# ML Project II: Worm tracking for personality detection

Guillem Bartrina I Moreno, Aitor Ganuza Izagirre, Eduardo Peña Royuela
*EPFL, Switzerland*

***Abstract* — The objective of this project is to track the position of a *C. elegans* in a low resolution microscope video. The impossibility to determine the position of the nematode from a single frame makes it necessary to extract temporal information. In this work, we model the pixels with Gaussian Mixture Models (GMM) for background substraction and apply image processing techniques to obtain candidate positions. Then, trackpy software [All+21] is used to perform enhanced localization and to link particle coordinates into trajectories. Finally, we select the most feasible trajectory with a simple heuristic. The obtained results are particularly noteworthy, and we believe that our method will impact on the research area of *C. elegans* personality detection.**

## I. Introduction

The present project was proposed by the Laboratory of the Physics of Biological Systems quantsysbio.com. The context of this project is the study of the personality of *C. elegans*, which is part of the doctoral thesis of our supervisor Matthieu Schmidt. Personality is defined as consistent behavioural differences among individuals, and keeping track of how the worms move is crucial to analyzing their behaviour.

In this report, we present a novel approach to tracking the position of a *C. elegans* in a microscope video. The video has a low resolution, making it difficult to see the worm with the human eye. Therefore, we rely on time information from the video to accurately track the *C. elegans*' position. The video was recorded from a zenithal perspective, showing the worm moving inside a Petri dish. We started this project from scratch, and we are not aware of any prior work on this topic.

In section II we delve into the dataset and explore its particularities. In section III we discuss multiple possible approaches and compare our problem setting with related scientific projects. Next, in section IV, we explain the pipeline of our proposed method, and discuss the results we obtain. Finally, in section V, we outline some guidelines for future improvements and conclude.

## II. Data Exploration

We have been given a dataset consisting of a video (divided into 3 parts to eliminate recording errors) and the position of the worm in each frame. The video is played at a speed of 50 fps and has a total length of 10 minutes and 35 seconds, giving a total of 31751 frames. Among those frames, only 30390 are annotated and there are frames with repeated annotations.

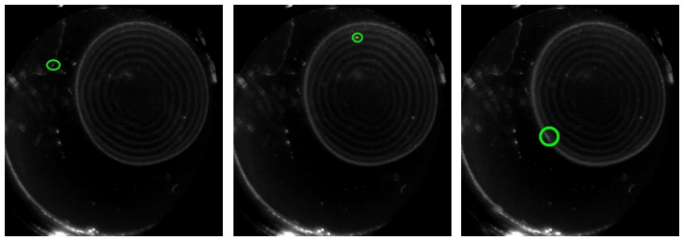

Fig. 1: Frames with a worm of different sizes

The video is low resolution and grayscale, utilizing only one color channel. As stated before, it shows a recording of a petri dish from a zenithal point of view, containing the worm as well as other elements such as a circle of bacteria, worm eggs, other unknown particles and light reflections on the petri dish. The video suffers from salt and pepper noise as well as a light tremor of the camera. The different elements of the video are displayed in Figure 1.

On top of the previous difficulties, the video presents more inconveniences. For one, there are lighting changes. During the video, the lighting of each frame changes and fluctuates, with a tendency for there to be more and more lighting in the scene. As a result, each particle's brightness changes frame by frame. In Figure 2 we can see the various brightness values attained by the worm, a pixel belonging to the bacteria circle and a pixel belonging to the background during the first part of the video. Due to the variety of values that the glow of the worm can take and their inconsistency, it is impossible to find a value that describes the worm and distinguishes it from the background.

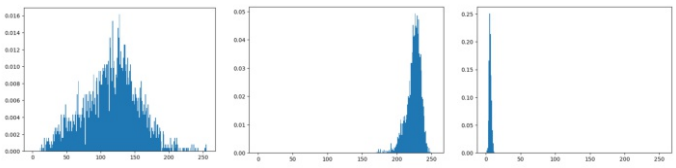

Fig. 2: Histogram brightness

Lighting issues extend even further. There are frames with excessive lighting wherein you can barely see anything. These

are few in number and could be considered outlier. Finally, different sides of the worm's body reflect light differently, and may not even reflect it, resulting in the worm not always being visible and appearing to "fade in and out".

Added to the above, the video records the worm from birth to adulthood, so that during the video the worm grows and increases its size. At the beginning of the recording the worm is so small that it is difficult to see even for the human eye, and temporal and movement information is necessary to distinguish it. Again, as in the case of lighting, the change in size makes it difficult to identify a size that describes the worm and distinguishes it from the background. In Figure 1 the worm is shown in various sizes that appear throughout the video.

Finally, it should be noted that in the case of wanting to train a machine learning model, the amount of data available for this project is rather scarce, even taking into account the possibility of performing data augmentation.

In summary, as we have exposed, the dataset presents several difficulties, mainly: the low resolution of the video, the noise of the image, the small size of the worm, the existence of worm-like particles, the change in illumination and the change in the worm size. This poses a challenge in the development of a model that makes it possible to obtain good results.

## III. Related Work and Possible Approaches

**Naive approach:** The first approach that comes to mind when dealing with images is to build from scratch a CNN tailored to the use case and proceed to train it, hoping to get some meaningful results. After many unsuccessful attempts, resulting from a mixture of our still immature expertise and lack of training data, we concluded that this approach was a dead end and began a thorough investigation of the literature.

Existing literature on the task of single small object tracking in a sequence of frames present different approaches:

### *A. Object detection*

These methods aim to locate, by means of bounding boxes or segmentation masks, and often classify, objects in images. Tracking is emulated by detecting the target object in each frame, so the temporal dimension of the data is practically unused.

The localization can be derived from features, motion or both. Popular feature-based methods include YOLO [Red+15] or Faster-RCNN [Ren+15]. These methods have not only shown poor results working with small objects, but also are hardly suitable for our scenario, provided that the worm 'features' are weak and not particularly informative. Exploratory testing of some of these methods did not yield any results. Subsequent iterations of testing making use of transfer learning also failed to yield results, due to the little training data available.

Motion-based methods consist mainly of background subtraction and are often robust and lightweight. A popular approach is to model the background using Gaussian distributions, which are then used to discriminate foreground pixels. ViBe [BV11; Yan+16] is a well-known method that applies this principle to a sequence of frames using a sliding window. [Testing of this method on our data showed fairly good results at discriminating the worm from the background. However, it very often classified other artifacts as foreground, which would have complicated subsequent steps in the pipeline.]

This principle of modeling the background using Gaussian distributions has been extensively expanded obtaining remarkable results, such as with the use of Kernel Density Estimators [HD12] or, mainly, Gaussian Mixture Models [GS18; SG00]. [Preliminary testing with various methods based on the latter yielded very promising results, and ended up begin the seed of our method.]

Other methods attempt to combine both features and motion to improve detection, and are typically based on CNNs. Some popular methods include ClusterNet+FoveaNet [LZS17] or T-Rex Net [Can+21]. These methods can prove effective, but they are computationally heavy, not widely available and tailored to very specific applications.

### *B. Object tracking*

These methods aim to keep track of objects in a sequence of frames based on its features, and heavily rely on initialization.

Common methods include the use of correlation filters [Hen+15], which attempt to localize the objects in consecutive frames, and some extensions to this principle [Xua+20; Du+18] that cope with usual limitations. [Exploratory testing with a couple of extensions on correlation filers did not yield very good results. They kept losing the worm as soon as it was not visible for a few frames or made a sudden movement.]

Such methods can be combined with object detection methods in order to avoid explicit initialization and gain in robustness. Instances of such ensembles are SORT [Bew+16] or DeepSORT [WBP17], but they use feature-based object detection.

### *C. Joint object detection and tracking*

These methods, which are mostly based on deep learning, learn to jointly detect and track objects, often by correlating learned features in time. Popular methods include ROLO [Nin+16], Track to Detect [FPZ17] or Tracktor++ [BML19]. Some are based on the popular attention mechanism, such as Patchwork [Cha19] or AiAiTrack [Gao+22]. Due to the fact these methods have been trained on very specific datasets, which hardly resemble our our data, their use would require retraining them (at least partially). And we have at hand very little data available.

### D. Other considerations

There are publicly available very few labeled datasets [Zhu+22] for the task of small object tracking in a sequence of frames, and our application is so niche that it has no significant representation in these datasets. One use of these datasets could be to train a deep learning model and then extract the learned features. But not only we have very little data to train the other model that makes use of the extracted features, but also we argued that in our scenario the features are not particularly informative.

There also exist some commercial [Bio] or open-source [BKS22; Ros] solutions specifically for worm tracking. We have not tried to adapt them, to our setting; however, even these may not be suitable due to the limiting properties of our data.

## IV. Our Method

We were provided with labels of the position of the worm in each frame, with the aim of performing supervised learning. However, as explained in section II, this data corresponds to a single video, and we concluded (after some testing) that it was not enough to train a supervised method, and that, it is very costly in terms of human time to label more data to even stand a chance. Therefore, we decided to use an unsupervised method.

Our method mainly relies on background subtraction techniques. However, a multistep pipeline is required for proper performance. First, there is a preprocessing step. Then, background removal is performed using GMMs for each of the pixels. The resulting image is then processed to remove noise and it is fed into the *trackpy* sub-pipeline, responsible for locating and linking particles. This outputs a set of particles that could possibly be the targeted *C. elegans*. With an heuristic based on the trajectories of these particles, we decide which of them is the inferred position.

### A. Preprocessing

The input video is a `.avi` file in gray-scale, so the first step is to convert the three RGB channels into one channel. This is simply done by taking the value of any of the channels, as the gray pixels have the three same RGB color space values.

Then, as the nematode does not come out of the Petri dish, we can safely remove the outside and borders of it. This is done by masking the video with a circle with manually adjustable center and radius.

### B. Background subtraction

Background subtraction is a widely used approach for detecting moving objects in videos from static cameras. Its goal is to make a segmentation between foreground (moving objects) and background, which is static. From a signal processing perspective, this can be interpreted as a high-pass filter in time.

We implemented two different methods for background subtraction. The **temporal average filter** is a naive approach that uses a sliding window in time of length $2T+1$. We model the background at frame $n$ as $bg[n] = \frac{1}{2T+1}\sum_{i=n-T}^{n+T} I[n+i]$, where $I[n]$ is the $n$'th frame of the video. Note that for the first and last $T$ frames, we simply do not compute anything. Then, $\tilde{I}[n] = |I[n] - bg[n]|$ is the new image at time $n$ without the background. Adjusting the hyperparameter $T$ is a compromise in the following sense. For greater $T$, we have a better (less noisy) model of the background (we remove most of the low-frequencies). However, it could happen that during the video the background changes; for example due to slow illumination variations or small camera displacements. Therefore, we would want the background of the model to be as *updated* or similar to the current background as possible, which is achieved reducing $T$.

Our best results were obtaining by modeling the background with **Gaussian mixture models**. In this context, a GMM is a probabilistic model that assumes each pixel $(x, y)$ to be generated from a mixture of a finite fixed number of $M$ Gaussian distributions. The implementation is from `Open-CV`, and it is based on [Zv06]. For now on, we fix the pixel position to $(x, y)$. The density is estimated using a reasonable time adaptation period $T$. At time $t$, the training data set is $\mathcal{V}_T = \{v_t, \ldots, v_{t-T}\}$, where $v_{\tilde{t}}$ indicates the value of the pixel $(x, y)$ at time $\tilde{t}$. At each new sample, we update $\mathcal{V}_T$ and reestimate the density. These samples might contain some values belonging to the foreground and some values belonging to the background. Hence, we denote the estimated density as $\hat{p}(v \mid \mathcal{V}, \text{BG + FG})$, and using a GMM:

$$\hat{p}(v \mid \mathcal{V}, \text{BG + FG}) = \sum_{m=1}^{M} \hat{\pi}_m \mathcal{N}(v, \hat{\mu}_m, \hat{\sigma}_m^2)$$

Note that we have a single channel, so the Gaussians are one-dimensional, and that we have the restrictions that $\hat{\pi}_m \geq 0$ and $\sum_{m=1}^{M} \hat{\pi}_m = 1$. Given a new data sample $w_t$, the parameters $\hat{\pi}_m, \hat{\mu}_m, \hat{\sigma}_m^2$, are updated using the following equations:

$$\hat{\pi}_m \leftarrow \hat{\pi}_m + \frac{1}{T}(o_m - \hat{\pi}_m)$$
$$\hat{\mu}_m \leftarrow \hat{\mu}_m + o_m \frac{\delta_m}{T\hat{\pi}_m}$$
$$\hat{\sigma}_m^2 \leftarrow \hat{\sigma}_m^2 + o_m \frac{\delta_m^2 - \hat{\sigma}_m^2}{T\hat{\pi}_m}$$

where $\delta_m = v_t - \hat{\mu}_m$, and $o_m$ indicates the *ownership* of the sample: it is equal to one if the sample is "close" (defined by Mahalanobis distance [Mah36]) to the Gaussian $m$, and 0 otherwise.

This is an on-line clustering algorithm. Here, foreground objects will be represented by some clusters with small weights $\hat{\pi}_m$. If an object remains static during some time, its associated $\hat{\pi}_m$ will become larger and eventually it will become part of

the background. Therefore, the background can be modeled by the largest $B$ mixtures:

$$\hat{p}(v \mid \mathcal{V}, \text{BG}) = \sum_{m=1}^{B} \hat{\pi}_m \mathcal{N}(v, \hat{\mu}_m, \hat{\sigma}_m^2)$$

Then, the decision that a pixel belongs to the background is made if $\hat{p}(v \mid \mathcal{V}, \text{BG}) > c_{thr}$, where $c_{thr}$ is a threshold value. Here, the hyperparameters are $B$ and $c_{thr}$, as the value for $M$ is selected by the provided algorithm.

### C. Noise removal

After the foreground-background segmentation, we use image processing techniques to de-noise the resulting binary image $\tilde{I}$, that has a value of 1 for foreground and 0 for background pixels. At this stage, the surrounding of the *C. elegans* position is expected to be cloud of foreground pixels. There are are some other foreground pixels in other parts of the image, which we want to eliminate. Therefore, we use a **morphologic operator** in the image. More precisely, we use an *opening*. This operation has two steps. First, it performs an *erosion*, which means that the value of each pixel is updated by the minimum value of its neighbors. Then, it performs a *dilation*, which does does the same but taking a maximum instead. In the first step small clouds of foreground disappear and the bigger cloud (corresponding to the worm) is reduced in size. In the second step, the bigger cloud returns to its previous size and denoised.

### D. Integration with trackpy

*Trackpy [CG96a] is a package for tracking blob-like features in video images, following them through time, and analyzing their trajectories. It is based on the widely-used Crocker–Grier particle tracking algorithm [CG96b], which separate tracking into three steps: feature finding (performing regional maximum selection of local brightness maxima), coordinate refinement (involving convolutions) and linking (based on a maximum likelihood estimation).*

The curated binary images resulting from the last step of the pipeline, which capture the best candidates of foreground pixels, are fed into trackpy's location algorithm, which outputs the refined coordinates of the particles found in each frame. This output is then fed into trackpy's linking algorithm, which matches the particles in time and builds their trajectories.

### E. Heuristics to make final prediction

Even if they are few, trackpy locates and links into trajectories more particles than the worm, such as the eggs or other interferences. For that, we apply a custom heuristic on the trajectories in order to determine which ones correspond to the worm. Since we have available the tracking information of the entire video, this task is not very complex. The current heuristic is simple but effective: in each frame, the particle that corresponds to the worm (if any) is the particle that has the longest lasting trajectory.

### F. Results

In order to show the results, our framework outputs the video with a green dot that follows the form. Also, as were provided with a file of ground truth labels for some frames, we created a custom metric to evaluate our accuracy. At each labeled frame $t$, our accuracy is:

$$\text{Acc}(t) = \begin{cases} 1 & \text{if } \|(x,y) - (\hat{x},\hat{y})\| \leq r \\ 0.5 & \text{if } r < \|(x,y) - (\hat{x},\hat{y})\| \leq R \\ 0 & \text{if } \|(x,y) - (\hat{x},\hat{y})\| > R \end{cases}$$

We chose the $\ell_\infty$ norm (here balls are squares) for its computational simplicity, but we could also use the 2-norm (where balls are circles) and obtain similar results. We set $r = 7$ and $R = 10$ pixels. We define the custom accuracy as the mean of the accuracies for all labelled $t$.

Our method achieves an average (custom) accuracy of $96.7\%$.

## V. Conclusions and Future Work

As we have been able to see, our method solves the problem posed by our supervisor in a very satisfactory way.

We want to emphasize that we have faced this project without any type of technical guidance or knowledge beyond that obtained in the Machine Learning course, and that we have invested a significant amount of time in this work doing research and testing different techniques and methods.

In addition to developing our method, in parallel, we have also implemented software for easy manipulation of the videos and high-level transformations, both to undertake different tests with our method and to use it in future improvements.

Finally, our results, while good, still have room for improvement. The biggest problem with our method is that the heuristic for differentiating the worm trajectory from the trajectories of other particles is very simple. The result is that in videos where the worm lays its own eggs, our method confuses eggs with the worm. We can think of several ways to solve this: improve the heuristics, develop a Deep Learning model that knows how to distinguish between trajectories and/or a Deep Learning model capable of recognizing if the pixel chosen at each frame is part of the worm or not. That would be how we could improve our project, which we would have tried if we had had the time.

All in all, the project has been a success as we have met the lab's requests and our contribution will be used in the personality study of *C. elegans*.

## References


[Mah36] Prasanta Chandra Mahalanobis. "On the generalized distance in statistics". In: *Proceedings of the National Institute of Sciences (Calcutta)* 2 (1936), pp. 49–55.

[CG96a] John C. Crocker and David G. Grier. "Methods of Digital Video Microscopy for Colloidal Studies". In: *Journal of Colloid and Interface Science* 179.1 (1996), pp. 298–310. ISSN: 0021-9797. DOI: https://doi.org/10.1006/jcis.1996.0217. URL: https://www.sciencedirect.com/science/article/pii/S0021979796902179.

[CG96b] John C. Crocker and David G. Grier. "Methods of Digital Video Microscopy for Colloidal Studies". In: *Journal of Colloid and Interface Science* 179.1 (1996), pp. 298–310. ISSN: 0021-9797. DOI: https://doi.org/10.1006/jcis.1996.0217. URL: https://www.sciencedirect.com/science/article/pii/S0021979796902179.

[SG00] C. Stauffer and W.E.L. Grimson. "Learning patterns of activity using real-time tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 747–757. DOI: 10.1109/34.868677.

[Zv06] Zoran Zivkovic and Ferdinand van der Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction". In: *Pattern Recognition Letters* 27.7 (2006), pp. 773–780. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0167865505003521.

[BV11] Olivier Barnich and Marc Van Droogenbroeck. "ViBe: A Universal Background Subtraction Algorithm for Video Sequences". In: *IEEE Transactions on Image Processing* 20.6 (2011), pp. 1709–1724. DOI: 10.1109/TIP.2010.2101613.

[HD12] Bohyung Han and Larry S. Davis. "Density-Based Multifeature Background Subtraction with Support Vector Machine". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5 (2012), pp. 1017–1023. DOI: 10.1109/TPAMI.2011.243.

[Hen+15] Joao F. Henriques et al. "High-Speed Tracking with Kernelized Correlation Filters". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.3 (Mar. 2015), pp. 583–596. DOI: 10.1109/tpami.2014.2345390. URL: https://doi.org/10.1109%5C%2Ftpami.2014.2345390.

[Red+15] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. DOI: 10.48550/ARXIV.1506.02640. URL: https://arxiv.org/abs/1506.02640.

[Ren+15] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. DOI: 10.48550/ARXIV.1506.01497. URL: https://arxiv.org/abs/1506.01497.

[Bew+16] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2016. DOI: 10.1109/icip.2016.7533003. URL: https://doi.org/10.1109%5C%2Ficip.2016.7533003.

[Nin+16] Guanghan Ning et al. *Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking*. 2016. DOI: 10.48550/ARXIV.1607.05781. URL: https://arxiv.org/abs/1607.05781.

[Yan+16] Yun Yang et al. "An improved ViBe for video moving object detection based on evidential reasoning". In: *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. 2016, pp. 26–31. DOI: 10.1109/MFI.2016.7849462.

[FPZ17] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. *Detect to Track and Track to Detect*. 2017. DOI: 10.48550/ARXIV.1710.03958. URL: https://arxiv.org/abs/1710.03958.

[LZS17] Rodney LaLonde, Dong Zhang, and Mubarak Shah. *ClusterNet: Detecting Small Objects in Large Scenes by Exploiting Spatio-Temporal Information*. 2017. DOI: 10.48550/ARXIV.1704.02694. URL: https://arxiv.org/abs/1704.02694.

[WBP17] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. *Simple Online and Realtime Tracking with a Deep Association Metric*. 2017. DOI: 10.48550/ARXIV.1703.07402. URL: https://arxiv.org/abs/1703.07402.

[Du+18] Bo Du et al. "Object Tracking in Satellite Videos by Fusing the Kernel Correlation Filter and the Three-Frame-Difference Algorithm". In: *IEEE Geoscience and Remote Sensing Letters* 15.2 (2018), pp. 168–172. DOI: 10.1109/LGRS.2017.2776899.

[GS18] Kalpana Goyal and Jyoti Singhai. "Review of background subtraction methods using Gaussian mixture model for video surveillance systems". In: *Artificial Intelligence Review* 50.2 (Aug. 2018), pp. 241–259. ISSN: 1573-7462. DOI: 10.1007/s10462-017-9542-x. URL: https://doi.org/10.1007/s10462-017-9542-x.

[BML19] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. "Tracking Without Bells and Whistles". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00103. URL: https://doi.org/10.1109%5C%2Ficcv.2019.00103.

[Cha19] Yuning Chai. *Patchwork: A Patch-wise Attention Network for Efficient Object Detection and Segmentation in Video Streams*. 2019. DOI: 10.48550/

ARXIV.1904.01784. URL: https://arxiv.org/abs/1904.01784.

[Xua+20] Shiyu Xuan et al. “Object Tracking in Satellite Videos by Improved Correlation Filters With Motion Estimations”. In: *IEEE Transactions on Geoscience and Remote Sensing* 58.2 (2020), pp. 1074–1086. DOI: 10.1109/TGRS.2019.2943366.

[All+21] Daniel B. Allan et al. *soft-matter/trackpy: Trackpy v0.5.0*. Version v0.5.0. Apr. 2021. DOI: 10.5281/zenodo.4682814. URL: https://doi.org/10.5281/zenodo.4682814.

[Can+21] Alessio Canepa et al. “T-RexNet-A Hardware-Aware Neural Network for Real-Time Detection of Small Moving Objects”. en. In: *Sensors (Basel)* 21.4 (Feb. 2021).

[BKS22] Shoubhik Chandan Banerjee, Khursheed Ahmad Khan, and Rati Sharma. “Deep-Worm-Tracker: Deep Learning Methods for Accurate Detection and Tracking for Behavioral Studies in C. elegans”. In: *bioRxiv* (2022). DOI: 10.1101/2022.08.18.504475. eprint: https://www.biorxiv.org/content/early/2022/08/19/2022.08.18.504475.full.pdf. URL: https://www.biorxiv.org/content/early/2022/08/19/2022.08.18.504475.

[Gao+22] Shenyuan Gao et al. *AiATrack: Attention in Attention for Transformer Visual Tracking*. 2022. DOI: 10.48550/ARXIV.2207.09603. URL: https://arxiv.org/abs/2207.09603.

[Zhu+22] Yabin Zhu et al. *Tiny Object Tracking: A Large-scale Dataset and A Baseline*. 2022. DOI: 10.48550/ARXIV.2202.05659. URL: https://arxiv.org/abs/2202.05659.

[Bio] MBF Bioscience. *Worm Tracking — mbfbioscience.com*. https://www.mbfbioscience.com/worm-tracking. [Accessed 16-Dec-2022].

[Ros] Sheryl Rosen. *Track-A-Worm — C. elegans Neurobiology Lab — health.uconn.edu*. https://health.uconn.edu/worm-lab/track-a-worm/. [Accessed 16-Dec-2022].