Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Automatic segmentation of *Caenorhabditis elegans* skeletons in worm aggregations using improved U-Net in low-resolution image sequences

Pablo E. Layana Castro, Antonio García Garví, Antonio-José Sánchez-Salmerón [*]

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

ARTICLE INFO

Keywords:
Caenorhabditis elegans
Skeletonizing
Synthetic dataset
Low-resolution image
U-Net

ABSTRACT

Pose estimation of *C. elegans* in image sequences is challenging and even more difficult in low-resolution images. Problems range from occlusions, loss of worm identity, and overlaps to aggregations that are too complex or difficult to resolve, even for the human eye. Neural networks, on the other hand, have shown good results in both low-resolution and high-resolution images. However, training in a neural network model requires a very large and balanced dataset, which is sometimes impossible or too expensive to obtain.

In this article, a novel method for predicting *C. elegans* poses in cases of multi-worm aggregation and aggregation with noise is proposed. To solve this problem we use an improved U-Net model capable of obtaining images of the next aggregated worm posture. This neural network model was trained/validated using a custom-generated dataset with a synthetic image simulator. Subsequently, tested with a dataset of real images. The results obtained were greater than 75% in precision and 0.65 with Intersection over Union (IoU) values.

1. Introduction

Caenorhabditis elegans (*C. elegans*) are widely studied animal models, their behavior provides valuable information for many researchers in the field of biology [1]. This animal model is attractive to study and evaluate the different effects of its mutations such as the treatment of aging, age-related pathologies and neurodegenerative disorders in at advanced ages [2].

Currently, there are many semi-automatic or automatic applications for predicting worm pose, and tracking single or multiple worms. These applications that use traditional computer vision techniques have shown good results in pose prediction [3,4], behavior analysis [5–7], locomotion [8], location of multiple worms [8], calculation of the average speed of multiple worms [9], and even using low-cost tools [10]. Pose prediction is a very important technique in tracking applications because it allows us to automate all these types of tests mentioned, lifespan and healthspan, among others. However, it is often a challenge when there is overlap and contact between worm bodies or plate noise. Some applications often have trouble resolving these cases [11–13]. In a previous work [14], we showed that using an optimizer with temporal information of the images, a better solution could be obtained in these cases of aggregation.

To train a neural network, a large dataset is necessary, and in certain applications such as *C. elegans* as well as some animal models

* Corresponding author.
E-mail address: asanchez@isa.upv.es (A.-J. Sánchez-Salmerón).

https://doi.org/10.1016/j.heliyon.2023.e14715
Received 16 September 2022; Received in revised form 14 March 2023; Accepted 15 March 2023
Available online 22 March 2023

or living organisms, it is usually difficult to obtain. Different methods have been proposed to try to address this issue, for instance some applications propose data augmentation techniques using synthetic images, while others propose a purely synthetic dataset. Both methods have obtained satisfactory results on high-resolution images. Currently, there are software applications that use neural networks for worm pose prediction. For multiple worm cases [15], recognizes live/dead worms, while [16,17] combine neural networks with traditional machine vision techniques to separate connected worms. These mentioned cases are performed on static images and images of high-resolution.

In this work, a novel method is proposed for predicting *C. elegans* poses (skeletons) for multi-worm aggregations in low-resolution gray image sequences. Although working with low-resolution images of complete 55 mm petri dishes allows automating many assays for motility, lifespan, etc, it also presents a limitation in extreme cases of aggregation. For example, in cases of aggregations with a large amount of noise, where a large part of the worm's body has been lost, the proposed method obtained a greater loss of identity than in the other cases. Certain cases of parallel aggregation, where the worms are closely linked, are also challenging, as the identity of one of them can be lost. The proposed method uses deep neural networks and traditional image processing techniques to predict worm skeletons and track them throughout a sequence of images. The neural network model was trained and validated with a purely synthetic low-resolution gray image dataset and tested with a real image dataset. Our synthetic images aim to simulate interactions between worms, as well as individual movements, while wormpose [18] only simulates individual worm behaviors. Specifically, the generated behaviors are parallel aggregations and cross aggregations. Simulating these behaviors is very interesting since unbalanced many of these cases are not available in the real dataset and they are the most difficult to solve in detection and monitoring applications. Although our synthetic images generated aggregation behaviors between worms, this method could only be applied to low-resolution images.

2. Methods

2.1. *Caenorhabditis elegans strain and culture conditions*

Two types of adult-young *C. elegans* strains were used: N2 and CB1370 *daf-2*(*e1370*). Both strains were obtained from worm eggs incubated at 20 °C in NGM plates measuring 55 mm in diameter. To prevent reproduction, FUDR (0.2 mM) was used. *Escherichia coli* strain OP50 was used as a standard diet. To avoid fungal contamination, fungizone (1 μg/mL) was added and the plates were closed with a lid. The standard method [19] was followed to remove contaminated plates. 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).

2.2. *Image capture system*

First, a laboratory operator collected the plates from the incubator and inserted them into the capture system. The program captured the 30-image sequence at an interval of 1 s per image. The ambient temperature inside and outside the incubator was 20 °C to avoid condensation. All experiments were performed with lids and *E. coli* OP50 seeded in the center of the plates to prevent worms from moving out of the field of view by climbing on or near the plate edges. The worms are adult-young and a time period of 1 s per image, or 1 Hz, was enough to capture their movements. In our images, the length of each worm varies from 20 to 40 pixels. At this stage of their life their movement is slower than when they are younger. This movement is between 1 and 15 pixels in our image

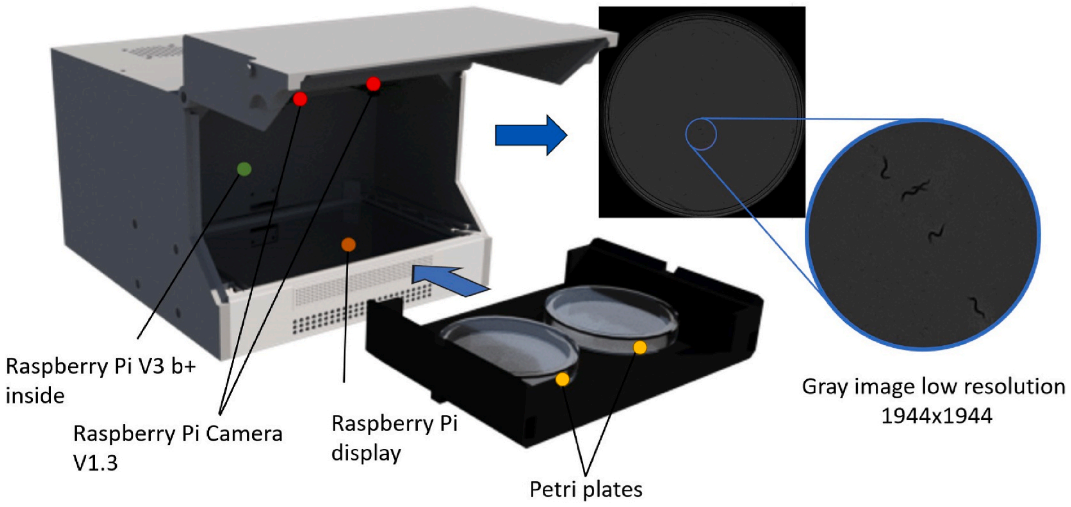

Fig. 1. SIVIS image capture system. The figure presents all the parts and elements of the SIVIS [20] capture system, such as an example of a grayscale image captured by the system.

settings.

The capture system used a Raspberry Pi v1.3 RGB camera (Rasberry Pi Foundation), OmniVision OV5647 with a resolution of 2592 × 1944 pixels, pixel size of 1.4 × 1.4 μm, field of view of 53.50° × 41.41°, optical size of 1/4″ and focal ratio of 2.9. The lighting system was based on a 7" Raspberry Pi screen with a resolution of 800 × 480 at 60 fps, 24-bit RGB color, and a Raspberry Pi 3 as a processing unit. The lighting system used active backlighting, a technique that reduces the variability of the captured images, keeping the grayscale more constant. This technique has proven to be effective in *C. elegans* image capture systems [20,21]. SIVIS [20] is a *C. elegans* monitoring machine that can analyze *C. elegans* cultures using standard Petri dishes seeded with *E. coli*. SIVIS [20] has a flexible and compact platform design, which allows it to be adapted to longevity assays, as well as other studies of *C. elegans*, such as toxicity, motility, and behavior. The SIVIS [20] image capture system (Fig. 1) is open hardware and software, whose assembly and image capture processes are described in detail in Ref. [20].

2.3. *Synthetic image acquisition method*

To train and validate the neural network we created a dataset of synthetic images based on 600 sequences of 30 images with worm aggregations. The synthetic images were generated in two stages. The first stage, Simulator, was used offline before training to generate *.PTS files. PTS files are text files and contain information on color, width, and location (X, Y) of each pixel of the worm's skeleton. The second stage was used during the training/validation phase and used the information from these *.PTS files to create low-resolution grayscale images of worms.

To create the *.PTS files (worm aggregate sequences) the Simulator used images of empty Petri dishes (free of worms), and individual moving worm sequence files. Each worm file (XML file) contained skeleton point information ordered during a sequence (30 skeletons), it also contained color information and width values for each skeleton point. The simulator randomly selected worm files and produced aggregations by randomly matching 1 skeleton point from each worm. Finally, an X–Y position and rotation angle were also randomly generated to position the worms within the Petri dish image.

In total, 628 worm files were used to produce all the new 30-move sequences. These worm files were manually tagged, by a tagger person, from the sequence of movements of worms moving freely in Petri dishes.

2.4. *Proposed neural network method*

This work aims to obtain the current pose of a worm within an aggregation using a sequence of 2 Gy images (previous and current image) and a binary image of the skeleton of the worm of interest (previous skeleton image). The proposed neural network model (Fig. 2) consists of six neural network blocks, three encoder blocks marked in Fig. 2 by blue, orange, and dark orange colors correspondingly, one Long Short-Term Memory block (LSTM) marked in Fig. 2 by green, one Linear block marked in Fig. 2 by violet, and one decoder block marked in Fig. 2 by yellow. The outputs of the encoders generate feature blocks for each input image. The feature blocks in the output of the gray image encoders are exploited by the LSTM to predict future pose. The linear block matches the feature depth of the LSTM output with the binary image encoder output feature block in order to concatenate features of the individual of interest. Finally, the decoder block reconstructs the output image from the input features and the concatenation of features during encoder compression.

Alexandre's U-Net convolutional neural network architecture [22] was used for the encoder and decoder blocks, this architecture is similar to the traditional U-Net [23] but after each convolution2d layer a BatchNorm2d layer is added. Convolution2d (torch.nn.conv2d) and BatchNorm2d (torch.nn.BatchNorm2d) are functions of Pytorch libraries. Convolution2d applies filter convolutions on its 2D input, while BatchNorm2d normalizes the input data for each channel independently. For all the 2 d Convolution layers in the encoders as well as in the decoder, kernel_size = 3 and padding = 1 were used, except for the last 2 d Convolution layer, where kernel_size = 1 and padding = 0. As mentioned in Ref. [24] applying a batch normalization step after each convolutional layer allows

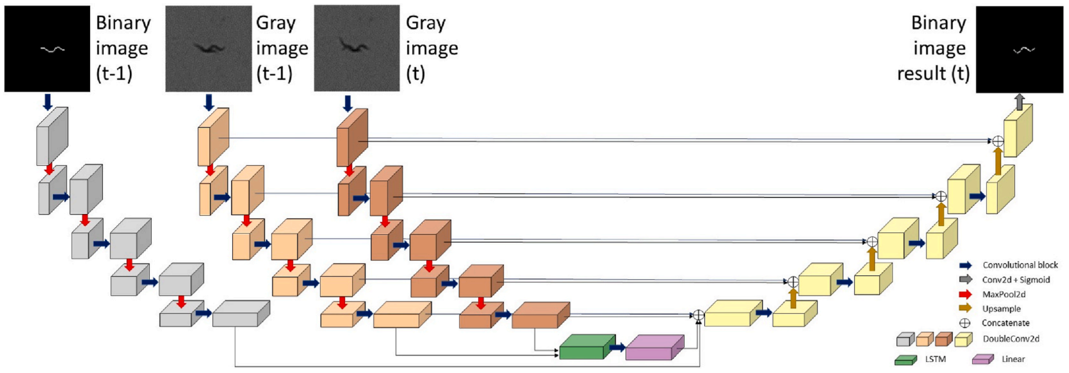

Fig. 2. Proposed neural network model. Blue, orange, dark orange, and yellow blocks correspond to the encoding and decoding blocks of Alexandre's U-Net convolutional neural network model [22]. The green block corresponds to the LSTM block and, lastly, the violet block corresponds to a linear block.

regularization of the data, such as better convergence, thereby reducing the internal covariate change in the dataset.

The input of all the encoder blocks as well as the output of the decoder block were 100 × 100 pixels images. This ensured that the next worm move (network output), measuring 40 pixels long, would fall within the window.

2.5. *Training/validation and testing method*

To evaluate the proposed model, two datasets were used, a dataset of synthetic images for the training and validation stages, and a dataset of real images to test the results in cases of multi-worm aggregation, and aggregation with noise (Fig. 3).

The image was divided into fixed windows of 100 × 100 pixels due to the input image dimensions (1944 × 1944 pixels) and the limitations of the hardware to train, validate and test the proposed network model (78.76 M parameters). Each window contained a worm in the image center. A 3 × 3 normalized box filter was applied using the OpenCV tool (cv.blur(3,3)) for all real and synthetic grayscale image windows. This filter uses a kernel (3 × 3 matrix) to average the pixels that lie in that area and replace the central element with the calculated average. This filter was essential to bring both datasets (synthetic and real) closer to the synthetic-real image domain. It allowed to generalize both domains. The 3 × 3 size is the minimum size within the OpenCV tool and allowed to perform this operation.

The loss function to train/validate the network was the PyTorch function “BCELoss()”. Binary cross entropy loss or BCELoss() is a Pytorch loss function for binary classification i.e. for single unit output classification. The metrics to evaluate the real dataset are detailed in “Results validation methods".

2.6. *Post-processing method*

This method (Fig. 4) comprised three stages, the first stage of possible skeleton solutions was generated using the method proposed in Ref. [4]. This method generated possible skeleton solutions using endpoints and intersection points in the improved skeleton image. The improved skeleton image was created using information of the worm widths, to do so the algorithm set pixels in the background (pixels equal to zero) when the thickness in a segmentation exceeded the maximum width of the worm. The maximum width of the worm is the distance value from the widest edge of the worm’s body to its opposite. This value is usually found in the center of the worm’s body. In our images, the length of each worm varies from 20 to 40 pixels, while the width varies from two to four pixels. This enabled skeletons to be separated during an aggregation or when self-occluding.

The post-processing second stage used the skeleton predictions obtained from the output of the neural network model and the possible skeleton solutions of each worm to filter those solutions with a coincidence percentage below 65% (F = 0.65). The F value selected was the value closest to 1 on a precision-recall curve.

The third and final stage used the optimizer proposed in Ref. [14] to obtain skeletons for each worm in an aggregation. The optimizer algorithm uses three criteria to evaluate all possible combinations of each worm skeleton in an aggregation. The worm skeletons used are obtained from the filter of possible solutions. The criteria are described in more detail in Ref. [14] and are: overlap with the previous pose, completeness of the segmentation, and color. The best possible solution is the minimum value of the sum of these three criteria in all possible combinations.

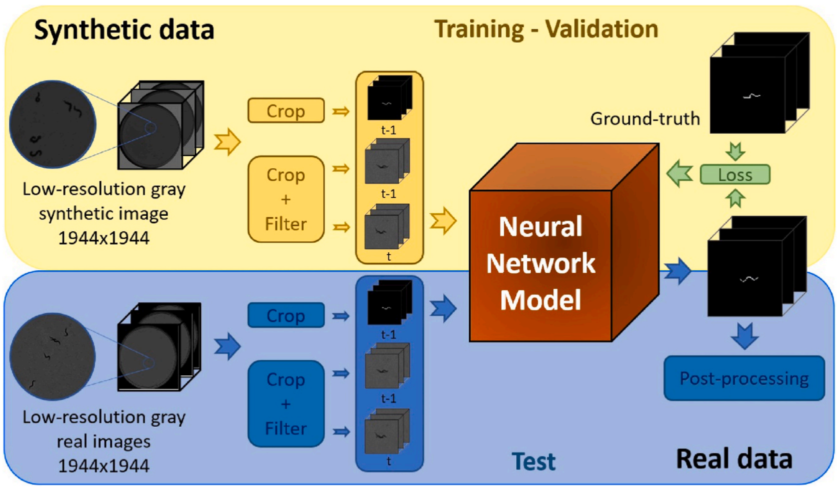

Fig. 3. Synthetic and real dataset pipeline. A dataset of synthetic images was used to train and validate the network. The trained network was used to test the real image domain dataset.

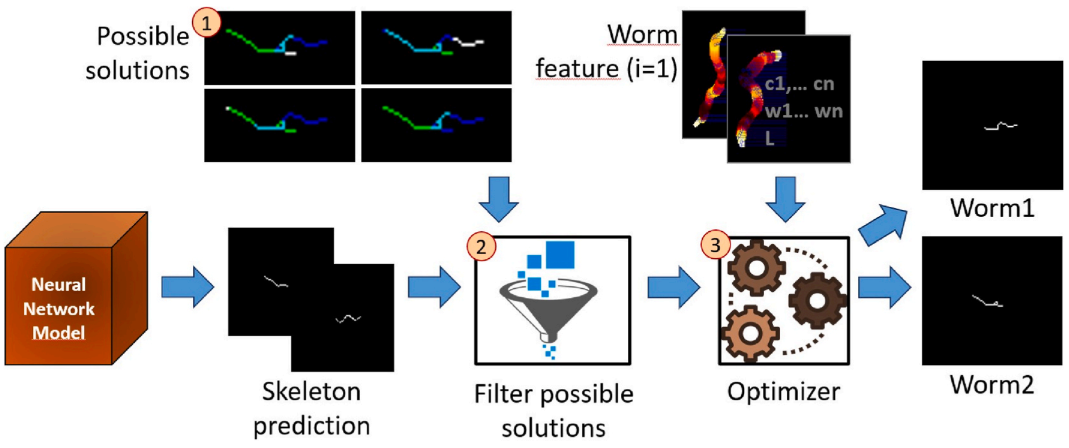

Fig. 4. Pipeline of the post-processing method. The post-processing method generates possible skeleton solutions for all worms in the actual skeleton image. Likewise, an optimizer proposed in Ref. [14] uses the filtered solutions, employing the predictions of the proposed neural network model and the features of each worm obtained in the first image to obtain the most probable skeletons for each worm.

2.7. Proposed tracking method

The tracking method consisted of going through an image sequence and predicting the current pose (skeleton) of each worm using the proposed neural network and a post-processing method to enhance skeletons. The input to the neural network included three cropped images measuring 100 × 100 pixels. Two images corresponded to the previous state (grayscale image and binary image of the worm skeleton) and one current grayscale image. The tracking method started at the second image because the first image was used to extract Worm Features (WFs), as well as being used to save the previous grayscale images (Img1) and skeleton images (SKL), and bounding boxes (tbbox). After each post-processing step an update process used the results obtained from skeletons and bounding boxes to move all worm windows (tbbox) to their new positions and to update the previous images. The process was repeated until all the images were finished, as well as shown in Algorithm 1.

In the first image, after obtaining the worm skeletons, the bounding boxes were obtained for each worm, placing the skeleton of each worm in the center of a 100 × 100 pixel window. The bounding boxes of each worm were used to crop 100 × 100 pixel windows for each worm and for each skeleton. Bounding boxes, as well as gray images and skeleton images, were saved in lists (tbbox, Img1, SKL, respectively). Before saving the cropped gray images, a 3 × 3 normalized squares filter was applied. To obtain the features of each worm, the cropped binary image of skeletons (location of pixels) was used and the color and width of each pixel of the skeleton was extracted from the cropped gray image (Img1). All this information on length, color and width was saved in a list for each worm (WFs).

3. Results validation method

The Intersection over Union (IoU) index was used to evaluate the precision of results obtained with the real dataset [25]. This metric, as its name indicates, measures precision by dividing the intersection of ground-truth areas and the results obtained for the union of these areas (Equation (1)).

$$IoU = \frac{\sum P_{w1} \bigcap P_{w2}}{\sum P_{w1} \bigcup P_{w2}} \tag{1}$$

Algorithm 1. Tracking algorithm. The algorithm shows the sequence taken by the images to track each worm. First, the features of each worm (WF|x|), the previous gray cropped images (Img1|x|), and their skeleton images (SKL|x|) are obtained. This information is used in the next stage (i>1), where the current cropped images (Img2|x|) are acquired and the poses of each worm are predicted (Pred|x|). A post_processing function generates all possible skeletons for each worm and filters them using the predictions to find the best solution. Information from tbbox, SKL and Img1 is updated with the obtained results. The procedure is repeated until the image sequence is complete.

```
Input: total_images, total_worms, path_network, gray_images
Net= Load(path_network)
for i = 1 to total_images do img = Get gray_images(i) for x = 1 to total_worms do
if i = = 1 then
tbbox[x] = Get bounding box ▷ bounding box for each worm Img1[x] = crop(img, tbbox[x])
SKL[x] = Get skeleton worm ▷ Binary image of the worm
skeleton
WFs[x] = Get worm's feature ▷ Colors of each skeleton pixel,
worm length
```

```
end if
if i > 1 then
Img2[x] = crop(img, tbbox[x])
Pred[x] = Net(Img1[x], Img2[x], SKL[x]) ▷ Network predictions of all worm skeleton
end if end for
if i > 1 then
SKL_opt, tbboxN = post_processing(img, Pred, WFs) ▷ Binary image of all worm skeleton and updated tbbox
Update: ▷ Update variables
tbbox = tbboxN SKL = SKL_opt
Img1 = crop(img, tbbox)
end if
end for
```

The ground-truth (manual labels) and the results obtained correspond to worm skeletons dilated with a disc of 2 to recover the shape of each worm (Fig. 5a and b respectively). The skeletons are sequences of pixels arranged on the centerline of each *C. elegans*. To obtain the skeletons of the manual labels (ground-truth), an annotator manually selected each point of the skeleton pixel by pixel, while the skeletons corresponding to the results were obtained after post-processing. For the examples in Fig. 5, d, e the IoU results were 0.7004, 0.8204 and 0.9409, respectively.

To obtain the value F = 0.65 (percentage of coincidence) used in the filter of possible solutions, two metrics were used: precision and recall (Equation (2), Equation (3) respectively). These metrics evaluated the performance of all the results obtained with the manual labels of the real dataset (ground-truth). These metrics, as well as the IoU index, were evaluated using the dilated skeletons of the manual labels and the predictions (Fig. 6c and d respectively). To obtain each of these metrics, four parameters were used: TP (true positives), FP (false positives), TN (true negative), FN (false negative) Fig. 6b. TP was obtained when ground-truth and prediction equaled one. FP was obtained when ground-truth equaled zero and prediction equaled one. FN was obtained when ground-truth equaled one and prediction equaled zero. Lastly, TN was obtained when ground-truth and prediction were equal to zero. The prediction and recall results for the examples in Fig. 6a, g were: precision = 0.8058, recall = 0.8426 (Fig. 6e and f) and precision = 0.9384, recall = 0.8671 for (Fig. 6h and i).

$$\text{Precision} = TP/TP + FP \tag{2}$$

$$\text{Recall} = TP/TP + FN \tag{3}$$

The Matlab 2018b Machine Learning Toolbox software was used to compare the statistics between the proposed tracking method with a filter equal to zero and a filter equal to 0.65. The Kolmogorov-Smirnov Test (for large samples greater than 50 data) was used to check data normality while the Wilcoxon Signed Rank Test was used to evaluate the statistical significance of the results with the F = 0.65 filter.

4. Experiments and results

A synthetic dataset was used to train/validate the proposed neural network model and a dataset of real images to test all the results. For the synthetic dataset, 600 sequences of 30 images (18000 images) were simulated, 70% of the sequences were used for training, that is, 420 sequences (12600 images), and the remaining 30%, 180 sequences (5400 images), for validation. Each simulated sequence image had two worms per plate, simulating different aggregation events. The images of each sequence were taken two by two starting from the first, that is, image1 with image2, image2 with image3, and so on until the end.

The hardware used for training and validation was a Gigabyte Technology Z390 AORUS PRO machine, Intel(R) Core (TM) i9-

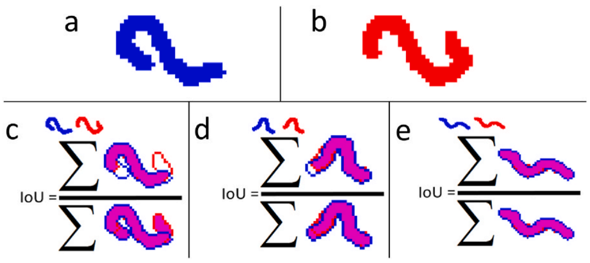

Fig. 5. IoU index. This index evaluated how accurate the response of the method was to the reference (ground-truth). The higher the IoU value, the more accurate the response to the reference. Predicted skeletons are not always exact, usually one or more pixels are offset from the reference (ground-truth). These offsets usually occur when the width of the worm is even, while when the width is odd, the skeleton pixel is the central one. To obtain a better measurement of the results, the evaluation was carried out on the dilated worm skeletons with a 2-pixel disc to recover the shape of each worm. (**a**) Body reconstructed from the manually labeled skeleton. (**b**) Body reconstructed from the skeleton obtained from the optimizer output. (**c-e**) Examples of validation of reconstructed skeletons, at the top of each example you can see the reference (blue), ground-truth, as well as the response of the automatic method (red).

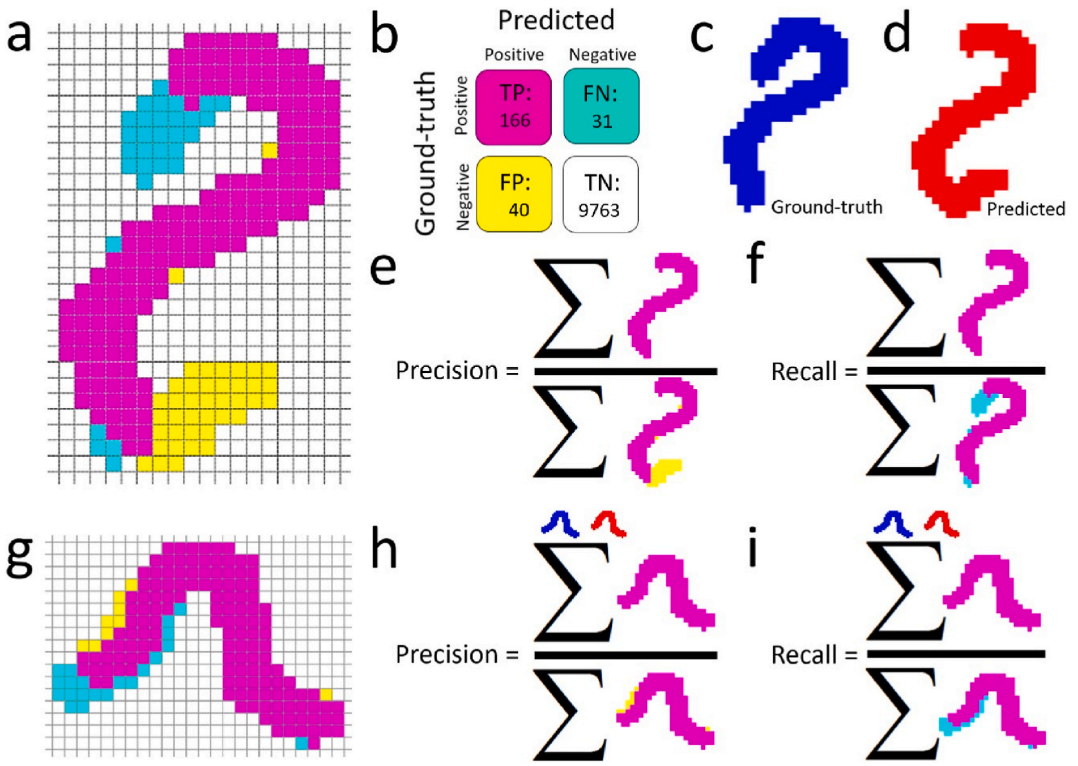

Fig. 6. Precision-Recall metrics. These metrics evaluated the results obtained with respect to the reference (ground-truth) to select the FP value used by the filter to eliminate possible solutions. (**a, g**) The *TP*, *FP*, FN and TN parameters used to calculate the precision and recall metrics, are shown in color (magenta, yellow, cyan and white, respectively). (**b**) *TP*, *FP*, FN and TN values for the example in Fig. 6a. (**c**) Reconstructed body of the manually labeled skeleton for the example in Fig. 6a. (**d**) Reconstructed body of the skeleton obtained from the optimizer response for the example in Fig. 6a. (**e, f, h, i**) Examples of the evaluation of precision and recall metrics of the reconstructed skeletons for the examples in Fig. 6a, g.

9900KF CPU @ 3.60 GHz ×16 with 32 GB of RAM and NVIDIA GeForce RTX 2080 Ti graphics card with 4352 Cuda cores, Ubuntu 19.04 64-bit operating system. The implementation was performed in a Python version 3.7.5 environment, using the Pytorch 1.18, OpenCV 4.5.4 and SWIG 3.2 libraries. The training and validation of the proposed neural network model took about 36 h with the aforementioned hardware and software. The hyper-parameters used for the training and validation phase were Batch_size = 10, num_workers = 10, maximum epoch = 100. The optimizer used was ADAM with a learning rate = 0.0001, betas = [0.95, 0.999], eps = 1e-8, BCELoss() as loss function for the training/validation step, and finally, the scheduler used was the ReduceLROnPlateau scheduler with hyperparameter mode = min and patience = 2.

To save the network trained during each epoch, the criterion was chosen if the average validation value was greater than the previous one, it was saved over the previous one, overwriting the saved network. The best average results were 0.001017 and

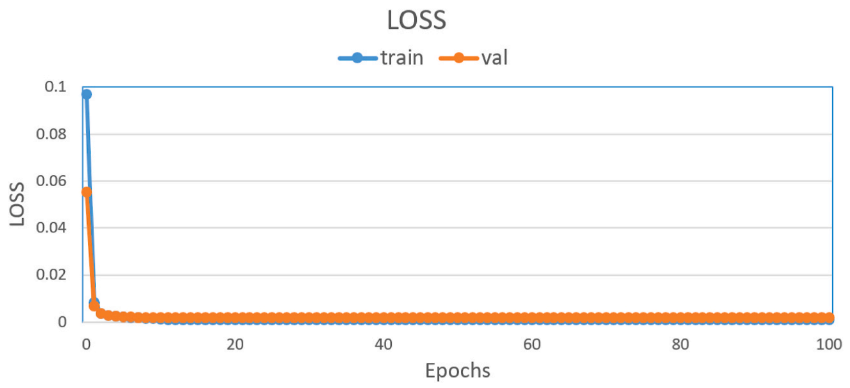

Fig. 7. Training/validation graph. This figure shows the average results (y-axis) of 5400 images for each epoch (x-axis). The training (blue line) and validation (orange line) results were obtained using the PyTorch BCELoss() function.

0.001935 for training and validation, respectively. The results obtained in each of the training and validation epochs are shown in Fig. 7.

For the real image dataset, 4500 Petri dish images (150 sequences of 30 images) were analyzed with subsequent selection of those difficult cases where worms aggregated with each other or aggregated with dish noise. To obtain greater variability of aggregations, image sequences with 10, 15, 30, 60 and 90 worms per plate were used. As mentioned in Refs. [26,27] the higher the number of worms, the higher the probability of aggregation events per plate. The total number of poses found was 788, 325 for cases of multi-worm aggregation and 463 aggregations with plate noise. We defined as plate noise all segmentation errors due to opaque debris and dark objects on the plate. Usually, these errors are dark with worm-like shapes, their grayscale is almost constant, so they are distinguished from worm bodies, which have different characteristics.

To obtain the best filter value (F), its value was varied from 0.00 to 1.00 in steps of 0.01. For each value, average precision and recall values were obtained for each skeleton for each prediction in multi-worm aggregations and aggregations with noise (788 aggregations). The selected filter value was the one whose precision and recall values were closest to 1. Table 1 shows the average precision and recall values obtained for a filter equal to 0 (predictions made without a filter) and with a filter equal to 0.65 (best precision and recall results values), the average precision and recall values for the cases of multi-worm aggregation and with plate noise are also shown.

To compare the proposed next-pose prediction method (next skeletons) with previous works [14], the filter of possible solutions to zero was used, which enabled comparison under the same conditions. The metric used to measure all the results was the IoU index, as in previous works. The average values without filter (filter equal to zero) are similar to the average values of previous works [14]. Table 2 shows the comparison of both methods (with filter and without filter), as well as all possible solutions and the filtered solutions with the best filter value obtained for all analyzed aggregation cases. As can be seen in the last column, since the filter reduces more than 75% of possible solutions Table 3, the optimizer only had a load of less than 25% (difference between possible solutions and filtered solutions). The results using the best filter value show an improvement over previous work [14], such as a considerable reduction in computational cost.

Statistical analysis was performed to compare the results obtained without the filter (previous work [14]) and using the filter of possible solutions (proposed work). The p-value (0.05) was used to evaluate statistical significance. Normality was first analyzed with the Kolmogorov-Smirnov Test to assess the difference between both methods. This test is used for large sample sizes ($n \geq 50$) and if the p-value was greater than or equal to the 5% level of significance (0.05), the null hypothesis H0 was accepted (the data came from a normal distribution), otherwise, the alternative hypothesis H1 was accepted (the data did not come from a normal distribution). The results indicated that they did not come from a normal distribution, p-value $= p < 0.001$, much lower than the significance value of 0.05 (Table 4); therefore, the alternative hypothesis H1 was accepted, and the Wilcoxon Signed Range Test was used (Table 5).

Wilconxon Signed Rank Test showed that the proposed method ($F = 0.65$) is statistically more significant than methods reported in previous works ($F = 0.00$), Table 5. The p-value obtained was 0.04, which is much lower than the significance value of 0.005.

Fig. 8 shows the average IoU values for both cases, as well as the mean (green line) and median (gray line) values using a box plot.

Finally, Fig. 9 shows an example of the results obtained for each worm in an aggregation. The proposed neural network model is used for each im-age series of each worm, Binary image worm1(t-1), gray image worm1,2(t-1) and gray image worm1,2(t) as network input for worm 1; and binary image worm2(t-1), gray image worm1,2(t-1), and gray image worm1,2(t) as input to the network for worm 2. Both network outputs enter the post-processing and the current worm skeletons in the aggregation are obtained: blue-cyan for worm1 and cyan-green for worm2. Other examples can be seen in:

https://github.com/playanaC/Skeleton_prediction/blob/main/Demo_videos.ipynb.

5. Discussion

The time required to load the input data (images) in the network at the time of training and validating the data was considered important. Accordingly, the number of worms per plate was considered to speed up this process. Each image in the synthetic dataset image sequence had only two worms, which speeded up the normalized 3×3 crop and filtering operations during the aforementioned stages. The 3×3 size is the minimum within the OpenCV tool and allowed to generalize both domains (synthetic and real). The width of the worms in our images varies from two to four pixels, so increasing the size of this filter can excessively blur the worm and even make it disappear. Another key factor was the size of the 100×100 window, these dimensions were selected to ensure that the next movement (skeleton) of a worm, no matter how fast, is within the window.

The proposed method allows tracking multiple worms and solving aggregations of any number of worms because it predicts each of them individually, which enabled tracking 2, 3, and 4-worm aggregations from the real image dataset, see Supplementary Figures 1, 2,

Table 1
Comparison of the average values of precision and recall metrics using a 0.65 filter and 0 filter. A total of 788 worm poses were used, of which 325 poses were multi-worm aggregations and 463 noise aggregation poses. The best results were obtained using the 0.65 filter.

Cases	Total pose	Avg. Precision		Avg. Recall	
		F = 0.00 F = 0.65	F = 0.65	F = 0.00	F = 0.65
Aggregation	325	0.7348	**0.7893**	0.7286	**0.7832**
Noise	463	**0.7892**	0.7877	**0.7573**	0.7562
Total	788	0.7649	**0.7884**	0.7445	**0.7683**

Table 2
Comparison of the average values using the IoU Index for 0.65 filter and 0 filter. This table shows the average values obtained using the IoU index with filter equal to 0.65 and without filter (IoU values similar to previous works [14]). The cases evaluated with the filter values were multi-worm aggregations (325) and aggregations with noise (463).

Cases	Total pose	Avg. IoU		Standard Deviation	
		F = 0.00	F = 0.65	F = 0.00	F = 0.65
Aggregation	325	0.6033	**0.6561**	0.2106	**0.1503**
Noise	463	**0.6507**	0.6504	**0.1856**	0.1876
Total	788	0.6295	**0.6529**	0.1985	**0.1718**

Table 3
The filtered solutions column shows the total possible solutions filtered using the coincidence filter. The difference between the possible solutions column and the filtered solutions column gives the total number of possible skeletons processed by the optimizer.

Cases	Total	Possible solutions	Filtered solutions
Aggregation	325	2505	1929
Noise	463	2552	1964
Total	788	5057	3893

Table 4
Normality test on the difference of the proposed method with filter at 0 (previous traditional methods) with filter equal to 0.65 (using the neural network). The p-value obtained was $p < 0.001$, 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. Kolmogorov Smirnov Test.

	Diff
N	788
Minimum	−0.7527
Maximum	0.7989
Mean	0.0234
Std. Deviation	0.1505
p-value	$p < 0.001$

Table 5
Wilcoxon Signed Rank Test. The Wilcoxon Signed Rank Test table shows the difference that exists in two related samples (F = 0.00 and F = 0.65) across positive, negative, and tied ranks. Wilcoxon Signed Rank Test.

	N		Mean rank	Sum ranks
Positive	321	*a*	326.3551	104760
Negative	297 [b]		291.2828	86511
Ties	170 [c]			
Total	788			

a. F = 0.65 > F = 0.00.
b. F = 0.65 < F = 0.00.
c. F = 0.65 = F = 0.00.

3. There are aggregation cases that are more complicated to solve than others, although in some instants of time there is a worm that is not perfectly matched with the ground-truth (see Supplementary Figure 1), the other worm presents better results, and as time progresses this error is corrected leaving the worms perfectly matched. On the other hand, in cases of aggregation with noise, the same happens during the time that they are aggregated with the noise of the plate (see Supplementary Figure 4), after this the pose prediction is more accurate. It should be noted that, although the accuracy of one of the worms is low, the identity is not lost.

As mentioned in the experiments and results sections, filter use (post-processing) reduces 77% of the possible solutions, which can translate into considerable computational savings. The total time to resolve an aggregation depends on the number of worms added, as well as potential aggregation complexity, which translates into many possible solutions. In a previous work [14], resolving these aggregations took several seconds, while in complex 4-worm aggregations, it took several minutes. The proposed method using the

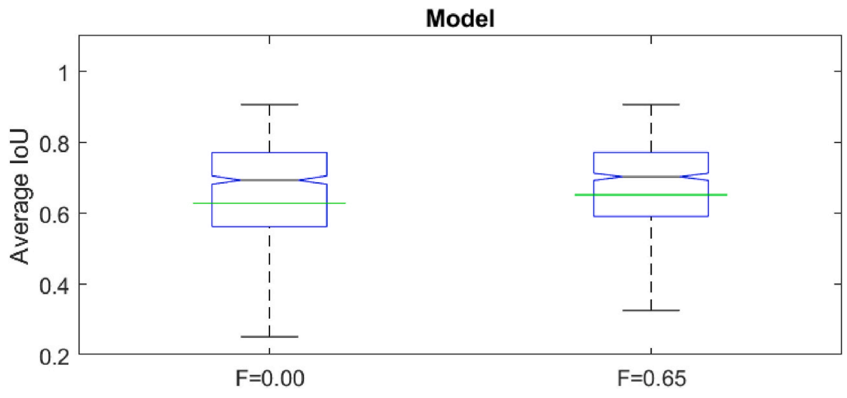

Fig. 8. Box plot and normality test of the difference of the 0 filter and 0.65 filter. Box plot, the green line indicates the mean in both graphs and the gray line indicates the median. Filter at 0.00, N = 788, mean = 0.6295, median = 0.6943, standard deviation = 0.1985, variance = 0.0394. Filter at 0.65, N = 788, mean = 0.6529, median = 0.7035, standard deviation = 0.1718, variance = 0.0295.

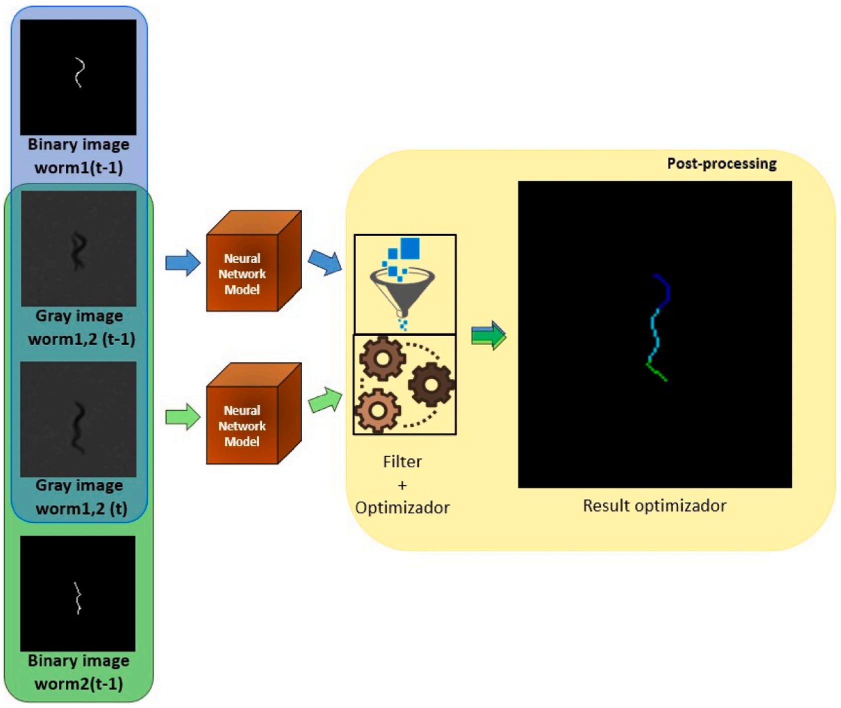

Fig. 9. Pipeline of a 2-worm aggregation. The trained network is used twice to obtain the prediction for the current skeleton of each worm (worm1, worm2). The results go through post-processing and skeletons are obtained for each worm. The current skeleton of worm1 is represented by the blue-cyan pixels, while the current skeleton of worm2 by the cyan-green pixels.

results of the neural network to filter possible solutions resolves 2-worm aggregations in less than 2 s, while it takes under 4.6 s to resolve 4-worm aggregations. Today there are many parallel processing or multiprocessing tools that can help speed up this processing time. Bearing this in mind and considering that the encoder and decoder can be replaced by similar models with fewer parameters, as demonstrated in Ref. [28], an online tracking application could be developed to solve cases of aggregation.

The prediction of the next postures or poses (skeleton) is really interesting in cases of aggregation since this helps to re-identify and not lose the identity of the subjects during the tracking. In this work, we have developed a technique to predict the next poses with results greater than 75% in precision and 0.60 with IoU values for multi-worm aggregations and aggregation with noise.

This technique could also be used for other types of aggregations using other small model organisms, such as bacteria cells, or zebrafish among others.

6. Conclusions

This paper presents a novel method for predicting *C. elegans* poses in low-resolution images of multiple worms in full 55 mm Petri dishes. The predicted poses are images of worm skeletons and are the result of a neural network trained to predict aggregation behaviors between worms. The proposed neural network was trained and validated with custom-generated synthetic images and successfully tested on real images. A multi-worm tracking application was developed, and for this, the results obtained from the neural network were used to filter possible solutions in an optimizer algorithm proposed in previous work [14]. The results were compared with this previous work [14] and an average improvement of 2.38% in precision and 2.34% with IoU values was obtained. On the other hand, the computational performance was also compared with previous work [14] and a greater than 75% reduction of the possible solutions generated by the optimizing algorithm was obtained, which is reflected in the computational costs. On the other hand, to determine the effectiveness of multiple worm tracking, we added the "MOTA" metric in the supplementary material. This metric measures the performance of the tracker, i.e., both the overall accuracy and the detection of objects in the images. We compared the results with previous work [14] and obtained an average improvement of 16.96%.

Despite very good results, the prediction of the next poses presents problems in cases of high aggregation, i.e. where more than 50% of the body of one of the individuals is aggregated with another or with noise from the plate, and the prediction does not improve until it starts to separate. Future work proposes to increase the datasets, better adjust the hyper-parameters, evaluate new architectures to improve the model results, as well as post-processing methods to obtain better pose predictions.

7. Source code

The proposed method was developed in Ubuntu-linux 19.04 64bits using python3.7.5 with the Pytorch1.5 libraries. The C++ and OpenCv4.5 libraries were used to create the synthetic images. To use the C++ functions (synthetic dataset) from Python we employed the SWIG3.2 library. The source code is on GitHub; it is open source and can be downloaded from the repository at https://github.com/playanaC/Skeleton_prediction. In this link there are a demo of the image simulator (synthetic dataset), demo of the proposed neural network method such as demo videos. The dataset with all the aggregation experiments can be downloaded from https://active-vision.ai2.upv.es/wp-content/uploads/2021/02/dataset_skeletons.zip. The imaging system, parts, and assembly can be downloaded from https://github.com/JCPuchalt/SiViS.

Author contribution statement

Pablo E. Layana Castro: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Antonio García Garví; Contributed reagents, materials, analysis tools or data. Antonio-José Sánchez-Salmerón: Conceived and designed the experiments; Analyzed and interpreted the data; Wrote the paper.

Funding statement

Prof. Antonio-José Sánchez-Salmerón; Pablo E. Layana Castro; Antonio García Garví were supported by Ministerio de Ciencia, Innovación y Universidades [RTI2018-094312-B-I00 (European FEDER funds); FPI PRE2019-088214; FPU20/02639]. This work was supported by Universitat Politècnica de València ["Funding for open access charge: Universitat Politècnica de València"].

Data availability statement

Data associated with this study has been deposited at https://github.com/playanaC/Skeleton_prediction.

Declaration of interest's statement

The authors declare no competing interests.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.heliyon.2023.e14715.

References

[1] D. Biron, G. Haspel (Eds.), C. Elegans, Springer Science+Business Media, New York, 2015, https://doi.org/10.1007/978-1-4939-2842-2.
[2] A. Olsen, M.S. Gill (Eds.), Ageing: Lessons from *C. elegans*, Springer International Publishing, Switzerland, 2017, https://doi.org/10.1007/978-3-319-44703-2.
[3] R. Sznitman, M. Gupta, G.D. Hager, P.E. Arratia, J. Sznitman, Multi- environment model estimation for motility analysis of *Caenorhabditis elegans*, PLoS One 5 (7) (2010), https://doi.org/10.1371/journal.pone.0011631.

[4] P.E. Layana Castro, J.C. Puchalt, A.-J. Sánchez-Salmerón, Improving skeleton algorithm for helping *Caenorhabditis elegans* trackers, Sci. Rep. 10 (1) (2020), 22247, https://doi.org/10.1038/s41598-020-79430-8.

[5] W.A. Boyd, G.L. Anderson, D.B. Dusenbery, P.L. Williams, Computer Tracking Method for Assessing Behavioral Changes in the Nematode Caenorhabditis Elegans, 1381, ASTM SPEC TECH PUBL, 2000, pp. 225–238, https://doi.org/10.1520/STP14426S.

[6] S.H. Simonetta, D.A. Golombek, An automated tracking system for *Caenorhabditis elegans* locomotor behavior and circadian studies application, J. Neurosci. Methods 161 (2) (2007) 273–280, https://doi.org/10.1016/j.jneumeth.2006.11.015.

[7] C. Restif, C. Ibañez-Ventoso, M. Driscoll, D. Metaxas, Tracking *C. elegans* swimming for high-throughput phenotyping, in: 2011 IEEE International Symposium on Biomedical Imaging: from Nano to Macro, 2011, pp. 1542–1548, https://doi.org/10.1109/ISBI.2011.5872695.

[8] D.B. Dusenbery, Using a microcomputer and video camera to simultaneously track 25 animals, Comput. Biol. Med. 15 (4) (1985) 169–175, https://doi.org/10.1016/0010-4825(85)90058-7.

[9] D. Ramot, B.E. Johnson, T.L. Berry Jr., L. Carnell, M.B. Goodman, The parallel worm tracker: a platform for measuring average speed and drug-induced paralysis in nematodes, PLoS One 3 (5) (2008) 1–7, https://doi.org/10.1371/journal.pone.0002208.

[10] N. Leonard, A.G. Vidal-Gadea, Affordable *Caenorhabditis elegans* Tracking System for Classroom Use, microPublication Biology 2021, 2021, https://doi.org/10.17912/MICROPUB.BIOLOGY.000377.

[11] P.B. Winter, R.M. Brielmann, N.P. Timkovich, H.T. Navarro, A. Teixeira-Castro, R.I. Morimoto, L.A. Amaral, A network approach to discerning the identities of *C. elegans* in a free moving population, Sci. Rep. 6 (1) (2016) 1–9, https://doi.org/10.1038/srep34859.

[12] A. Javer, M. Currie, C.W. Lee, J. Hokanson, K. Li, C.N. Martineau, E. Yemini, L.J. Grundy, C. Li, Q. Ch'ng, et al., An open-source platform for analyzing and sharing worm-behavior data, Nat. Methods 15 (9) (2018) 645–646, https://doi.org/10.1038/s41592-018-0112-1.

[13] M. Koopman, Q. Peter, R.I. Seinstra, M. Perni, M. Vendruscolo, C.M. Dobson, T.P. Knowles, E.A. Nollen, Assessing motor-related phenotypes of *Caenorhabditis elegans* with the wide field-of-view nematode tracking platform, Nat. Protoc. 15 (6) (2020) 2071–2106, https://doi.org/10.1038/s41596-020-0321-9.

[14] P.E. Layana Castro, J.C. Puchalt, A. García Garví, A.-J. Sánchez-Salmerón, *Caenorhabditis elegans* multi-tracker based on a modified skeleton algorithm, Sensors 21 (16) (2021), https://doi.org/10.3390/s21165622.

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

[16] L. Chen, M. Strauch, M. Daub, X. Jiang, M. Jansen, H.-G. Luigs, S. Schultz-Kuhlmann, S. Krüssel, D. Merhof, 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, 2020, pp. 508–512, https://doi.org/10.1109/ISBI45749.2020.9098465.

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

[18] L. Hebert, T. Ahamed, A.C. Costa, L. O'Shaughnessy, G.J. Stephens, Wormpose: image synthesis and convolutional networks for pose estimation in *C. elegans*, PLoS Comput. Biol. 17 (4) (2021) 1–20, https://doi.org/10.1371/journal.pcbi.1008914.

[19] T. Stiernagle, Maintenance of C. Elegans, 2006, https://doi.org/10.1895/wormbook.1.101.1. https://www.ncbi.nlm.nih.gov/books/NBK19649/?report=classic.

[20] J.C. Puchalt, A.-J. Sánchez-Salmerón, I. Eugenio, S. Llopis, R. Martínez, P. Martorell, Small flexible automated system for monitoring *Caenorhabditis elegans* lifespan based on active vision and image processing techniques, Sci. Rep. 11 (2021) 1–11, https://doi.org/10.1038/s41598-021-91898-6.

[21] J.C. Puchalt, J.F. Gonzalez-Rojo, A.P. Gómez-Escribano, R.P. Vázquez-Manrique, A.-J. Sánchez-Salmerón, Multiview motion tracking based on a cartesian robot to monitor *Caenorhabditis elegans* in standard petri dishes, Sci. Rep. 12 (1) (2022) 1–11, https://doi.org/10.1038/s41598-022-05823-6.

[22] M. Alexandre, Pytorch-unet, Code, 2019. https://github.com/milesial/Pytorch-UNet.

[23] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention 9351, Springer, Cham, 2015, pp. 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.

[24] S. Ioffe, C. Szegedy, 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, Vol. 37 of Proceedings of Machine Learning Research, PMLR, Lille, France, 2015, pp. 448–456.

[25] A. Koul, S. Ganju, M. Kasam, Practical Deep Learning for Cloud, Mobile, and Edge: Real-World AI & Computer-Vision Projects Using Python, Keras & TensorFlow, O'Reilly Media, 2019, pp. 679–680. https://books.google.es/books?id=GcS2DwAAQBAJ.

[26] N.A. Swierczek, A.C. Giles, C.H. Rankin, R.A. Kerr, High-throughput behavioral analysis in C. Elegans, Nat. Methods 8 (7) (2011) 592–598. 10.1038/nmeth.1625. https://www.nature.com/articles/nmeth.1625.

[27] J.C. Puchalt, A.-J. Sánchez-Salmerón, E. Ivorra, S. Genovés Martínez, R. Martínez, P. Martorell Guerola, Improving lifespan automation for *Caenorhabditis elegans* by using image processing and a post-processing adaptive data filter, Sci. Rep. 10 (1) (2020) 8729, https://doi.org/10.1038/s41598-020-65619-4.

[28] K. Trebing, T. Stanczyk, S. Mehrkanoon, Smaat-unet: precipitation nowcasting using a small attention-unet architecture, Pattern Recogn. Lett. 145 (2021) 178–186, https://doi.org/10.1016/j.patrec.2021.01.036.