International Journal of Computer Vision
https://doi.org/10.1007/s11263-023-01818-6

S.I. : COMPUTER VISION APPROACH FOR ANIMAL TRACKING AND MODELING

# Skeletonizing *Caenorhabditis elegans* Based on U-Net Architectures Trained with a Multi-worm Low-Resolution Synthetic Dataset

**Pablo E. Layana Castro[1] · Antonio García Garví[1] · Francisco Navarro Moya[1] · Antonio-José Sánchez-Salmerón[1]**

Received: 24 March 2022 / Accepted: 15 May 2023
© The Author(s) 2023

**Abstract**
Skeletonization algorithms are used as basic methods to solve tracking problems, pose estimation, or predict animal group behavior. Traditional skeletonization techniques, based on image processing algorithms, are very sensitive to the shapes of the connected components in the initial segmented image, especially when these are low-resolution images. Currently, neural networks are an alternative providing more robust results in the presence of image-based noise. However, training a deep neural network requires a very large and balanced dataset, which is sometimes too expensive or impossible to obtain. This work proposes a new training method based on a custom-generated dataset with a synthetic image simulator. This training method was applied to different U-Net neural networks architectures to solve the problem of skeletonization using low-resolution images of multiple *Caenorhabditis elegans* contained in Petri dishes measuring 55 mm in diameter. These U-Net models had only been trained and validated with a synthetic image; however, they were successfully tested with a dataset of real images. All the U-Net models presented a good generalization of the real dataset, endorsing the proposed learning method, and also gave good skeletonization results in the presence of image-based noise. The best U-Net model presented a significant improvement of 3.32% with respect to previous work using traditional image processing techniques.

**Keywords** Synthetic dataset · Low-resolution image · U-net · Skeletonizing · End points · *Caenorhabditis elegans*

## 1 Introduction

Extracting the central line or skeleton of a worm from images is not an easy task, and much less so in low-resolution images when there are aggregations between worms or the worms aggregate with plate noise. We use the word "noise" to refer to dark objects or segmentations, residues, stains or worm shapes, which are not actually worms. All these cases cause classical skeletonization algorithms to fail, leading to erroneous results. As demonstrated in Layana Castro et al. (2020), compared to classical skeletonization techniques, improved techniques can better locate and identify worms in the aforementioned cases, facilitating the automation of monitoring tasks, posture recognition, behavioral studies, etc.

Over the years, many applications have been developed for automatic monitoring and inspection of *C. elegans* using classical image processing techniques. Many of these applications solve this problem by identifying the central line or skeleton of the *C. elegans*. Skeleton identification basically reduces the shape of the worms without losing information about their posture. In order to identify the worms within images, some methods have been implemented to extract characteristics of the worm from the skeleton. The best-known characteristics include endpoints Tsibidis and Tavernarakis (2007), smoothness Rizvandi et al. (2008a, b), length Wöhlby et al. (2012), previous segmentation Uhlmann and Unser (2015); Winter et al. (2016), or a combination of these Layana Castro et al. (2021). Other applications for extracting worm skeletons use neural network techniques Chen et al. (2020); Li et al. (2020b); Hebert et al. (2021).

Methods that use neural networks are becoming more reliable and precise, helping many professionals and researchers to achieve their goals in many fields of science. However, a problem implicit to using these techniques is having a dataset large enough to train and validate the model, in addition to

✉ Antonio-José Sánchez-Salmerón
asanchez@isa.upv.es

[1] Universitat Politècnica de Valéncia, Instituto de Automática e Informática Industrial, Camino de Vera S/n, Edificio 8 G Acceso D, 46022 Valencia, Valencia, Spain

Published online: 08 June 2023

testing the results. Moreover, creating a labeled dataset is usually a time-consuming manual task. In this context, some neural network applications have proven that the use of synthetic data can solve this problem either partially (mixing real and synthetic data) or completely (purely synthetic data).

Generally, in *C. elegans* assays, these applications are performed using high-resolution images where there is only one worm or very few worms per plate. The advantage is that better processing results are obtained due to the number of pixels and that overlaps or aggregations between worms are avoided. A recent work Hebert et al. (2021) proposes a simulator to generate purely synthetic high-resolution images using reverse skeletonization, this technique employs small rectangular image patches to generate worm images.

We present a new convolutional neural network skeletonization method trained using purely synthetic low-resolution images. The convolutional network architecture used is the U-Net Plebani et al. (2022). We take advantage of this U-shaped architecture with an encoder and decoder to produce encoded images of worm skeletons from low-resolution grayscale images. Instead of generating new synthetic frames of individual poses with rollings and self-intersections as in Hebert et al. (2021), our simulator generates new synthetic frames of multi-worm poses with intersection behaviors and parallel contacts. The results show a significant improvement of 3.32% compared to a previous work Layana Castro et al. (2020) which improved classical skeletonization methods.

Highlights:

- A *C. elegans* skeletonization method is proposed based on U-Net type neural networks with low-resolution images and noise.
- A new method for generating low-resolution synthetic images is proposed to easily generate a custom-labeled dataset for different *C. elegans* behaviors.
- A neural network has been trained with a low-resolution synthetic image and successfully tested in the domain of real images.
- Different U-Net architectures have been compared with an algorithm based on traditional image processing techniques.

## 2 Related Work

In this section, we review the state of the art of various works related to different network architectures and neural network techniques applied to *C. elegans*.

### 2.1 *Caenorhabditis elegans* and Neural Networks

*Caenorhabditis elegans* is one of the most widely studied organisms and has acquired great importance in the field of biology Biron and Haspel (2015). Its genome has been annotated in great detail, and research shows that many human diseases have homologues in the genome of this nematode Olsen and Gill (2017), making it an attractive animal model for the study of human pathologies. The advantages offered by this organism with respect to others include its short life cycle (around 21 days), short reproductive period, small size (around 1 mm long), and feeding based on bacterial strains such as *Escherichia coli*; all of which facilitate its large-scale culture Conn (2017).

In the past, *C. elegans* assays were monitored manually but nowadays many researchers choose more automatic technologies, thus reducing both processing times and the fatigue of technicians, who would otherwise spend hours looking through the microscope daily. This is where computer-vision applications have a great advantage over these manual practices. Due to the flexible body and the different poses that *C. elegans* can adopt, many automatic applications use skeletonization techniques to solve problems related to healthspan (Hahm et al., 2015; Le et al., 2020; Di Rosa et al., 2020), lifespan (Jung et al., 2014; Kumar et al., 2019; Puchalt et al., 2020), tracking (Javer et al., 2018; Koopman et al., 2020), behavior monitoring (Yu et al., 2014; Pitt et al., 2019), etc. These techniques are very useful in microscopic images without noise, but low-resolution and noisy images present a challenge difficult to overcome. Other automatic methods use neural networks (NN), which are more robust against these problems.

Neural networks have different topologies and these can perform tasks of classification, segmentation, detection, and so on. All these architectures can automatically extract features, with which *C. elegans* applications are developed, such as skeletonization Hebert et al. (2021), extreme segmentation Mane et al. (2020), and others (Wiehman and de Villiers, 2016; Wang et al., 2019, 2020; Yu et al., 2021). Furthermore, networks that have an encoder and a decoder have proven capable of solving more complex tasks such as posture classification Javer et al. (2019), skeleton definition in microscope images Chen et al. (2020) or patch acquisition to resolve aggregation Mais et al. (2020).

### 2.2 U-Nets

The U-Net is a neural network with an encoder and a decoder. Since it first appeared, the U-Net Ronneberger et al. (2015) has been widely used not only in medical applications, for which it was first introduced, but also in the segmentation of animals (Han et al., 2019; Padubidri et al., 2021), objects (Zhao et al., 2019; Wiles & Zisserman, 2019), etc. Various authors have taken this network architecture as a reference and have modified it to achieve greater convergence in training. These modifications consist of adding, removing, or replacing convolutional layers with others, thereby increas-

ing or decreasing the size of the network. However, making the network deeper (increasing the number of convolution layers) or increasing the width of the network does not always result in a better prediction, it all depends on our dataset.

SmaAt-UNet Trebing et al. (2021) proposes reducing the size of the base U-Net Ronneberger et al. (2015) by adding spatial-channel attention, and shows that it can achieve similar precision values, this leads to reducing the inference time or the resources needed during the exploitation phase. On the other hand, UMF U-Net Plebani et al. (2022) modifies the U-Net standard by adding BatchNorm and dropOut layers and shows that careful choice of hyperparameters and other training configurations play a very important role in network development. As well as these cases, there are other more recent modifications (Alexandre, 2019; Moradi et al., 2019; Tang et al., 2019; Tschandl et al., 2019; Li et al., 2020a; Liu et al., 2020; Qamar et al., 2020; Baheti et al., 2020; McManigle et al., 2020; Huang et al., 2020; Cao et al., 2020; Isensee et al., 2021) that present significant improvements in the precision of their respective datasets compared to the standard U-Net.

When working with neural networks, it is problematic to obtain a well-labeled and balanced dataset. This can be a very expensive or even an impossible task; therefore, to alleviate this problem, different data augmentation methods have been developed, as well as simulators for generating synthetic images.

### 2.3 Data Augmentation and Synthetic Images

The use of synthetic data can help attain network convergence, avoid overfitting, and improve data generalization. On the one hand, the mixture of synthetic data with real data has been shown to help improve the training of neural networks (Pashevich et la., 2019; Doshi, 2019; Bargsten & Schlaefer, 2020; Dewi et al., 2021), even in applications with *C. elegans* García Garví et al. (2021). In general, these types of techniques convert synthetic images to the domain of real images, in order to achieve similar distributions. This domain change is achieved thanks to architectures such as GAN Li et al. (2020b), encoders, and decoders Chen et al. (2021).

On the other hand, applications that only use synthetic images also provide good results in the domain of real images, (Schraml, 2019; Hinterstoisser et al., 2019; Mayershofer et al., 2021). The use of simulators to create purely synthetic images pave the way to being able to create larger and more variable datasets, thus being able to generate cases or events that occur infrequently in real images. The simulation of *C. elegans* in low-resolution images is a challenge that, to our knowledge, has not been addressed before. Accordingly, Hebert et al. (2021) uses a high-resolution synthetic image simulator of a single worm to train a neural network.

In this work, we aim to generate low-resolution synthetic images to train a segmentation neural network of worm skeletons from 20 to 40 points. Our method obtains outstanding results, outperforming previous skeletonization work that improved classical image processing techniques.

## 3 Methods

### 3.1 Strain and Culture Conditions of *C. elegans*

The *C. elegans* used were wild type (N2) and CB1370, *daf*-2(e1370) young adults, obtained from worm eggs incubated at 20°C in 55 mm diameter NGM plates at the Caenorhabditis Genetics Center (CGC), University of Minnesota. *Escherichia coli* strain OP50 was used as food. To prevent reproduction, FUdR (0.2 mM) was used, and to prevent contamination by fungi, fungizone (1µg/mL) was added and the plates were closed with a lid. The standard method Stiernagle (2006) was followed to remove those plates with contamination. Plates with 10, 15, 30, 60, and 90 nematodes were cultured to obtain greater variability and analyze different types of behavior (aggregation of two or more *C. elegans*, aggregations with plate noise and occlusions).

### 3.2 Real-Image Acquisition Method

Images of complete 55 mm diameter Petri dishes were captured. Image acquisition was performed by a laboratory operator at a temperature of 20°C using the hardware and software described in Puchalt et al. (2021). To capture the images, the laboratory operator removed the plates from the incubator and placed them in the capture system (Fig. 1) where the system proceeded to capture a sequence of images of $1944 \times 1944$ pixels at a frequency of 1Hz. The *Escherichia coli (E. coli)* OP50 strain was seeded in the center of the plate to prevent *C. elegans* from moving out of the field of view, either by climbing up the edges of the plates or by positioning themselves near the plate edges. Those plates with condensation on the cover were withdrawn from the image acquisition process.

The abovementioned system Puchalt et al. (2021) was developed with open hardware and software, using a Raspberry Pi v1.3 RGB camera, OmniVision OV5647 with resolution of $2592 \times 1944$ pixels, and pixel size of $1.4 \times 1.4\ \mu$ m, field of view of $53.50\circ \times 41.41\circ$, with 1/4" optical size and 2.9 focal ratio, a lighting system based on a 7" Raspberry Pi screen with a resolution of $800 \times 480$ at 60 fps, 24-bit RGB color and as processing unit a Raspberry Pi 3, (Fig. 1). The mounting process and image capture are detailed in Puchalt et al. (2021). The lighting technique used was active backlighting. This technique has been shown effective for low-resolution *C. elegans* applications for both the afore-

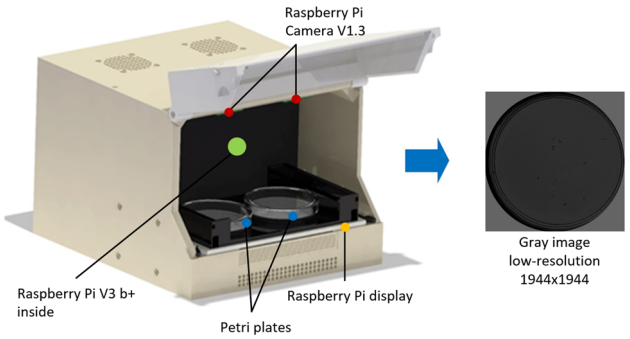


**Fig. 1** Image capture system. Location of the Petri dishes, as well as the other parts of the capture system Puchalt et al. (2021)

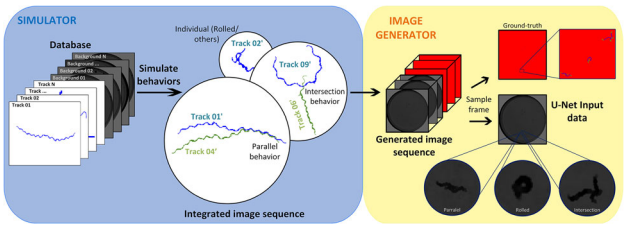


**Fig. 2** Conceptual outline of the synthetic image generation process

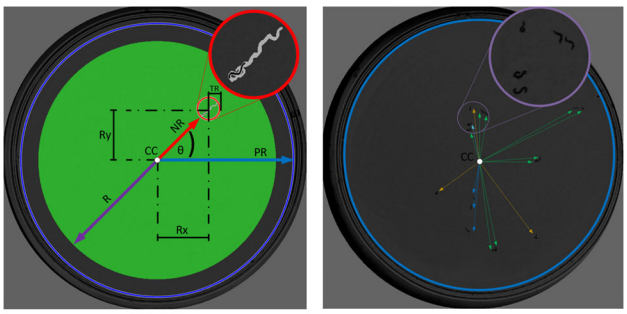


**Fig. 3** Development of synthetic images. **a** Random generation of the track of a worm. **b** Synthetic image with 16 worms on the plate

mentioned Puchalt et al. (2019) and Puchalt et al. (2022) capture systems. Active backlighting consists of reducing the variability of the captured images by keeping gray scales more constant. This made it possible to differentiate the background of *C. elegans* easily in all images. To capture the real image sequences, our object of interest (Petri dish with worms) was placed between the illumination system and the camera, as described in Puchalt et al. (2021). With this configuration the nematodes have a maximum size of 40 × 4 pixels and a minimum of 20 × 3 pixels. In reality, the worms measure 1 mm. Working with low resolutions may complicate the problem in some cases, but it has advantages in terms of computational and memory efficiency. This resolution is sufficient to automate assays such as lifespan, healthspan, etc.

### 3.3 Image Simulation Method

A simulator has been designed capable of generating new sequences both of aggregation behaviors between worms (parallel behaviors and intersections) and of augmenting individual behaviors (free motion, coiling) with geometric transformations. Figure 2 shows a conceptual outline of the synthetic image generation process. The simulator starts from a database containing manually labeled real *C. elegans* paths and images of Petri dishes without worms. The labels contain the location (X,Y) of each of the points of the skeleton, as well as the color and width of each point of the skeleton. The process to simulate a sequence consists of selecting K paths randomly, applying rotation and translation transformations to them, and combining them to obtain the desired behaviors, thus obtaining an integrated sequence. To generate the parallel aggregation behavior, one path is randomly selected and a new path is generated by modifying its color and XY position. The XY position of the first worm is at NR, while the second worm is at NR + W1. W1 is the smallest width of the worm body (1 or 2 pixels). The simulator generates two types of parallel behaviors, in the first, both worms navigate in the same direction, while in the second one of them navigates in the opposite direction. To achieve the second case the path of one of the two worms is rotated 180°, i.e., the worm of the rotated path starts at the end of the path and navigates in parallel with the other worm approximately in the middle of the path. To generate the intersection aggregation behavior, two paths are randomly selected, the intersection point will be a random point on the skeleton of each worm from a random pose of each path. Each path has 30 poses and each worm can have between 20 and 40 skeleton points. We have observed in the real dataset that aggregation behaviors (intersection and parallel) are accompanied by speed changes and pauses, so we randomly added these interactions within the simulator. Pauses are simulated by repeating poses for two instants of time. Speed changes are obtained by skipping a pose in the path.

The trajectory simulation consisted of applying a rotation angle ($\theta$) to all the skeletons of a worm sequence and moving that sequence to a random X-Y point inside the plate. The values of N, $\theta$ and CC the centroid of the Petri dishes (pre-process info) were needed to calculate the new X-Y position of the worm trajectory (Eqs. 2 and 3), Fig. 3a and b. The angle $\theta$ and NR were randomly generated between $0 - 2\pi$ and $0 - R$, respectively. The value of R was obtained from the difference between the plate radius (P) and the diameter of the trajectory found in the pre-process info file (2T), Eq. 1.

$$R = P - 2T \tag{1}$$

$$N_x = CC_x + N\cos\theta \tag{2}$$

$$N_y = CC_y + N\sin\theta \tag{3}$$

To rotate the worm skeletons, a random angle ($\alpha$) between $0-2\pi$ was generated. The rotation operation was performed by multiplying each skeleton pixel (Eq. 5) by a rotation matrix (Eq. 4):

$$M_a = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \tag{4}$$

$$P_i = \begin{bmatrix} P_x \\ P_y \end{bmatrix} \tag{5}$$

Then, using the integrated sequence, the image sequence is generated by drawing the paths on an empty Petri dish image. The paths are drawn by inserting circles of diameter equal to the width value of the skeleton point stored in the database into each of the pixels of the skeletons. To color the circle, the value of the skeleton point is used and also averaged with the background. Finally, a blur filter ($3 \times 3$) is applied to the images. This filter was essential to bring the synthetic image domain closer to the real image domain. In addition, it favored convergence in the training of the networks. In this step, in addition to the gray image sequence, ground-truth masks are also generated. This simulator, which has been designed to obtain sequences of behaviors, allows an image to be selected from the generated sequence as input to the network. The details of the code implementation have been added to Appendix 1.

### 3.4 Classical Skeletonization Method

Classical skeletonization techniques have proven easy-to-implement for extracting shapes and predicting worm behaviors, which are problematic when worms coil or aggregate with each other, or with plate noise. When this happens, part of the skeleton is absent or displaced, and this is because these skeletonization techniques reduce the segmentation pixels until they achieve the central line. In these cases, skeleton prediction errors occur, as shown in Fig. 4. If the aggregation occurs in a large part of the worm body it can lead to a large error.

### 3.5 Skeletonization Method Using Improved Skeleton

This method involves obtaining improved skeletons (Fig. 5a–c) from the width and length characteristics. These characteristics are obtained when the worms are free and not coiled during a previous preprocessing Layana Castro et al. (2020). The advantage of this technique, unlike classical skeletonization techniques, is that it can separate connected or coiled worms through new skeletons. In general, other applications cancel tracks where aggregation between worms, aggregation with noise, and coiling occur; however, the improved technique is very useful in these cases, recognizing skeletons (poses) and predicting behaviors. Layana Castro et al. (2020) showing that an improved skeleton together with worm-specific features such as color and temporal image features can solve problems of aggregation between worms or with noise in image sequences.

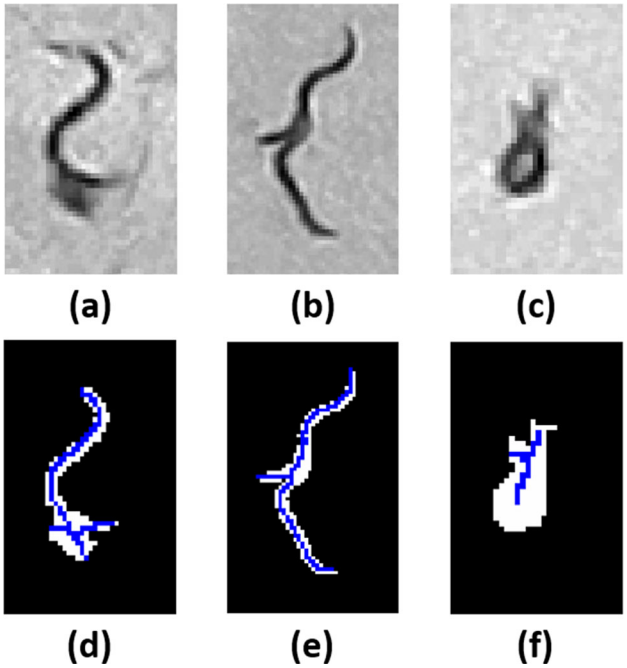


**Fig. 4** Classical skeletonization of problematic cases. **a** Grayscale image of worm aggregated with noise. **b** Grayscale image of two worms aggregated at one end and part of the body. **c** Gray image of worm coiled upon itself. **d**, **e**, **f** Result of classical skeletonization of images **a**, **b**, **c**, respectively. The white pixels show the segmentation using a threshold of 35, while the blue pixels show the result of skeletonizing that segmentation

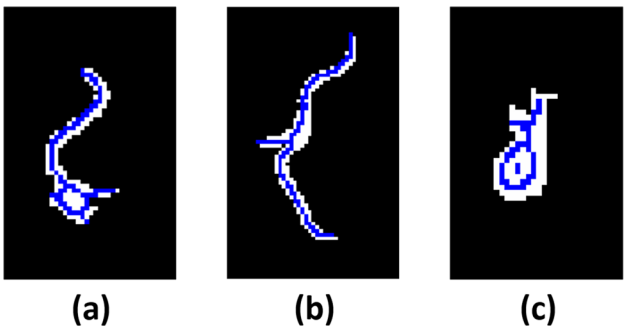


**Fig. 5** Skeletonization with enhanced algorithm (ISA) Layana Castro et al. (2020). **a**, **b**, **c** Skeletonization result with an improved algorithm (ISA) of the gray images from Fig. 4a, b, c, respectively. The white pixels show the segmentation using a threshold of 35, while the blue pixels show the result of skeletonizing that segmentation

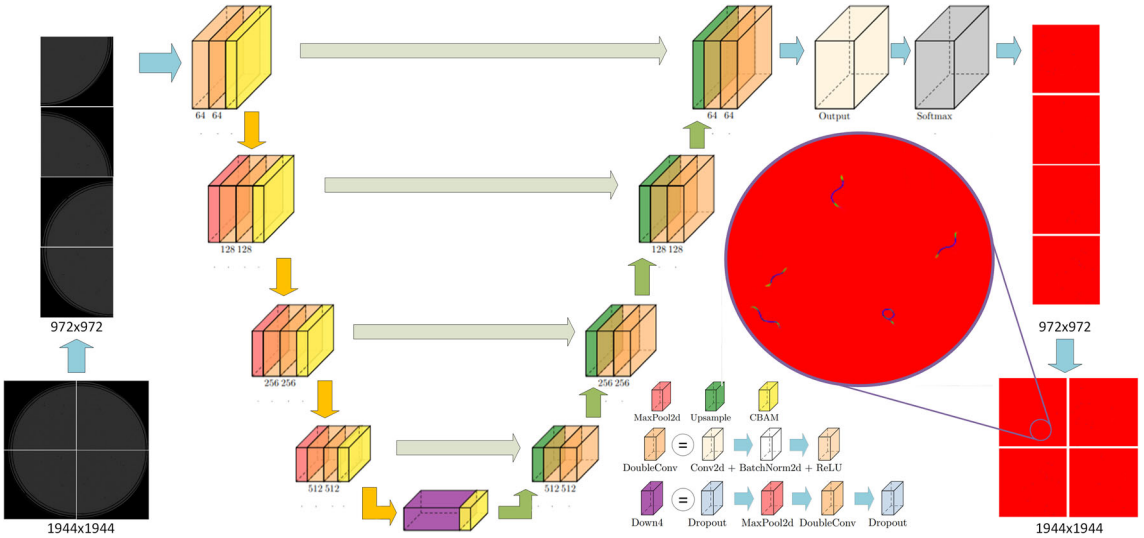


**Fig. 6** Image pipeline through U-Net architecture. The blocks of the U-Net architecture were made using the PlotNeuralNet tool Iqbal (2018). The image is divided into 4 parts, each part enters the network and the result is reassembled to form a single image

### 3.6 Proposed Skeletonization Method

The model used for the segmentation of worm skeletons was the convolutional neural network U-Net. Different U-Net architectures were analysed to compare their performance. Comparison was made of the models U-Net standard Ronneberger et al. (2015), Alexandre's U-Net Alexandre (2019), SmaAt-UNet, U-Net with DSC, U-Net with CBAM, U-Net with DSC, CBAM Trebing et al. (2021), UMF U-Net Plebani et al. (2022) and all showed good results.

Figure 6 shows all the blocks used with the different architectures. For the standard U-Net Ronneberger et al. (2015), the Doubleconv block does not have the white BatchNorm2d block and the yellow CBAM blocks and the purple Down4 block does not have the Dropout blocks. Alexandre's U-Net Alexandre (2019) is the same as the standard U-Net but includes the BatchNorm2d block inside the Doubleconv block. UMF U-Net Plebani et al. (2022) is the same as Alexandre's U-Net but on the purple Down4 block it does have the Dropout blocks. SmaAt-UNet Trebing et al. (2021) on the other hand has the spatial-channel attention blocks, it also has no dropout blocks in the purple Down4 block, and the models with DSC instead of the Doubleconv blocks have depthwise-separable convolutions blocks.

These models predict three classes: background, worm ends, and worm body. The background class (red pixels in Fig. 6) are all those pixels that do not correspond to the worm skeletons, such as the plate edge and interior of the Petri dish, dark spots, residues inside the dish, etc. The worm-ends class (green pixels in Fig. 6) includes those pixels corresponding to the head and tail of the worm skeleton. These vary between 5 and 10 pixels at each end, they are generally lighter pixels (higher grayscale intensity). And finally, the worm-body class (blue pixels in Fig. 6) are pixels in the center of the skeleton and are darker than the worm-end pixels (less grayscale intensity).

Given the dimensions of the input image and the limitations of the hardware to train and validate the architecture, both the input image and the ground-truth image were divided into 4 equal parts. To reconstruct the prediction image, each output prediction part of the network was joined in the same order as the input image, as shown in Figs. 6 and 7.

## 4 Evaluation Method

To evaluate all the U-Net models, two datasets were used, a synthetic dataset that was used in the training and validation stages of the networks, and a dataset of real images to test the results in the cases of coiling, aggregation between worms, and aggregation with noise (Fig. 8).

To evaluate the real dataset, the Jaccard index, also known as intersection over union (IoU), and euclidean distance were used. The IoU coefficient measures the accuracy of a prediction with respect to a ground-truth Koul et al. (2019). And as its name indicates, it is obtained by dividing the total area of the intersection by the union of these areas, Eq. 6. On the other hand, the euclidean distance measures the average error in pixels of a prediction with respect to a ground-truth, Eq. 7

$$IoU = \frac{\sum P_{w1} \bigcap P_{w2}}{\sum P_{w1} \bigcup P_{w2}} \tag{6}$$

$$E.D. = \sum_{i=1}^{nw} \frac{\sqrt{(X1_i - X2_i)^2 + (Y1_i - Y2_i)^2}}{nw} \tag{7}$$

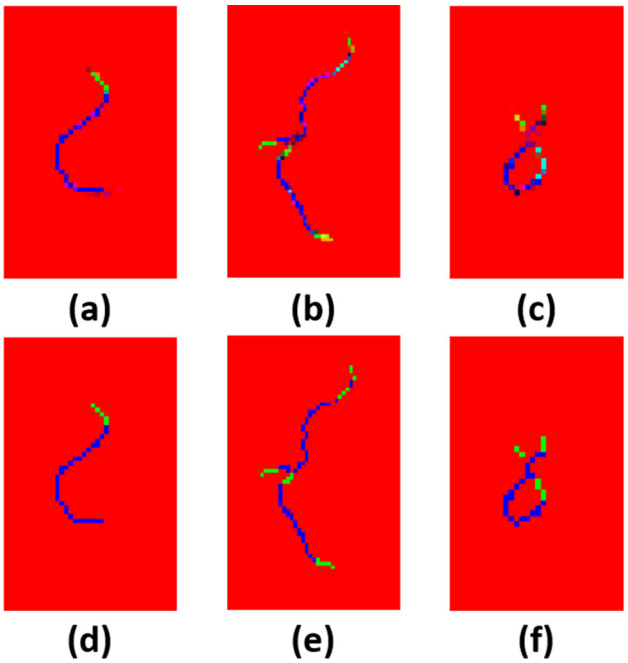


**Fig. 7** Coding of the output image from the network. **a**, **b**, **c** Resulting skeletons using the UMF U-Net Plebani et al. (2022), network from the gray images in Fig. 4a, b, c respectively. **d**, **e**, **f** Pixel encoding using the maximum value of the RGB channels. Red pixels are background pixels, blue pixels are worm body pixels, and green pixels are worm-end pixels. The results obtained with the rest of the models are similar to these

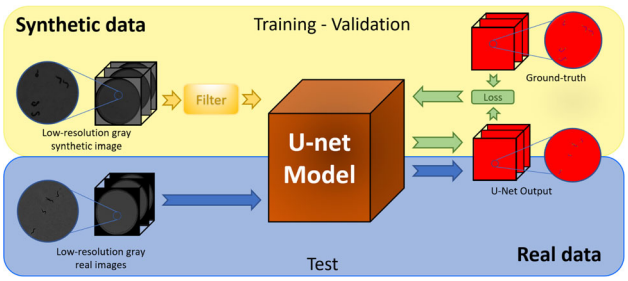


**Fig. 8** Synthetic and real dataset pipeline. The synthetic dataset was used to train and validate the U-Net neural network. The trained network was used to test the real image domain

The Precision and Recall metrics were used to evaluate the results of the detection experiment, and the MOTA metric was used to evaluate the results of tracking experiment. To obtain Precision (Eq. 8) and Recall (Eq. 9) metrics, three parameters were used: TP (true positives), FP (false positives), FN (false negative). On the other hand, to obtain MOTA metric (Eq. 10), the FN, FP, IDS and GT parameters were used. GT was the total number of worms in the aggregation, IDS value was increased by 1 when the body of a predicted worm overlapped more with another worm than with its respective GT. For the overlap, the IoU value and a threshold of 0.5 were considered.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDS_t}{\sum_t GT_t} \tag{10}$$

The synthetic dataset labels (ground-truth) were obtained automatically from the simulator, while the real dataset labels (ground-truth) were obtained by manually labeling worm skeletons. This task was performed using a pixel labeling application. The pixels of each worm skeleton were selected one by one until the skeleton was complete.

The predictions are not always exact for all the pixels, usually, one or more pixels are displaced with respect to the real label (ground-truth), thus obtaining low measurement errors and incorrect skeleton indicators. When the worm is 3 pixels in diameter, the skeleton pixel is the center pixel, but if the worm has an even number of pixels, it is impossible to select the center pixel, which may result in false errors between manual labeling and pixel predictions of the skeleton. To solve this problem and obtain a better measurement of results, we decided to use the worm body to obtain a more significant IoU value that would better reflect the prediction of the skeleton. To recover the shape and body of the worm, a dilation operation was performed on all the pixels of the skeleton with a disk of radius 2 (approximate diameter of the worm). This operation was performed for all manual labels. IoU and Euclidean distance metrics have been calculated for the following classes: worm ends, worm body and worm (fusion of body and ends).

## 5 Experiments and Results

### 5.1 Method Comparison

In this experiment, different U-Net architectures were compared to find the most accurate one for our case. In addition, it was compared with the results of a method based on traditional computer vision techniques.

As previously mentioned, a synthetic dataset was used to train and validate the networks and a real dataset to test all the results. For the synthetic dataset, 400 sequences of 30 images were simulated, giving a total of 12000 images, 70% was used to train the network (8400) and the other 30% was used for the validation stage (3600). Each image of the sequence had 16 worms per plate, in which different cases of aggregation were simulated. The hardware used for training and validation of the different networks was a Gigabyte Technology Z390 AORUS PRO machine, Intel(R) Core (TM) i9-9900KF CPU @ 3.60GHz x16 with 32GB of RAM, and NVIDIA GeForce RTX 2080 Ti graphics card with 4352 Cuda cores,

**Table 1** Synthetic dataset loss and IoU results

| Model | loss | Avg. worm-ends class | | Avg. worm-body class | | Total average worm class | |
|---|---|---|---|---|---|---|---|
| | | IoU | E.D | IoU | E.D | IoU ± IC 95% | E.D. ± IC 95% |
| U-Net | 6.47E-06 | 0.8883 | 0.1875 | 0.9315 | 0.0747 | 0.9322±2.40E-04 | 0.1067±5.12E-04 |
| U-Net A | 3.05E-04 | 0.8179 | 0.1262 | 0.8573 | 0.1269 | 0.8597±1.24E-03 | 0.1270±1.06E-03 |
| UMF U-Net | **1.54E-06** | 0.9317 | 0.0019 | **0.9371** | **0.0006** | **0.9361±9.30E-04** | **0.0009±6.99E-05** |
| SmaAt DS | 2.16E-05 | 0.9153 | 0.0241 | 0.9297 | 0.0184 | 0.9287±1.09E-03 | 0.0199±3.68E-04 |
| SmaAt DS AT | 1.60E-05 | 0.9191 | 0.0226 | 0.9305 | 0.0112 | 0.9295±1.03E-03 | 0.0141±3.38E-04 |
| SmaAt DS AT 4C | 1.23E-04 | 0.8120 | 0.1263 | 0.8625 | 0.1073 | 0.8615±1.07E-03 | 0.1123±7.15E-04 |
| SmaAt AT | 3.06E-06 | **0.9324** | **0.0017** | 0.9364 | 0.0011 | 0.9355±9.42E-03 | 0.0012±9.44E-05 |

Both columns show the average for the results of the 3600 evaluation images of the synthetic dataset. The loss function used was CrossEntropyLoss(), the IoU index, and the euclidean distance (E.D.) were the ones described in the evaluation method

**Table 2** Average IoU results of the actual dataset

| Model | N parameters | Avg. aggregation | | Avg. Agg. with noise | | Avg. rolled | | Total average worm | |
|---|---|---|---|---|---|---|---|---|---|
| | | IoU | E.D | IoU | E.D | IoU | E.D | IoU ± CI 95% | E.D. ± CI 95% |
| ISA | | 0.7625 | 0.5659 | 0.6421 | 2.1726 | **0.8122** | **0.5540** | 0.6936±0.0125 | 1.6185±0.1517 |
| U-Net | 17.2576M | **0.7634** | **0.5551** | 0.6510 | 0.9001 | 0.7600 | 0.6678 | 0.6923±0.0134 | 0.8092±0.0466 |
| U-Net A | 17.2664M | 0.6858 | 0.7822 | 0.6980 | 0.6712 | 0.7172 | 0.6089 | 0.6977±0.0105 | 0.6274±0.0277 |
| UMF U-Net | 17.2664M | 0.6992 | 0.6090 | 0.7313 | 0.7259 | 0.7622 | 0.5686 | **0.7279±0.0066** | **0.6097±0.0276** |
| SmaAt DS | 3.9536M | 0.6339 | 0.5737 | 0.7133 | 0.6562 | 0.7481 | 0.5905 | 0.6993±0.0086 | 0.6529±0.0323 |
| SmaAt DS AT | 4.0320M | 0.5759 | 0.5870 | 0.6849 | 0.6505 | 0.7004 | 0.6150 | 0.6613±0.0107 | 0.6642±0.0347 |
| SmaAt DS AT 4C | 3.9986M | 0.5713 | 0.7471 | 0.7193 | 0.7032 | 0.7497 | 0.6053 | 0.6884±0.0125 | 0.6240±0.0194 |
| SmaAt AT | 17.3447M | 0.6767 | 0.8359 | **0.7338** | **0.6459** | 0.7610 | 0.6146 | 0.7240±0.0082 | 0.6476±0.0232 |

Average IoU values of problematic cases using the encoding of the maximum value of RGB channels

Ubuntu 19.04 64bits operating system. The implementation was carried out in a Python version 3.7.5 environment, using the Pytorch 1.18, OpenCV 4.5.4, and SWIG 3.2 libraries. Different U-Net architectures were compared and the training for each of these took about 48 h with the abovementioned hardware. The hyper-parameters used for the training and validation phase were Batch_size = 1, num_workers = 1, maximum epoch = 10. The optimizer used was ADAM with a learning rate = 0.0001, betas = [0.95, 0.999], eps = 1e-8, CrossEntropyLoss() as the loss function, and the ReduceLROnPlateau scheduler with hyper-parameters mode = 'min, and patience = 2. All the U-Net architectures used were trained and evaluated using the same training and validation dataset. After each training, the model with the lowest loss value in the validation phase was selected to evaluate all the results. The average loss resulting from the loss function CrossEntropyLoss() and average IoU values for each model are shown in Table 1.

For the real dataset, 4500 images of Petri dishes were analyzed and those difficult cases were selected in which the worms coiled on themselves, aggregated to each other, or presented noise from the dish, in 90, 157, and 417 images, respectively. In order to obtain all this variability, the images contained 10, 15, 30, 60, and 90 worms.

The output images of all U-Net models were encoded using the maximum RGB channel value. Once the coded images of prediction and ground-truth were obtained, the pixels of the worm body and the worm ends were joined to form a single skeleton, then the shape of the worm was recovered as indicated in the evaluation method and the precision result was obtained using the IoU index. Table 2 shows the total parameters of each model and the average of the results obtained from the evaluation with these models and for all the cases analyzed. Appendix 2 shows the metrics obtained for each of the problematic cases: aggregation between worms (Table 4), aggregation with noise (Table 5) and rolled cases (Table 6).

As shown, the results of the networks were also compared with the values obtained in a previous work (ISA) Layana Castro et al. (2020). The average results showed that for the real dataset UMF U-Net is the best skeletonization method, showing a statistically significant difference with the previous work Layana Castro et al. (2020) Fig. 9. It should be noted that for cases of aggregation with noise it is the best option. Figure 10 compares the previous work with respect to the UMF U-Net architecture using a box plot. Figure 11 shows the results obtained in an image section of the different architectures used. Although the results are similar, Fig. 11d


**Fig. 9** Statistical analyses. **a** Normality test on the difference of methods (ISA-UMF U-Net). The *p*-value obtained was 5.88E-61 less than the significance value of 0.05, thus the null hypothesis was rejected and the alternative hypothesis H1 was accepted (the data did not come from a normal distribution). Once the alternative hypothesis was accepted, the Wilcoxon signed-rank test was used to evaluate both methods. **b** The Wilcoxon signed-rank test table shows the difference between two related samples across positive, negative and tied ranks. **c** The *p*-value obtained with the Wilcoxon rank test was 0.0040 less than the significance value of 0.05, thus concluding there was a statistically significant difference between both models

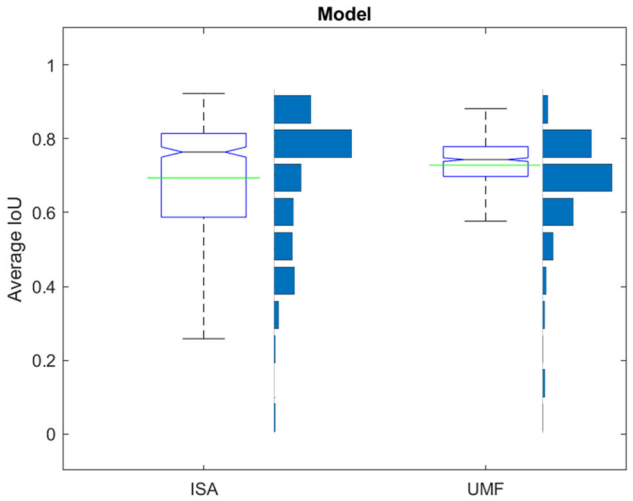


**Fig. 10** Comparison of previous work Layana Castro et al. (2020) with UMF U-Net Plebani et al. (2022). The green line indicates the mean in both graphs and the gray line indicates the median. ISA N = 664, mean = 0.6936, median = 0.7635, standard deviation = 0.1649, variance = 0.0272. UMF U-Net N = 664, mean = 0.7279, median = 0.7430, standard deviation = 0.0871, variance = 0.0076

of the UMF U-Net predicts more connected skeletons than the other architectures.

## 5.2 Real Versus Synthetic Image Training

To demonstrate the need to use a simulator, an experiment was performed comparing the results of training with synthetic images alone versus training with the real dataset available using standard data augmentations. The labeling effort required to generate the starting database for the simulator was also analyzed. For this purpose, we compared the accuracy obtained by the network when training with different numbers of starting worms to generate the simulation. The model trained in all cases was the UMF U-Net. The network

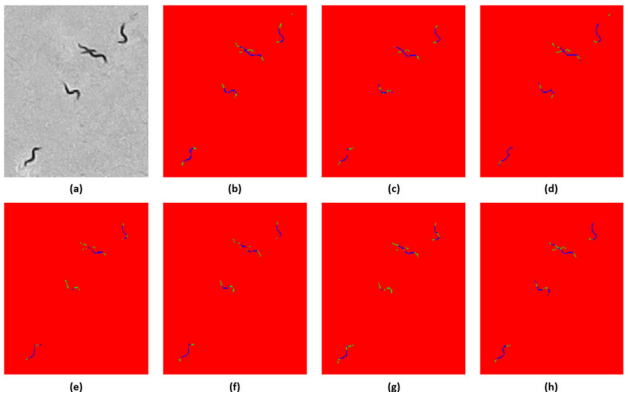


**Fig. 11** Comparison of skeletons obtained with the different U-Net architectures. An image was selected and results were obtained coded for all the different architectures, then the same section was cropped in all the images. **a** Grayscale image. **b** Result with U-Net standard Ronneberger et al. (2015), **c** Result with Alexandre's U-Net Alexandre (2019), **d** Result with UMF U-Net Plebani et al. (2022), **e**, **f**, **g**, **h** Result with SmaAt-UNet Trebing et al. (2021) (SmaAT Ds, SmaATDs At, SmaATDs At 4CBAMs and SmaAT, respectively)

was trained for 20 epochs using the same hyperparameters as in the UNets comparison test. The training with real images only used data augmentation based on affine transformations (rotations (90,180 and 270 degree angles), vertical (1–30 pixels up/down) and horizontal (1–30 pixels left/right) translations, brightness and contrast changes). Table 3 shows the results obtained using the IoU metrics and Euclidean distance between skeleton points broken down by the different cases (aggregation, aggregation with noise and rolled). As can be seen in the results, the training with real images obtains worse results than all the models trained with synthetic images.This result justifies simulator use since training with only 30 base poses already gives good results (IoU = 0.6850), which are much better than those obtained with the training with real image by increasing data (IoU = 0.4265). Regarding the comparison between the different models trained with synthetic images, we find that the higher the number of base poses of the simulator, the better the results obtained. We also observe that the higher the number of base poses, the lower the number of epochs required for model convergence. However, these differences are not highly significant. This means that good results can be obtained with less labeling effort to generate the base poses. In this experiment, we also wanted to analyze the learning of the rolling case. Comparing the accuracy obtained in the trial using 30 base poses (none rolled up) versus the trial using 18810 base poses (3780 rolled up) the improvement is not very significant. The fact that the network can skeletonize rolled poses without having examples in training may be because it learns from the cases of parallel motion and cross aggregation, where it also has to solve the problem of skeletonizing wide areas.

**Table 3** Results of the experiment Real versus synthetic image

| Train | | | | Test | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | Annotated | Best | Avg. aggregation | | Avg. Agg. with noise | | Avg. rolled | | Total average | |
| | worms | poses | epoch | IoU | E. D | IoU | E. D | IoU | E. D | IoU ± CI 95% | E. D. ± CI 95% |
| Real | 627 | 18810 | 11 | 0.4987 | 1.7913 | 0.3965 | 3.5523 | 0.4405 | 2.3572 | 0.4265±0.0094 | 2.9776±0.1403 |
| Synthetic | 1 | 30 | 14 | 0.6358 | 0.6487 | 0.6914 | 0.7795 | 0.7424 | 0.6146 | 0.6850±0.0115 | 0.7267±0.0419 |
| Synthetic | 16 | 480 | 13 | 0.6782 | 0.6581 | 0.7097 | 0.7346 | 0.7264 | 0.6554 | 0.7044±0.0077 | 0.6824±0.0323 |
| Synthetic | 30 | 900 | 9 | **0.6997** | **0.5777** | 0.7267 | 0.7299 | 0.7432 | 0.6387 | 0.7225±0.0082 | 0.6818±0.0332 |
| Synthetic | 627 | 18810 | 5 | 0.6992 | 0.6090 | **0.7313** | **0.7259** | **0.7622** | **0.5686** | **0.7279±0.0066** | **0.6097±0.0276** |

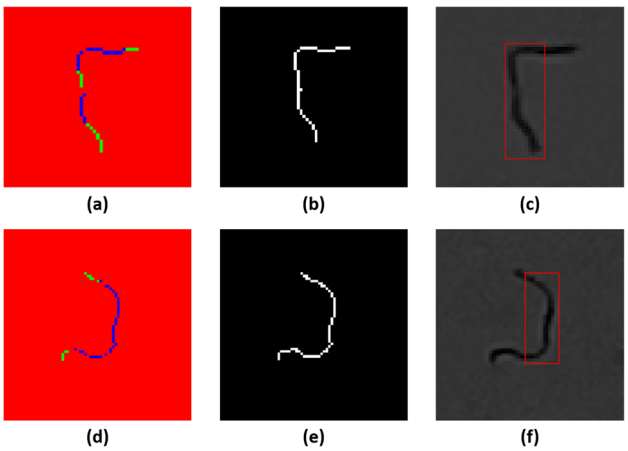


**Fig. 12** Noise-generated connected components detection. **a**, **d** UMF U-Net network output encoding. **b**, **e** Joining method between final pixels of head/tail and body classes. **c**, **f** Detection result

## 5.3 Detection Application

This experiment consists of analyzing the accuracy of a detection method based on the proposed skeletonization method. For this experiment, 120 dataset-real images were used, these images contained different aggregation behaviors between worms and individual behaviors, all of these plates had high noise content. The selected images were passed through the trained UMF U-Net network and were encoded as indicated in Fig. 7 d, e, and f (Fig. 12a, b). Then we proceeded to analyze whether there were skeletons of broken worms, that is, if there was a distance greater than 1 and less than 4 pixels between the final pixels of the head/tail and body classes, if that was the case, both endpoints were joined (Fig. 12b, e). After this, each connected component was analyzed to obtain possible skeleton solutions. To do so, a recursive algorithm used the endpoints and intersections of connected points to go through the connected component and obtain sequences of points between 20 and 40. All these possible solutions were analyzed by an optimizer which evaluated whether they were individual worms or aggregations between worms and obtained the best solution for each case using 2 evaluation criteria: Minimum skeleton length (20 pixels), completeness (the solutions occupy the entire connected component). To detect aggregations, it was considered if the connected component had more than 3 head/tail pixel connected components (green pixels) and if the connected component had more than 30 pixels. The Precision and Recall metrics were used to measure the precision of this experiment. The results were 83.50% and 73.77% respectively. The Precision and Recall metrics are detailed in the evaluation method section. For cases of cross aggregations and individual behaviors, the precision and recall values were very high, close to 1. The precision errors were due to problems with the method of joining body and endpoint stubs. The recall errors were due to noise-generated connected components that complied with worm characteristics, Fig. 12c, f. In order to use the proposed skeletonization method in detection applications, the joining method and noise filtering should be improved.

## 5.4 Tracking Application

This experiment consists of analyzing the accuracy of a tracking method based on the proposed skeletonization method. In addition, the IoU and computational cost results were compared with the WT-ISA method Layana Castro et al. (2021). The WT-ISA tracking method is based on a skeletonization method Layana Castro et al. (2020) and an optimization method. The skeletonization method was designed specifically to solve rolling and aggregation cases. In the aggregation cases, the optimizer obtains the best solution among all possible combinations of endpoints and branches (possible solutions). The proposed tracking method consists of the UMF U-Net skeletonization method and the same optimization method used in WT-ISA. The accuracy evaluation was performed with paths from the real dataset. This dataset was divided between individual behaviors and aggregations. The aggregations dataset contains 72 paths of aggregations between 2, 3 and 4 worms and with plate noise. The result of paths of individual behaviors such as rolling and self-occlusions obtained an IoU and MOTA value close to 1. The results obtained with the aggregations dataset showed

an average IoU value of 0.66 and a MOTA value of 0.70, an identity loss (IDS) of 7%. The IDS is distributed 95% in aggregations between worms (mostly in parallel) and 5% aggregation with noise. In most of these cases, after separating the worms, the identity of each individual was recovered. Regarding the comparison, the WT-ISA (Layana Castro et al. (2021)) method obtained an IoU value of 0.70 and generated 75626 possible solutions for the 72 aggregation paths, while the proposed method obtained an IoU value of 0.66 and generated 10997 possible solutions. Demonstration videos using the proposed skeletonization method were uploaded to github and a GoogleColab notebook to show worm tracking in image sequence. https://github.com/playanaC/Skeletonizing_Unet/blob/main/Demo_videos.ipynb.

## 6 Discussion

Noise on the plate (dark objects, residues, stains, or worm shapes that are not worms) usually results in constant scales of gray across the plate surface, thus it is distinguishable from worm bodies, which have different characteristics. Head and tail pixels are lighter than body pixels. However, these colors are altered by the aggregation of the worm with the noise on the plate, producing connected segmentations and causing the failure of fixed-threshold skeletonization techniques. Notwithstanding, neural networks are capable of learning features that are robust to changes in illumination and intensity. In this scenario, the variability of the dataset is a key piece in network training, thus changes in skeleton intensity, as well as variability in social or individual cases included in the network, will result in better predictions, even surpassing improved skeletonization techniques Layana Castro et al. (2020) as shown in Table 2.

Worm curling cases, however, are more complex to detect, especially in low-resolution images. Depending on the extent to which the worms are coiled, it may be possible to identify skeletons and worm ends or, on the contrary, it may be an almost impossible task for either traditional skeletonization techniques or neural networks. Previous work Layana Castro et al. (2020) showed that information on the width and length of the worm can be used to create an improved skeleton in order to provide a better pose and skeleton of the worm's body. As indicated in Trebing et al. (2021), U-Nets with spatial-channel attention can achieve similar results to other architectures with a greater number of parameters, with which lighter applications or applications with multiple models can be developed.

All networks except the U-Net standard have Batch normalization layers, it is clear that applying batch normalization after each convolutional layer allows data regularization, as well as better convergence, reducing internal covariate change as mentioned in Ioffe and Szegedy (2015). UMF U-Net Plebani et al. (2022), on the other hand, adds another regularization technique (Dropout) at the end of the encoder, obtaining better skeletons with synthetic and real data.

## 7 Conclusions and Future Work

In this work, we have proposed a skeletonization method for low-resolution images of Petri dishes containing *C. elegans* based on U-Net type neural networks. A synthetic dataset of different *C. elegans* behaviors was generated using a low-resolution image simulator to train different U-Net architectures. Good skeletonization results were achieved with all models trained on real images. Finally, the results of networks were compared with a skeletonization method based on traditional image processing techniques, showing that all networks were superior to those in previous work Layana Castro et al. (2020) in terms of aggregations with noise. In future works, the low-resolution synthetic image simulator will be used together with U-Net plus a temporary network to predict worm behaviors and aim to perform tracking by resolving aggregations between worms.

**Acknowledgements** ADM Nutrition, Biopolis S.L. and Archer Daniels Midland supplied the *C. elegans* plates. Some strains were provided by the CGC, which is funded by NIH Office of Research Infrastructure Programs (P40 OD010440). Mrs. Maria-Gabriela Salazar-Secada developed the skeleton annotation application. Mr. Jordi Tortosa-Grau and Mr. Ernesto-Jesús Rico-Guardioa annotated worm skeletons.

**Funding** Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This study was supported by the Plan Nacional de I+D with Project RTI2018-094312-B-I00, FPI Predoctoral contract PRE2019-088214 and by European FEDER funds.

**Data Availability** The dataset with all aggregation experiments can be downloaded from https://active-vision.ai2.upv.es/wp-content/uploads/2021/02/dataset_skeletons.zip. The image capture system, parts, and assembly can be downloaded from https://github.com/JCPuchalt/SiViS.

**Code Availability** Codes and the pretrained model are publicly available at https://github.com/playanaC/Skeletonizing_Unet. Simulator demo is available at https://github.com/playanaC/Skeletonizing_Unet/blob/main/Simulator_image.ipynb. UMF U-Net demo is available at https://github.com/playanaC/Skeletonizing_Unet/blob/main/Demo_test.ipynb. Video demos from tracking is available at https://github.com/playanaC/Skeletonizing_Unet/blob/main/Demo_videos.ipynb.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copy-

right holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

## Appendix A. Other Images

The synthetic image sequence was generated using two functions: Simulator and Create Image (Image generator). The first generated PTS and backgrounds files from preprocessing files (TXT), worm skeleton files (XML), and background images without worms. While the second function used these files to generate the images (Fig. 13).The PTS files contained skeleton point information, as well as color and width values for each skeleton pixel. The synthetic image generator was developed in C++ using the OpenCV tool for image processing and integrated into Python through the SWIG application.

Figure 14 shows different skeleton errors. Figure 14a, d show errors occurred by the presence of noise similar to the worm. Figure 14b, e show errors when worms are highly aggregated, and Fig. 14c, f shows errors when worms are small and coiled upon itself. Figure 15 show full image of Fig. 11, and Fig. 16 show the comparison of previous work Layana Castro et al. (2020) with U-nets architectures using a box plot.

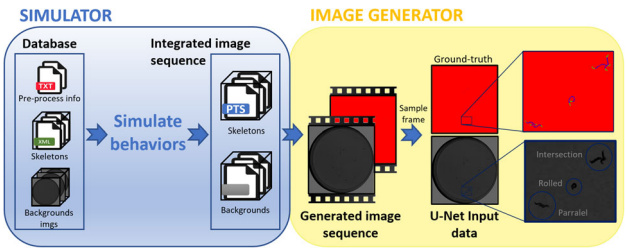


**Fig. 13** Pipeline code

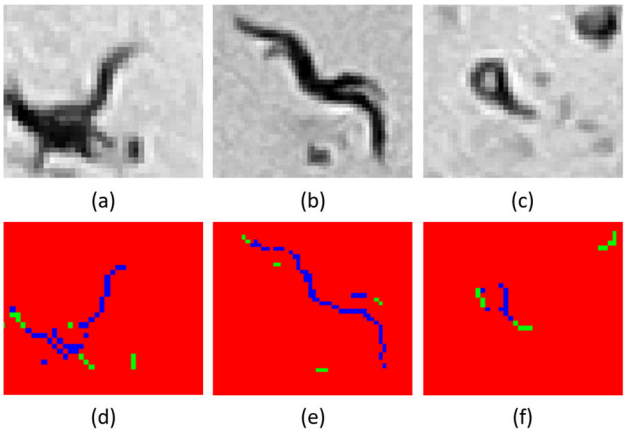


**Fig. 14** Error example of UMF U-Net model. **a** Grayscale image of worm aggregated with noise. **b** Grayscale image of two worms aggregated at one end and part of the body. **c** Gray image of worm coiled upon itself. **d**, **e**, **f** Pixel encoding using the maximum value of the RGB channels. Red pixels are background pixels, blue pixels are worm body pixels, and green pixels are worm-end pixels. The results obtained with the rest of the models are similar to these

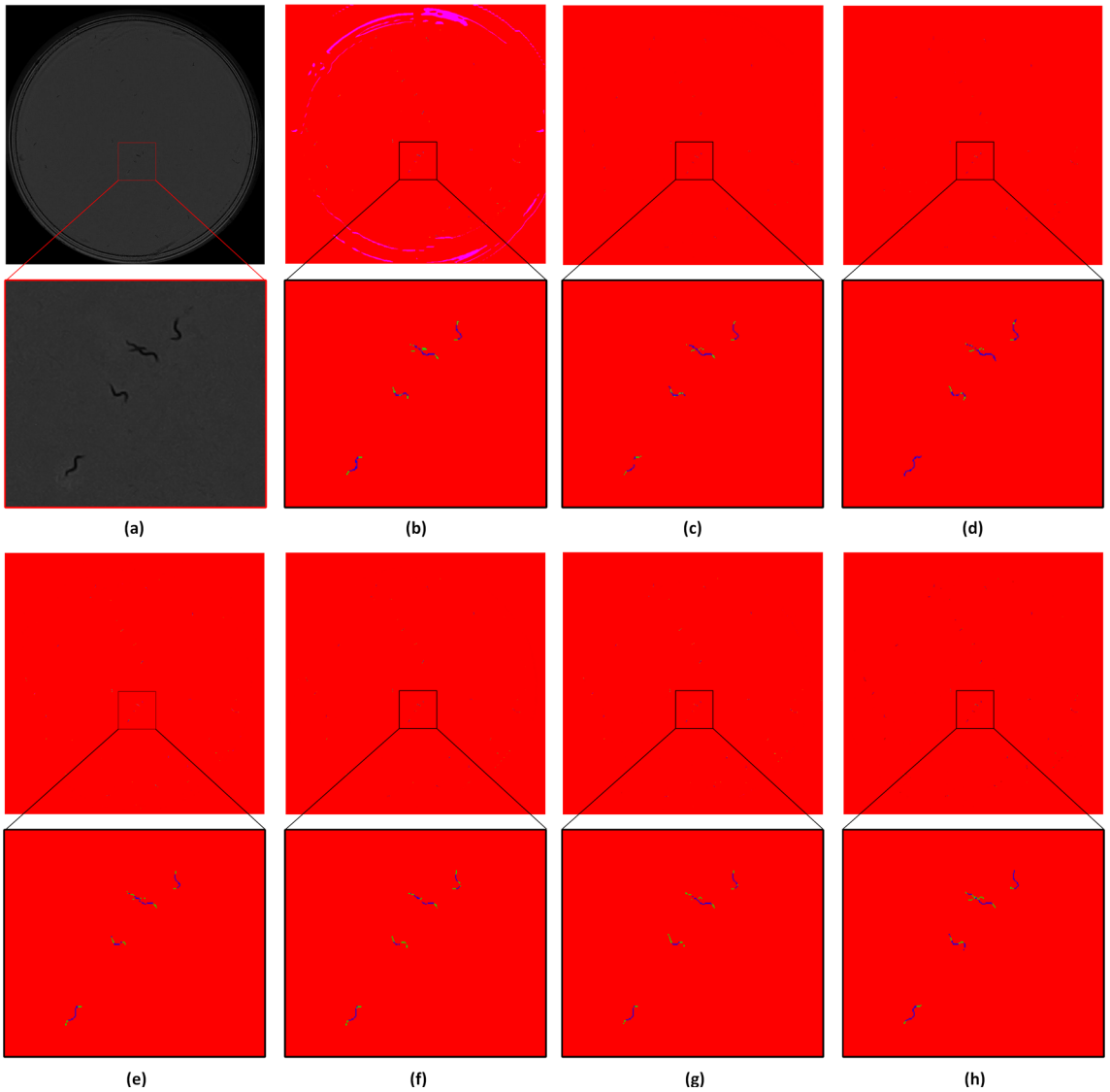


**Fig. 15** Comparison of skeletons obtained with the different U-Net architectures. Full image of Fig. 11. Each image presents a zoom to a certain area. **a** Grayscale image. **b** Result with standard U-Net Ronneberger et al. (2015), **c** Result with Alexandre's U-Net Alexandre (2019), **d** Result with UMF U-Net Plebani et al. (2022), **e**, **f**, **g**, **h** Result with SmaAt U-Net Trebing et al. (2021) (SmaAT Ds, SmaATDs At, SmaATDs At 4CBAMs and SmaAT respectively)

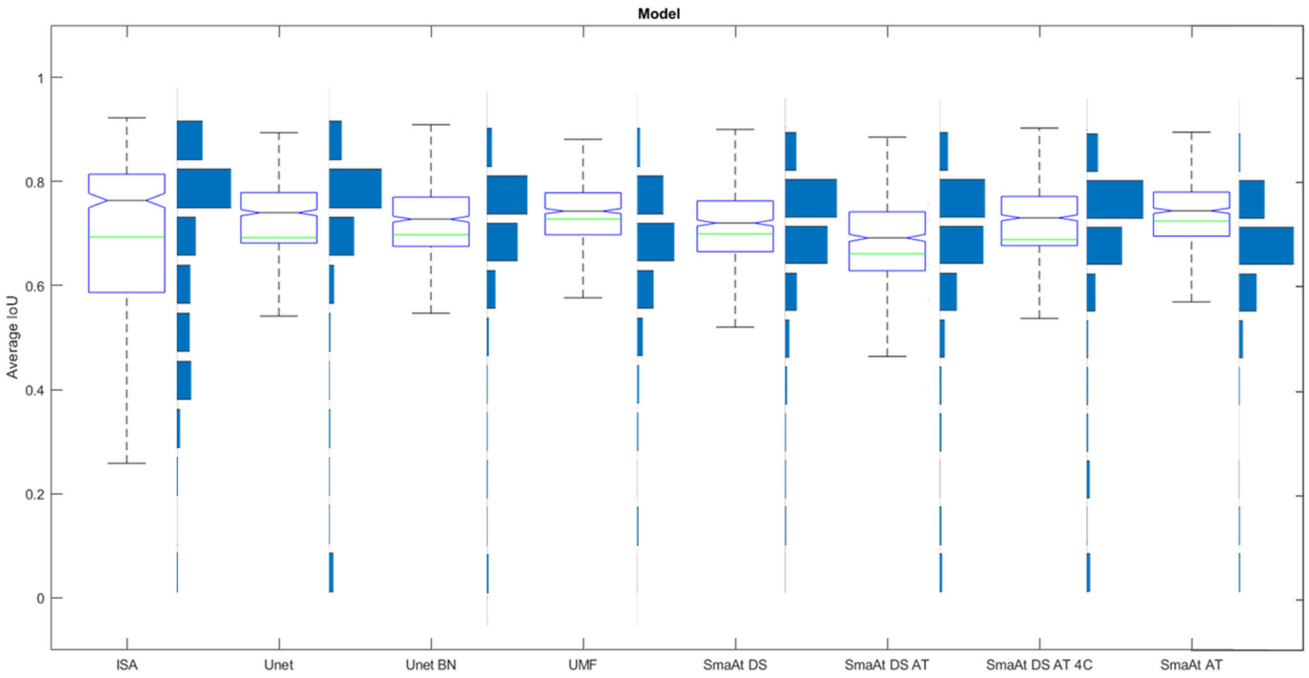


**Fig. 16** Comparison of previous work Layana Castro et al. (2020) with U-Nets architectures. Green line indicates the mean in both graphs and gray line indicates the median. ISA N = 664, mean = 0.6936, median = 0.7635, standard deviation = 0.1649, variance = 0.0272. U-Net N = 664, mean = 0.6923, median = 0.7400, standard deviation = 0.1757, variance = 0.0309. U-Net A. N = 664, mean = 0.6977, median = 0.7276, standard deviation = 0.1378, variance = 0.0190. UMF U-Net N = 664, mean = 0.7279, median = 0.7430, standard deviation = 0.0871, variance = 0.0076. SmaAT DS N = 664, mean = 0.6993, median = 0.7201, standard deviation = 0.1135, variance = 0.0129. SmaAT DS AT N = 664, mean = 0.6613, median = 0.6916, standard deviation = 0.1407, variance = 0.0198. SmaAT DS AT 4C N = 664, mean = 0.6884, median = 0.7302, standard deviation = 0.1646, variance = 0.0271. SmaAT AT N = 664, mean = 0.7240, median = 0.7436, standard deviation = 0.1080, variance = 0.0117

## Appendix B. Analysis Results Tables

Table 4, Table 5, Table 6, show the average results from IoU and error pixel distance (E.D.) for each problematic case and for each skeleton segmentation class.

**Table 4** Analysis of results by class (worm ends, worm body) for aggregation cases

| | Avg. worm-ends class | | Avg. worm-body class | | Total average worm | |
|---|---|---|---|---|---|---|
| Model | IoU | E.D | IoU | E.D | IoU ± IC 95% | E.D.± IC 95% |
| ISA | 0.6627 | 0.5787 | 0.7635 | 0.5604 | 0.7625±0.0065 | 0.5659±0.0206 |
| U-Net | **0.6725** | **0.5634** | **0.768** | **0.5554** | **0.7634±0.0046** | **0.5551±0.0186** |
| U-Net A | 0.5391 | 0.8740 | 0.7027 | 0.6915 | 0.6858±0.0103 | 0.7822±0.0714 |
| UMF U-Net | 0.5848 | 0.6093 | 0.7400 | 0.6138 | 0.6992±0.0083 | 0.6090±0.0151 |
| SmaAt DS | 0.4558 | 0.5673 | 0.6552 | 0.5810 | 0.6339±0.0141 | 0.5737±0.0217 |
| SmaAt DS AT | 0.4343 | 0.5989 | 0.5819 | 0.5822 | 0.5759±0.0184 | 0.5870±0.0282 |
| SmaAt DS AT 4C | 0.4374 | 0.7422 | 0.5637 | 0.7512 | 0.5713±0.0219 | 0.7471±0.0394 |
| SmaAt AT | 0.5551 | 0.8472 | 0.7135 | 0.8385 | 0.6767±0.0107 | 0.8359±0.0430 |

**Table 5** Analysis of results by class (worm ends, worm body) for aggregation with noise cases

| | Avg. worm-ends class | | Avg. worm-body class | | Total average worm | |
|---|---|---|---|---|---|---|
| Model | IoU | E.D | IoU | E.D | IoU ± IC 95% | E.D.± IC 95% |
| ISA | 0.5836 | 2.2960 | 0.7031 | 1.6793 | 0.6421±0.0137 | 2.1726±0.1731 |
| U-Net | 0.5650 | 0.9349 | 0.6678 | 0.8802 | 0.6510±0.0160 | 0.9001±0.0470 |
| U-Net A | 0.5540 | 0.7804 | 0.7359 | 0.6232 | 0.6980±0.0116 | 0.6712±0.0335 |
| UMF U-Net | 0.5959 | 0.8476 | 0.7691 | 0.6358 | 0.7313±0.0053 | 0.7259±0.0412 |
| SmaAt DS | 0.5501 | 0.7772 | 0.7509 | 0.5992 | 0.7133±0.0080 | 0.6562±0.0332 |
| SmaAt DS AT | 0.5237 | 0.7426 | 0.7135 | 0.6125 | 0.6849±0.0085 | 0.6505±0.0199 |
| SmaAt DS AT 4C | 0.5908 | 0.8142 | 0.7493 | 0.6481 | 0.7193±0.0079 | 0.7032±0.0188 |
| SmaAt AT | **0.6041** | **0.7312** | **0.7783** | **0.6097** | **0.7338±0.0049** | **0.6459±0.0382** |

**Table 6** Analysis of results by class (worm ends, worm body) for rolled cases

| | Avg. worm-ends class | | Avg. worm-body class | | Total average worm | |
|---|---|---|---|---|---|---|
| Model | IoU | E.D | IoU | E.D | IoU ± IC 95% | E.D.± IC 95% |
| ISA | **0.7268** | **0.5685** | **0.8066** | **0.5083** | **0.8122±0.0040** | **0.554±0.0122** |
| U-Net | 0.6796 | 0.6797 | 0.7507 | 0.6497 | 0.7600±0.0037 | 0.6678±0.0114 |
| U-Net A | 0.6126 | 0.6267 | 0.7108 | 0.6027 | 0.7172±0.0097 | 0.6089±0.0164 |
| UMF U-Net | 0.6854 | 0.6142 | 0.7644 | 0.5680 | 0.7622±0.0048 | 0.5686±0.0124 |
| SmaAt DS | 0.6373 | 0.6190 | 0.7463 | 0.5762 | 0.7481±0.0046 | 0.5905±0.012 |
| SmaAt DS AT | 0.5845 | 0.6404 | 0.6897 | 0.6103 | 0.7004±0.0059 | 0.6150±0.0138 |
| SmaAt DS AT 4C | 0.6640 | 0.6369 | 0.7391 | 0.5911 | 0.7497±0.0044 | 0.6053±0.0164 |
| SmaAt AT | 0.6602 | 0.6038 | 0.7576 | 0.6173 | 0.7610±0.0077 | 0.6146±0.0125 |

## References


Alexandre, M. (2019). Pytorch-unet. Code https://github.com/milesial/Pytorch-UNet.

Baheti, B., Innani, S., Gajre, S., et al. (2020). Eff-unet: A novel architecture for semantic segmentation in unstructured environment. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Seattle, pp. 1473–1481, https://doi.org/10.1109/CVPRW50498.2020.00187.

Bargsten, L., & Schlaefer, A. (2020). Specklegan: a generative adversarial network with an adaptive speckle layer to augment limited training data for ultrasound image processing. *International Journal of Computer Assisted Radiology and Surgery, 15*(9), 1427–1436. https://doi.org/10.1007/s11548-020-02203-1

Biron, D., Haspel, G. (eds) (2015) *C . elegans*. Springer Science+Business Media, New York. https://doi.org/10.1007/978-1-4939-2842-2

Cao, K., & Zhang, X. (2020). An improved res-unet model for tree species classification using airborne high-resolution images. *Remote Sensing*. https://doi.org/10.3390/rs12071128

Chen, L., Strauch, M., Daub, M., et al (2020) A cnn framework based on line annotations for detecting nematodes in microscopic images. In: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE, Iowa City, IA, USA, pp. 508–512. https://doi.org/10.1109/ISBI45749.2020.9098465

Chen, Z., Ouyang, W., Liu, T., et al. (2021). A shape transformation-based dataset augmentation framework for pedestrian detection. *International Journal of Computer Vision, 129*(4), 1121–1138. https://doi.org/10.1007/s11263-020-01412-0

Conn, P. M. (Ed.). (2017). *Animal models for the study of human disease*. Texas: Sara Tenney.

Dewi, C., Chen, R. C., Liu, Y. T., et al. (2021). Yolo v4 for advanced traffic sign recognition with synthetic training data generated by various gan. *IEEE Access, 9*, 97,228-97,242. https://doi.org/10.1109/ACCESS.2021.3094201

Di Rosa, G., Brunetti, G., Scuto, M., et al. (2020). Healthspan enhancement by olive polyphenols in C. elegans wild type and Parkinson's models. *International Journal of Molecular Sciences*. https://doi.org/10.3390/ijms21113893

Doshi, K. (2019) Synthetic image augmentation for improved classification using generative adversarial networks. arXiv preprint arXiv:1907.13576.

García Garví, A., Puchalt, J. C., Layana Castro, P. E., et al. (2021). Towards lifespan automation for Caenorhabditis elegans based on deep learning: Analysing convolutional and recurrent neural networks for dead or live classification. *Sensors*. https://doi.org/10.3390/s21144943

Hahm, J. H., Kim, S., DiLoreto, R., et al. (2015). C. elegans maximum velocity correlates with healthspan and is maintained in worms with an insulin receptor mutation. *Nature Communications, 6*(1), 1–7. https://doi.org/10.1038/ncomms9919

Han, L., Tao, P., & Martin, R. R. (2019). Livestock detection in aerial images using a fully convolutional network. *Computational Visual Media, 5*(2), 221–228. https://doi.org/10.1007/s41095-019-0132-5

Hebert, L., Ahamed, T., Costa, A. C., et al. (2021). Wormpose: Image synthesis and convolutional networks for pose estimation in C. elegans. *PLOS Computational Biology, 17*(4), 1–20. https://doi.org/10.1371/journal.pcbi.1008914

Hinterstoisser, S., Pauly, O., Heibel, H., et al (2019) An annotation saved is an annotation earned: Using fully synthetic training for object detection. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, Seoul, Korea (South), pp. 2787–2796. https://doi.org/10.1109/ICCVW.2019.00340

Huang, H., Lin, L., Tong, R., et al (2020) Unet 3+: A full-scale connected unet for medical image segmentation. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Barcelona, Spain, pp. 1055–1059. https://doi.org/10.1109/ICASSP40776.2020.9053405

Ioffe, S., Szegedy, C. (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: F. Bach, D. Blei (eds) *Proceedings of the 32nd International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol 37. PMLR, Lille, France, pp. 448–456

Iqbal, H. (2018) Harisiqbal88/plotneuralnet v1.0.0. Code https://github.com/HarisIqbal88/PlotNeuralNet.

Isensee, F., Jaeger, P. F., Kohl, S. A., et al. (2021). nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods, 18*(2), 203–211. https://doi.org/10.1038/s41592-020-01008-z

Javer, A., Currie, M., Lee, C. W., et al. (2018). An open-source platform for analyzing and sharing worm-behavior data. *Nature Methods, 15*(9), 645–646. https://doi.org/10.1038/s41592-018-0112-1

Javer, A., Brown, A.E., Kokkinos, I., et al. (2019). Identification of *C. elegans* strains using a fully convolutional neural network on behavioural dynamics. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, vol 11134. Springer, Cham, pp. 0–0. https://doi.org/10.1007/978-3-030-11024-6_35

Jung, S. K., Aleman-Meza, B., Riepe, C., et al. (2014). Quantworm: A comprehensive software package for Caenorhabditis elegans phenotypic assays. *PLOS ONE, 9*(1), 1–9. https://doi.org/10.1371/journal.pone.0084830

Koopman, M., Peter, Q., Seinstra, R. I., et al. (2020). Assessing motor-related phenotypes of Caenorhabditis elegans with the wide field-of-view nematode tracking platform. *Nature protocols, 15*(6), 2071–2106. https://doi.org/10.1038/s41596-020-0321-9

Koul, A., Ganju, S., Kasam, M. (2019). Practical Deep Learning for Cloud, Mobile and Edge: Real-World AI and Computer Vision Projects Using Python, Keras and TensorFlow. O'Reilly Media, Incorporated. https://www.oreilly.com/library/view/practical-deep-learning/9781492034858/

Kumar, S., Egan, B. M., Kocsisova, Z., et al. (2019). Lifespan extension in C. elegans caused by bacterial colonization of the intestine and subsequent activation of an innate immune response. *Developmental Cell, 49*(1), 100-117.e6. https://doi.org/10.1016/j.devcel.2019.03.010

Layana Castro, P. E., Puchalt, J. C., & Sánchez-Salmerón, A. J. (2020). Improving skeleton algorithm for helping Caenorhabditis elegans trackers. *Scientific Reports, 10*(1), 22,247. https://doi.org/10.1038/s41598-020-79430-8

Layana Castro, P. E., Puchalt, J. C., García Garví, A., et al. (2021). Caenorhabditis elegans multi-tracker based on a modified skeleton algorithm. *Sensors*. https://doi.org/10.3390/s21165622

Le, K. N., Zhan, M., Cho, Y., et al. (2020). An automated platform to monitor long-term behavior and healthspan in Caenorhabditis elegans under precise environmental control. *Communications Biology, 3*(1), 1–13. https://doi.org/10.1038/s42003-020-1013-2

Li, H., Fang, J., Liu, S., et al. (2020). Cr-unet: A composite network for ovary and follicle segmentation in ultrasound images. *IEEE Journal of Biomedical and Health Informatics, 24*(4), 974–983. https://doi.org/10.1109/JBHI.2019.2946092

Li, S., Günel, S., Ostrek, M., et al. (2020b) Deformation-aware unpaired image translation for pose estimation on laboratory animals. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, pp. 13155–13165. https://doi.org/10.1109/CVPR42600.2020.01317

Liu, X., Zhou, T., Lu, M., et al. (2020). Deep learning for ultrasound localization microscopy. *IEEE Transactions on Medical Imaging, 39*(10), 3064–3078. https://doi.org/10.1109/TMI.2020.2986781

Mais, L., Hirsch, P., Kainmueller, D. (2020). Patchperpix for instance segmentation. In: *European Conference on Computer Vision, Springer*, vol. 12370. Springer, Cham, pp. 288–304. https://doi.org/10.1007/978-3-030-58595-2_18

Mane, M. R., Deshmukh, A. A., Iliff A. J. (2020) Head and tail localization of *C. elegans*. arXiv preprint arXiv:2001.03981. https://doi.org/10.48550/arXiv.2001.03981

Mayershofer, C., Ge, T., Fottner, J. (2021). Towards fully-synthetic training for industrial applications. In: *LISS 2020*. Springer, Singapore, pp. 765–782. https://doi.org/10.1007/978-981-33-4359-7_53

McManigle, J. E., Bartz, R. R., Carin, L. (2020). Y-net for chest x-ray preprocessing: Simultaneous classification of geometry and segmentation of annotations. In: *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*. IEEE, Montreal, QC, Canada, pp. 1266–1269. https://doi.org/10.1109/EMBC44109.2020.9176334

Moradi, S., Oghli, M. G., Alizadehasl, A., et al. (2019). Mfp-unet: A novel deep learning based approach for left ventricle segmentation in echocardiography. *Physica Medica, 67*, 58–69. https://doi.org/10.1016/j.ejmp.2019.10.001

Olsen, A., Gill, M. S., (eds) (2017) Ageing: Lessons from *C. elegans*. Springer International Publishing, Switzerland. https://doi.org/10.1007/978-3-319-44703-2.

Padubidri, C., Kamilaris, A., Karatsiolis, S., et al. (2021). Counting sea lions and elephants from aerial photography using deep learning with density maps. *Animal Biotelemetry, 9*(1), 1–10. https://doi.org/10.1186/s40317-021-00247-x

Pashevich, A., Strudel, R., Kalevatykh, I., et al (2019) Learning to augment synthetic images for sim2real policy transfer. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Macau, China, pp. 2651–2657. https://doi.org/10.1109/IROS40897.2019.8967622.

Pitt, J. N., Strait, N. L., Vayndorf, E. M., et al. (2019). Wormbot, an open-source robotics platform for survival and behavior analysis in C. elegans. *GeroScience, 41*(6), 961–973. https://doi.org/10.1007/s11357-019-00124-9

Plebani, E., Biscola, N. P., Havton, L. A., et al. (2022). High-throughput segmentation of unmyelinated axons by deep learning. *Scientific Reports, 12*(1), 1–16. https://doi.org/10.1038/s41598-022-04854-3

Puchalt, J. C., Sánchez-Salmerón, A. J., Martorell Guerola, P., et al. (2019). Active backlight for automating visual monitoring: An analysis of a lighting control technique for Caenorhabditis elegans cultured on standard petri plates. *PLOS ONE, 14*(4), 1–18. https://doi.org/10.1371/journal.pone.0215548

Puchalt, J. C., Layana Castro, P. E., & Sánchez-Salmerón, A. J. (2020). Reducing results variance in lifespan machines: An analysis of the influence of vibrotaxis on wild-type Caenorhabditis elegans for the death criterion. *Sensors*. https://doi.org/10.3390/s20215981

Puchalt, J. C., Sánchez-Salmerón, A. J., Eugenio, I., et al. (2021). Small flexible automated system for monitoring Caenorhabditis elegans lifespan based on active vision and image processing techniques. *Scientific Reports*. https://doi.org/10.1038/s41598-021-91898-6

Puchalt, J. C., Gonzalez-Rojo, J. F., Gómez-Escribano, A. P., et al. (2022). Multiview motion tracking based on a cartesian robot to monitor Caenorhabditis elegans in standard petri dishes. *Scientific Reports, 12*(1), 1–11. https://doi.org/10.1038/s41598-022-05823-6

Qamar, S., Jin, H., Zheng, R., et al. (2020). A variant form of 3d-unet for infant brain segmentation. *Future Generation Computer Systems, 108*, 613–623. https://doi.org/10.1016/j.future.2019.11.021

Rizvandi, N. B., Pizurica, A., Philips, W. (2008a). Machine vision detection of isolated and overlapped nematode worms using skeleton analysis. In: *2008 15th IEEE International Conference on Image Processing*. IEEE, San Diego, CA, USA, pp. 2972–2975. https://doi.org/10.1109/ICIP.2008.4712419

Rizvandi, N. B., Pižurica, A., Rooms, F., (2008b) Skeleton analysis of population images for detection of isolated and overlapped nematode C. elegans. In: *16th European Signal Processing Conference*, pp. 1–5. Lausanne, Switzerland: IEEE.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention, Springer*, vol. 9351, pp. 234–241. Cham: Springer.

Schraml, D. (2019). Physically based synthetic image generation for machine learning: a review of pertinent literature. In: *Photonics and Education in Measurement Science 2019, International Society for Optics and Photonics, Jena, Germany*, pp. 111440J. https://doi.org/10.1117/12.2533485.

Stiernagle, T. (2006). Maintenance of *C. elegans*. https://doi.org/10.1895/wormbook.1.101.1. https://www.ncbi.nlm.nih.gov/books/NBK19649/?report=classic

Tang, P., Liang, Q., Yan, X., et al. (2019). Efficient skin lesion segmentation using separable-unet with stochastic weight averaging. *Computer Methods and Programs in Biomedicine, 178*, 289–301. https://doi.org/10.1016/j.cmpb.2019.07.005

Trebing, K., Stanczyk, T., & Mehrkanoon, S. (2021). Smaat-unet: Precipitation nowcasting using a small attention-unet architecture. *Pattern Recognition Letters, 145*, 178–186. https://doi.org/10.1016/j.patrec.2021.01.036

Tschandl, P., Sinz, C., & Kittler, H. (2019). Domain-specific classification-pretrained fully convolutional network encoders for skin lesion segmentation. *Computers in Biology and Medicine, 104*, 111–116. https://doi.org/10.1016/j.compbiomed.2018.11.010

Tsibidis, G. D., & Tavernarakis, N. (2007). Nemo: a computational tool for analyzing nematode locomotion. *BMC Neuroscience*. https://doi.org/10.1186/1471-2202-8-86

Uhlmann, V., Unser, M. (2015) Tip-seeking active contours for bioimage segmentation. In: *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*. IEEE, Brooklyn, NY, USA, pp. 544–547. https://doi.org/10.1109/ISBI.2015.7163931.

Wang, D., Lu, Z., Bao, Z. (2019). Augmenting *C. elegans* microscopic dataset for accelerated pattern recognition. arXiv preprint arXiv:1906.00078. https://doi.org/10.48550/arXiv.1906.00078

Wang, L., Kong, S., Pincus, Z., et al. (2020). Celeganser: Automated analysis of nematode morphology and age. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Seattle, WA, USA, pp. 4164–4173. https://doi.org/10.1109/CVPRW50498.2020.00492

Wiehman, S., de Villiers, H. (2016). Semantic segmentation of bioimages using convolutional neural networks. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Vancouver, BC, Canada, pp. 624–631, https://doi.org/10.1109/IJCNN.2016.7727258.

Wiles, O., & Zisserman, A. (2019). Learning to predict 3d surfaces of sculptures from single and multiple views. *International Journal of Computer Vision, 127*(11), 1780–1800. https://doi.org/10.1007/s11263-018-1124-0

Winter, P. B., Brielmann, R. M., Timkovich, N. P., et al. (2016). A network approach to discerning the identities of C. elegans in a free moving population. *Scientific Reports, 6*, 34859. https://doi.org/10.1038/srep34859

Wöhlby, C., Kamentsky, L., Liu, Z., et al. (2012). An image analysis toolbox for high-throughput C. elegans assays. *Nature methods, 9*, 714–6. https://doi.org/10.1038/nmeth.1984

Yu, C. C. J., Raizen, D. M., & Fang-Yen, C. (2014). Multi-well imaging of development and behavior in Caenorhabditis elegans. *Journal of Neuroscience Methods, 223*, 35–39. https://doi.org/10.1016/j.jneumeth.2013.11.026

Yu, X., Creamer, M. S., Randi, F., et al. (2021). Fast deep neural correspondence for tracking and identifying neurons in C. elegans using semi-synthetic training. *eLife, 10*, e66,410. https://doi.org/10.7554/eLife.66410

Zhao, X., Yuan, Y., Song, M., et al. (2019). Use of unmanned aerial vehicle imagery and deep learning unet to extract rice lodging. *Sensors*. https://doi.org/10.3390/s19183859

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.