Phase 1 - preprocessing, ridge detection, energy mapping, curve fitting

This phase handles the preprocessing of provided dataset images and generates intermediate data required by the actual segmentation process. Since EMT of mitochondria can be a set of reconstructed images obtained from a preprocessed specimen, it may contain unsharp borders, have a low-contrast intensity distribution, and some artifacts involving extreme high and low-intensity levels. The first step of the preprocessing defines a region of interest automatically (if the user does not provide it) by cropping the image from the borders that do not contain useful information. This is achieved by removing borders in which the sum of squared Laplacian of pixel values are less than a threshold. In the second step, an auto-contrast adjustment method [9] is employed in order to remove the extremity in the image histogram and normalize intensity values into 0 - 255. Then, the input images are subsampled to 2 nm per pixel in the third step to facilitate the

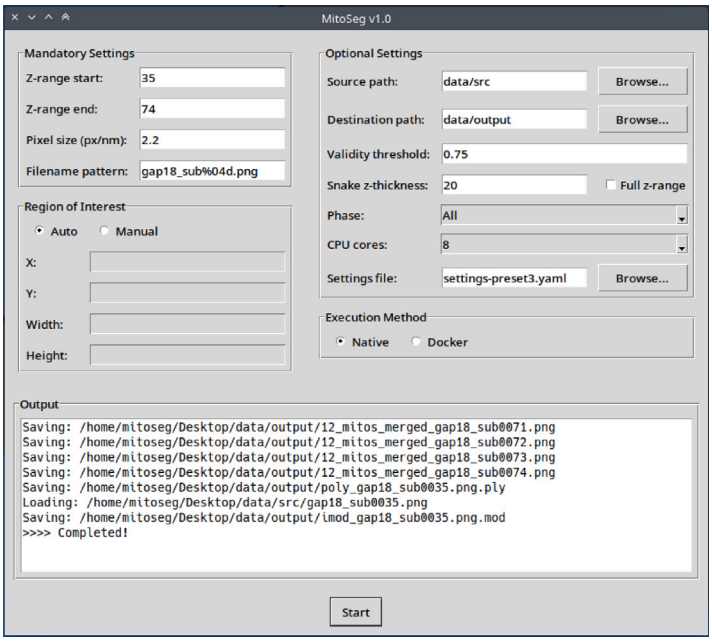

Fig. 2. MitoSeg Graphical User Interface.

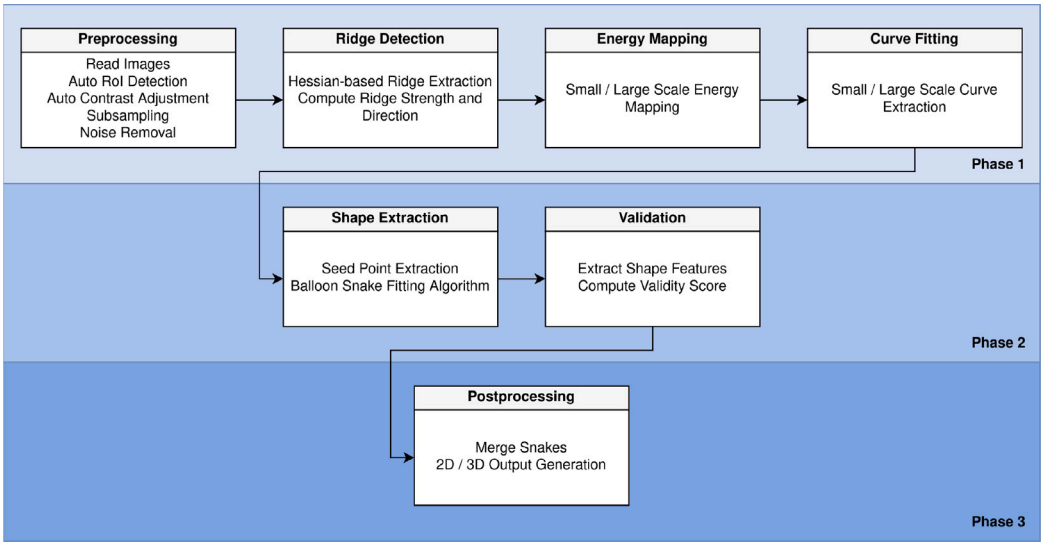

Fig. 3. Flowchart of the algorithm.

parameter tuning. In the last step of the preprocessing, bilateral and Gaussian filtering are applied to input images to emphasize membrane structures while eliminating unwanted noise as depicted in Fig. 4(a).

The preprocessed images are then used in a Hessian matrix-based ridge detection process [9] to locate membrane-like structures (Fig. 4(b)). Note that membranes can be elongated, such as in the periphery of the mitochondrion or relatively short curvy structures (e.g., cristae) as shown in Fig. 1(b). In order to distinguish between the two, an energy mapping process [9] is utilized on large and small scales individually, which calculates the total strength of ridges sharing the same direction, thus providing valuable information on membranes such as curvature, strength, and orientation (Fig. 6(b)–6(c)). Then, a curve fitting process that uses a parabolic arc model [9,28] is employed to extract small and large-scale curve segments as illustrated in Figs. 4(c) and 6(a). Experiments conducted in [9,28] show that extracted curve segments in different scales are useful for locating the boundary and internal structures of mitochondria.

2.1.2. Phase 2 - shape extraction, validation

In this phase, extracted curve segments are used to segment mitochondrial regions. First, seed points are located near curve segments by implementing a modified Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [9,29]. Then, seed points are fed to a pseudo-3D balloon snake [9], which is a variant of an active contour model. This model extracts potential closed regions bounded by the mitochondrial membranes.

However, it is also possible to erroneously segment a region in a different organelle (e.g., somewhere in the endoplasmic reticulum) due to the initialization of the model outside of mitochondria. To overcome this problem, a validation step is performed to filter out incorrect segmentations. A validator function checks several properties of the candidate segmentation, such as the average energy of the boundary and internal region, area of the segmented region, discontinuity, curvature, and signature of the contour. It should be noted that those criteria have been formed by the general physical properties of mitochondria

appearing in tomograms as discussed in [9]. All segmentations identified as valid by the validator function are conveyed to the next phase (Fig. 4(d)).

2.1.3. Phase 3 - postprocessing

In this phase, valid segmented regions are merged if they overlap as illustrated in Fig. 4(e). Finally, boundary points on segmented and merged regions are utilized to generate a 3D mesh as a final product packed in IMOD and PLY file formats so that they can be easily visualized and processed for further applications as shown in Fig. 4(g)–4(h).

The algorithm outlined here is a simplified explanation. A more detailed description of the algorithm, along with further discussion and various examples, can be found in [9].

2.2. Software functionalities

MitoSeg provides several functionalities that enhance the proposed tool's overall performance and usability, such as command line arguments, additional algorithm settings, multi-threaded execution, and a Docker environment for running MitoSeg on all major OS families. The details of each functionality are described below.

2.2.1. Mandatory and optional command line arguments

MitoSeg offers the users various command line options to alter the segmentation process. These options are presented to the users in two forms: mandatory and optional arguments. Below is an example of how to use MitoSeg from the command line:

```
mitoseg ––src /home/user/data ––dst /home/user/output \
        ––psize 2.4 ––zrange 100 200 \
        ––pattern dataset%04d.tif
```

The mandatory command line arguments are listed below:

- **pattern:** Filename pattern for each slice that can be iterated with C-style formatting[9] (e.g., `--pattern mito%03d.tif` can expand into a range of files from mito000.tif to mito999.tif).
- **psize:** Pixel size in nm/px (e.g., `--psize 2.1`).
- **zrange:** Range of slice numbers to be processed (e.g., `--zrange 40 80`).

Following is the list of available command line options:

- **src:** Path of source images (e.g., `--src /home/user/Desktop/data`). Images belonging to the same dataset in this directory must follow the same naming pattern (e.g., slice0001.bmp to slice0200.bmp or mito1.tif to mito100.tif). Source images will be looked for in the current working directory if not specified.
- **dst:** Path of the directory where the intermediate and final output files are stored (e.g., `--dst /home/user/Desktop/output`). If not specified, outputs will be stored in the current working directory.
- **roi:** The region of interest in the provided slices is specified as (*left*, *top*, *width* and *height*), where *left* and *top* represent the x- and y-coordinates of the top-left corner of the rectangular region, respectively, and *width* and *height* denote the horizontal and vertical extent of the region, respectively (e.g., `--roi 100 150 600 800`). If not specified, RoI will be calculated automatically. Manually specifying the RoI can improve performance and help produce better results by focusing on a specific area in a high-resolution image set.
- **phase:** By default, MitoSeg runs the previously explained phases in succession. This option allows for individual running of only a specified phase (e.g., `--phase 2`).
- **valid:** Specifies the threshold of the validator function between 0 and 1 (e.g., `--valid 0.85`). If not specified, it is set to 0.75, which is recommended in [9]. In general, higher values cause increased precision and decreased recall.
- **thick:** Sets the snake thickness (e.g., `--thick 30`). It can be set to a value between 5 and 500 (inclusive) or to "**full**" (i.e., `--thick full`) to use all slices specified by the `zrange` option. It is set to 20 by default.
- **cores:** Number of CPU cores utilized simultaneously for parallel processing (e.g., `--cores 8`). It is set to 1 by default.
- **settings-file:** Path to a YAML file to load custom settings instead of using the predefined ones. If not specified, default settings will be used. (e.g., `--settings-file /home/user/Desktop/settings.yaml`)

2.2.2. Custom algorithm settings

In addition to the command line arguments and options listed above, MitoSeg requires additional settings[10] (e.g., threshold values and iteration amounts) for fine-tuning the algorithm. MitoSeg ships with the alternative settings discussed in [9] as three separate setting files. It allows researchers to use the supplied settings or create additional custom settings files by modifying the provided ones and then using them via the `settings-file` option.

2.2.3. Phase selection

As explained previously, MitoSeg runs in three phases, each focusing on a different segmentation stage. Each phase generates its own intermediate output files, which are utilized by consequent phases. If desired, only a specific phase can be executed to experiment with different settings and observe intermediate results before moving on to the next phase.

2.2.4. Multithreaded execution

Most modern computers are equipped with multiple CPU cores; instead of using only a single core, MitoSeg can employ multiple threads to parallelize the segmentation operations during the first and second phases, thereby enhancing overall time performance. In the first phase, since each slice can be processed independently, the parallelization is realized on a slice-by-slice basis. In the second phase, multiple snake outputs can be extracted independently; therefore, the parallelization is performed on a snake-by-snake basis.

2.2.5. 3D mesh outputs for external tools

In the third phase, in addition to the final boundary images, MitoSeg also exports the generated outputs as .ply and .mod files, which can be displayed using 3D mesh viewers and editors (e.g., MeshLab[11], IMOD[12]).

2.2.6. MitoSeg as a Docker application

MitoSeg has been primarily developed and tested on GNU/Linux-based operating systems. For researchers using other operating systems, a suitable environment for MitoSeg can be set up using Docker[13]. For this purpose, a Dockerfile for building MitoSeg, along with execution scripts that manage runtime options and input/output files, are provided along with the MitoSeg source code. The provided execution scripts `docker-mitoseg.sh` (for Linux/Mac) and `docker-mitoseg.cmd` (for Microsoft Windows) include explanatory comments to guide users in modifying the scripts according to their specific requirements.

[9] https://www.geeksforgeeks.org/format-specifiers-in-c/

[10] https://github.com/fstasel/mitoseg/wiki/Setting-file-description

[11] https://www.meshlab.net/

[12] https://bio3d.colorado.edu/imod/

[13] https://www.docker.com/

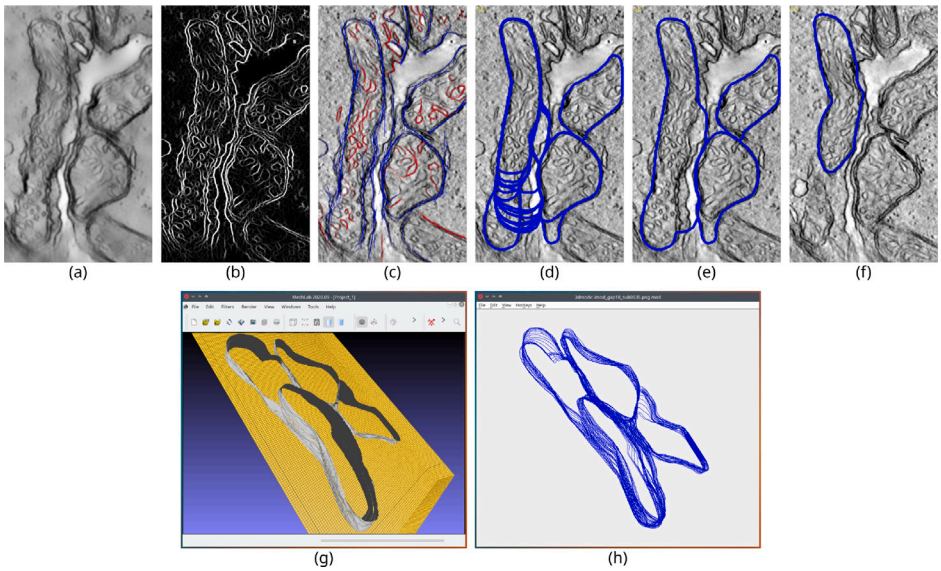

Fig. 4. Outputs. (a) Smoothed image, (b) ridge image, (c) extracted curves, (d) validated boundaries, (e) merged boundaries from slice 35, (f) merged boundaries from slice 55, (g) final output (.ply file), (h) final output (.mod file).

Table 1
Memory usage and total duration for test execution.

	Elapsed time	Peak memory usage
Phase 1 Only	10.61 s	246.2 MB
Phase 2 Only	66.55 s	907.3 MB
Phase 3 Only	4.06 s	683.6 MB
All Phases	76.35 s	907.3 MB

3. Illustrative examples

For demonstration purposes, MitoSeg was used to identify mitochondria in the tomographically reconstructed form of *gap18* dataset (accession number: 8747) [11,30] from Cell Centered Database[14]. Detection and segmentation results for different datasets are presented and discussed in [9].

The tests were conducted on a desktop PC running a 64-bit Ubuntu 22.04 GNU/Linux system with a 6-cores (12 threads) Intel Core i7-8700 CPU and 16 GB RAM. MitoSeg was executed on slices 35 through 74 (total: 40) of the aforementioned dataset with the `psize` option set to 2.2nm/px, while the rest of the options and settings were kept at their default values. Each image in the dataset is a 350 × 600 PNG image, with the total size of the selected slices being 7.0 MB.

The tests were repeated multiple times to observe the elapsed time and memory usage for each phase individually (with the use of `phase` option) and for all phases altogether. The results of these tests are presented in Table 1.

Since the value of the model z-thickness parameter was kept at its default of 20, the final output contains two layers of extracted mitochondria boundaries: the first is from slices 35 to 54, and the second is from slices 55 to 74. Fig. 4 shows intermediate outputs and the extracted boundaries from sample slices together with the produced .ply and .mod outputs as viewed in MeshLab and IMOD, respectively.

The effects of core utilization on execution time have also been tested. For this demonstration, the total duration of each phase has been recorded for the increasing number of CPU cores utilized. The time performance of MitoSeg for the *gap18* dataset (using the same parameters as the previous experiment) is presented in Fig. 5(a). It is observed that there is a significant decrease in the execution times for up to 6 cores in the first and second phases in which parallelization operations are performed. Note that these results highly depend on the number of mitochondria and the size of the dataset provided as input. It achieved a 5.1x speed-up in terms of overall execution time by utilizing all CPU cores compared to single-core execution, as illustrated in Fig. 5(b).

The algorithm implemented in MitoSeg has been evaluated on various TEM datasets with resolutions ranging from 1.1 to 2.4 nm using diverse parameter settings. It achieved an average segmentation accuracy with Dice coefficients reaching up to 0.87 and a median symmetric boundary error (MSBE) as low as 14 nm. The algorithm performance in the presence of noise was also assessed using a phantom image, showing that it can effectively segment mitochondrial boundaries with a signal-to-noise ratio (SNR) of down to 0.8, corresponding to −0.97 dB. Additional details can be found in Appendix A in [9].

4. Impact

In the modern medical field, there is growing interest in understanding the connection between neurodegenerative diseases and mitochondrial structure. Several mitochondrial characteristics, including the thickness of the inner and outer membranes, the structure and number of cristae, the crista junctions, and the size of contact sites, have been measured and suggested to influence mitochondrial function [11]. In this context, MitoSeg plays a crucial role in distinguishing mitochondrial regions from non-mitochondrial ones, serving as a foundation for methods aimed at studying the internal structure of mitochondria since it is able to process non-condensed mitochondrial images. Additionally, the curve-fitting technique employed by MitoSeg can be considered to assist in analyzing crista structures [9]. The extracted small- and large-scale curves resemble a rough segmentation of the boundary and crista, respectively, as depicted in Fig. 6(a). Figs. 6(b) and 6(c) display the energy mapping images, with dashed lines indicating the major direction within each block.

MitoSeg is developed as a fully automated, user-friendly tool to address the challenge above. Its features, listed in the software functionalities section, make it highly flexible and increase MitoSeg's usability. The source code is openly available for researchers, including biologists and computer scientists, who wish to use or modify the methods implemented in MitoSeg.

MitoSeg is particularly important in generating datasets containing isolated mitochondrial regions. Mitochondrial images often contain

[14] https://library.ucsd.edu/dc/object/bb81936790

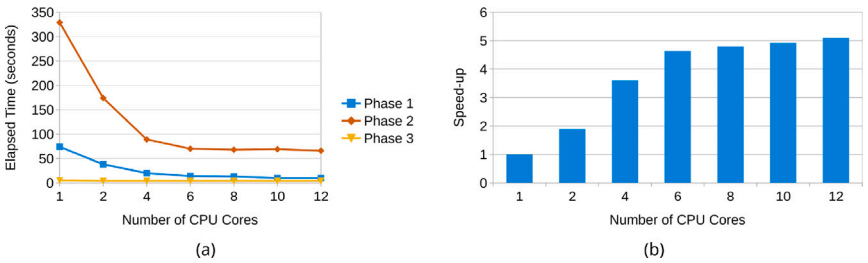

Fig. 5. Number of utilized CPU cores vs (a) time performance, (b) speed-up.

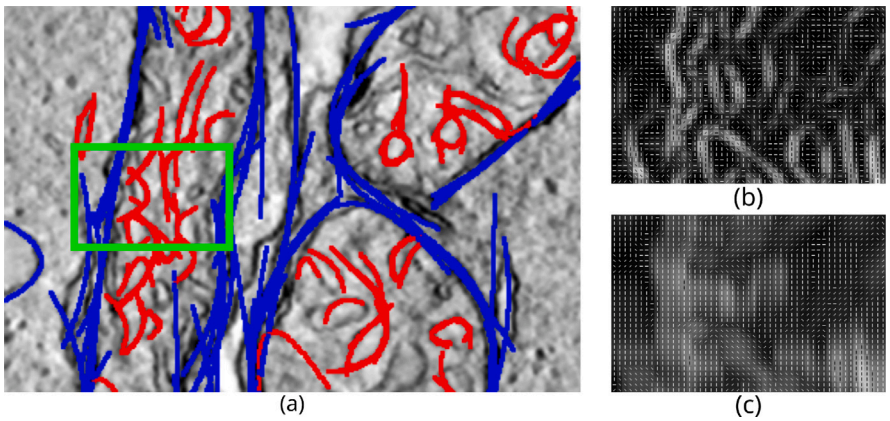

Fig. 6. (a) Large-scale (blue) and small-scale (red) curves extracted from a specified region in the 35th slice of the *gap18* dataset; (b) high, and (c) low energy mapping images for the highlighted green region.

other cellular structures, like the endoplasmic reticulum, in addition to mitochondria. By eliminating these structures, MitoSeg aids in creating new datasets that can be used in approaches requiring training, such as CNNs, for the segmentation of mitochondrial internal structures and exploring their links to diseases.

Since MitoSeg uses features from the general physical structure of mitochondria in its segmentation method, the approach can be helpful in developing new methods for the segmentation of images obtained with new modalities as well as TEM/SBF-SEM imaging techniques without the need for a training phase (and therefore a training dataset). It is also possible to adapt MitoSeg to work with mitochondria images obtained via different preparation techniques (e.g., condensed mitochondria images) by fine-tuning the parameters in the setting file.

The software integrates the careful implementation of sophisticated methods. The algorithm that is used by MitoSeg and presented in [9] has inspired various studies to date, including membrane segmentation [31] and segmentation using CNNs [26,32]. Furthermore, MitoSeg has the potential to support future research in the identification and segmentation of other intracellular structures.

5. Conclusions

In this paper, we have presented MitoSeg, a utility designed for detecting and segmenting mitochondria boundaries in EMT images. MitoSeg integrates preprocessing, ridge detection, energy mapping, curve fitting, shape extraction, validation, and postprocessing into three distinct phases. Through these steps, our evaluations demonstrate that MitoSeg is capable of accurately detecting mitochondria, even when applied to unprocessed raw datasets that suffer from low contrast, oversampling, or noise.

MitoSeg also provides users with several options that enable the segmentation process to be fine-tuned. In addition to generating 2D output images, it can produce 3D mesh representations. The utility supports multicore systems, thereby reducing execution time, and is accessible across major operating systems through a Docker environment.

Tomographic images produced by TEM are typically characterized by relatively low thickness along the z-axis compared to the x and y axes, making pseudo-3D segmentation approaches suitable for TEM images. However, such approaches can introduce discontinuities between independently processed layers, as observed in Figs. 4(g) and 4(h). To address this issue, the segmentation process can be refined by enhancing the balloon snake model to connect segmented mitochondria contours on adjacent layers, followed by re-executing the fitting algorithm. For other imaging modalities, such as SBF-SEM and cryo-EM, fully 3D segmentation models can be directly employed.

Future enhancements of MitoSeg may include adapting the model to GPU parallelization, given the suitability of the balloon snake model for such optimizations.

CRediT authorship contribution statement

Faris Serdar Taşel: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Conceptualization. **Efe Çiftci:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to express their gratitude to Erkan U. Mumcuoglu, Reza Z. Hassanpour, and Guy Perkins for their support in algorithm conceptualization, clarification of biological background and significance, and for providing the datasets.

References

[1] Payne T, Burgess T, Bradley S, Roscoe S, Sassani M, Dunning MJ, et al. Multimodal assessment of mitochondrial function in parkinson's disease. Brain 2024;147(1):267–80. http://dx.doi.org/10.1093/brain/awad364.

[2] Pradeepkiran JA, Baig J, Seman A, Reddy PH. Mitochondria in aging and Alzheimer's disease: focus on mitophagy. Neurosci 2024;30(4):440–57. http://dx.doi.org/10.1177/10738584221139761.

[3] Reiss AB, Gulkarov S, Jacob B, Srivastava A, Pinkhasov A, Gomolin IH, et al. Mitochondria in Alzheimer's disease pathogenesis. Life 2024;14(2):196. http://dx.doi.org/10.3390/life14020196.

[4] Borsche M, Pereira SL, Klein C, Grünewald A. Mitochondria and Parkinson's disease: clinical, molecular, and translational aspects. J Parkinson' s Dis 2021;11(1):45–60. http://dx.doi.org/10.3233/JPD-201981.

[5] Malpartida AB, Williamson M, Narendra DP, Wade-Martins R, Ryan BJ. Mitochondrial dysfunction and mitophagy in Parkinson's disease: from mechanism to therapy. Trends Biochem Sci 2021;46(4):329–43. http://dx.doi.org/10.1016/j.tibs.2020.11.007.

[6] Monzio Compagnoni G, Di Fonzo A, Corti S, Comi GP, Bresolin N, Masliah E. The role of mitochondria in neurodegenerative diseases: the lesson from Alzheimer's disease and Parkinson's disease. Mol Neurobiol 2020;57:2959–80. http://dx.doi.org/10.1007/s12035-020-01926-1.

[7] Wang W, Zhao F, Ma X, Perry G, Zhu X. Mitochondria dysfunction in the pathogenesis of Alzheimer's disease: Recent advances. Mol Neurodegener 2020;15:1–22. http://dx.doi.org/10.1186/s13024-020-00376-6.

[8] Xu CS, Hayworth KJ, Lu Z, Grob P, Hassan AM, García-Cerdán JG, et al. Enhanced FIB-SEM systems for large-volume 3D imaging. Elife 2017;6:e25916. http://dx.doi.org/10.7554/eLife.25916.

[9] Tasel SF, Mumcuoglu EU, Hassanpour RZ, Perkins G. A validated active contour method driven by parabolic arc model for detection and segmentation of mitochondria. J Struct Biol 2016;194(3):253–71. http://dx.doi.org/10.1016/j.jsb.2016.03.002.

[10] Titze B, Genoud C. Volume scanning electron microscopy for imaging biological ultrastructure. Biol Cell 2016;108(11):307–23. http://dx.doi.org/10.1111/boc.201600024.

[11] Perkins GA, Ellisman MH, Fox DA. Three-dimensional analysis of mouse rod and cone mitochondrial cristae architecture: bioenergetic and functional implications. Mol Vis 2003;9:60–73.

[12] Nešić N, Heiligenstein X, Zopf L, Blüml V, Keuenhof KS, Wagner M, et al. Automated segmentation of cell organelles in volume electron microscopy using deep learning. Microsc Res Tech 2024;87(8):1718–32. http://dx.doi.org/10.1002/jemt.24548.

[13] Luo N, Sun R, Pan Y, Zhang T, Wu F. Electron microscopy images as set of fragments for mitochondrial segmentation. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 38, (4):2024, p. 3981–9. http://dx.doi.org/10.1609/aaai.v38i4.28191.

[14] Conrad R, Narayan K. Instance segmentation of mitochondria in electron microscopy images with a generalist deep learning model trained on a diverse dataset. Cell Syst 2023;14(1):58–71. http://dx.doi.org/10.1016/j.cels.2022.12.006.

[15] Pan Y, Luo N, Sun R, Meng M, Zhang T, Xiong Z, et al. Adaptive template transformer for mitochondria segmentation in electron microscopy images. In: Proceedings of the IEEE/CVF international conference on computer vision. 2023, p. 21474–84. http://dx.doi.org/10.1109/iccv51070.2023.01963.

[16] Wu G-H, Smith-Geater C, Galaz-Montoya JG, Gu Y, Gupte SR, Aviner R, et al. Cryoet reveals organelle phenotypes in huntington disease patient iPSC-derived and mouse primary neurons. Nat Commun 2023;14(1):692. http://dx.doi.org/10.1038/s41467-023-36096-w.

[17] Franco-Barranco D, Pastor-Tronch J, González-Marfil A, Muñoz-Barrutia A, Arganda-Carreras I. Deep learning based domain adaptation for mitochondria segmentation on EM volumes. Comput Methods Programs Biomed 2022;222:106949. http://dx.doi.org/10.1016/j.cmpb.2022.106949.

[18] Franco-Barranco D, Muñoz-Barrutia A, Arganda-Carreras I. Stable deep neural network architectures for mitochondria segmentation on electron microscopy volumes. Neuroinformatics 2022;20(2):437–50. http://dx.doi.org/10.1007/s12021-021-09556-1.

[19] Gallusser B, Maltese G, Di Caprio G, Vadakkan TJ, Sanyal A, Somerville E, et al. Deep neural network automated segmentation of cellular structures in volume electron microscopy. J Cell Biol 2022;222(2):e202208005. http://dx.doi.org/10.1083/jcb.202208005.

[20] Li M, Chen C, Liu X, Huang W, Zhang Y, Xiong Z. Advanced deep networks for 3D mitochondria instance segmentation. In: 2022 IEEE 19th international symposium on biomedical imaging. IEEE; 2022, p. 1–5. http://dx.doi.org/10.1109/ISBI52829.2022.9761477.

[21] Somani A, Ahmed Sekh A, Opstad IS, Birna Birgisdottir Å, Myrmel T, Singh Ahluwalia B, et al. Virtual labeling of mitochondria in living cells using correlative imaging and physics-guided deep learning. Biomed Opt Express 2022;13(10):5495–516. http://dx.doi.org/10.1364/BOE.464177.

[22] Lefebvre AE, Ma D, Kessenbrock K, Lawson DA, Digman MA. Automated segmentation and tracking of mitochondria in live-cell time-lapse images. Nature Methods 2021;18(9):1091–102. http://dx.doi.org/10.1038/s41592-021-01234-z.

[23] Fischer CA, Besora-Casals L, Rolland SG, Haeussler S, Singh K, Duchen M, et al. MitoSegNet: easy-to-use deep learning segmentation for analyzing mitochondrial morphology. Iscience 2020;23(10). http://dx.doi.org/10.1016/j.isci.2020.101601.

[24] Mekuč MŽ, Bohak C, Hudoklin S, Kim BH, Kim MY, Marolt M, et al. Automatic segmentation of mitochondria and endolysosomes in volumetric electron microscopy data. Comput Biol Med 2020;119:103693. http://dx.doi.org/10.1016/j.compbiomed.2020.103693.

[25] Peng J, Yi J, Yuan Z. Unsupervised mitochondria segmentation in EM images via domain adaptive multi-task learning. IEEE J Sel Top Signal Process 2020;14(6):1199–209. http://dx.doi.org/10.1109/JSTSP.2020.3005317.

[26] Xiao C, Chen X, Li W, Li L, Wang L, Xie Q, et al. Automatic mitochondria segmentation for EM data using a 3D supervised convolutional network. Front Neuroanat 2018;12:92. http://dx.doi.org/10.3389/fnana.2018.00092.

[27] Lucchi A, Smith K, Achanta R, Knott G, Fua P. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. IEEE Trans Med Imaging 2011;31(2):474–86. http://dx.doi.org/10.1109/TMI.2011.2171705.

[28] Tasel SF, Hassanpour R, Mumcuoglu EU, Perkins GC, Martone M. Automatic detection of mitochondria from electron microscope tomography images: a curve fitting approach. In: Medical imaging 2014: image processing, Vol. 9034, SPIE; 2014, p. 1116–23. http://dx.doi.org/10.1117/12.2043517.

[29] Ester M, Kriegel H-P, Sander J, Xu X, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, Vol. 96, (34):1996, p. 226–31.

[30] Fox D, Perkins GA. Microscopy product ID: 8747. In cell centered database. 2017, http://dx.doi.org/10.6075/J0SF2W0K, URL https://library.ucsd.edu/dc/object/bb81936790.

[31] Siggel M, Jensen RK, Maurer VJ, Mahamid J, Kosinski J. ColabSeg: An interactive tool for editing, processing, and visualizing membrane segmentations from cryo-ET data. J Struct Biol 2024;216(2):108067. http://dx.doi.org/10.1016/j.jsb.2024.108067.

[32] Xiao C, Li W, Chen X, Han H, Xie Q. An effective fully deep convolutional neural networks for mitochondria segmentation based on ATUM-SEM. In: Medical imaging 2018: image processing, Vol. 10574, SPIE; 2018, p. 65–70. http://dx.doi.org/10.1117/12.2293291.