# An improved neural network model enables worm tracking in challenging conditions and increases signal-to-noise ratio in phenotypic screens

Weheliye H Weheliye[1,2], Javier Rodriguez[1,2], Luigi Feriani[1,2,*], Avelino Javer[1,2,#], Virginie Uhlmann[3], André EX Brown[1,2]

[1]MRC Laboratory of Medical Sciences, London, UK
[2]Institute of Clinical Sciences, Imperial College London, London, UK
[3]European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Cambridge, UK
*Present address: Australian Synchrotron - ANSTO, Melbourne, Australia
[#]Apheris, Berlin, Germany
Correspondence to: andre.brown@imperial.ac.uk

## Abstract

High-resolution posture tracking of *C. elegans* has applications in genetics, neuroscience, and drug screening. While classic methods can reliably track isolated worms on uniform backgrounds, they fail when worms overlap, coil, or move in complex environments. Model-based tracking and deep learning approaches have addressed these issues to an extent, but there is still significant room for improvement in tracking crawling worms. Here we train a version of the DeepTangle algorithm developed for swimming worms using a combination of data derived from Tierpsy tracker and hand-annotated data for more difficult cases. DeepTangleCrawl (DTC) outperforms existing methods, reducing failure rates and producing more continuous, gap-free worm trajectories that are less likely to be interrupted by collisions between worms or self-intersecting postures (coils). We show that DTC enables the analysis of previously inaccessible behaviours and increases the signal-to-noise ratio in phenotypic screens, even for data that was specifically collected to be compatible with legacy trackers including low worm density and thin bacterial lawns. DTC broadens the applicability of high-throughput worm imaging to more complex behaviours that involve worm-worm interactions and more naturalistic environments including thicker bacterial lawns.

## Introduction

High-resolution tracking that measures both position and posture has long been applied to the nematode *C. elegans*[1–12] and has enabled applications in genetics, neuroscience, and drug screening among others. One reason that tracking postural details in *C. elegans* has been accessible since the turn of the century is that worms have a simple body morphology and can be imaged with high contrast while confined to two dimensions on the surface of an agar plate. High contrast imaging of sparsely distributed animals with a simple morphology is well suited to classic computer vision approaches for segmentation and skeletonisation. Despite these successes in worm tracking, most methods developed for pose estimation were brittle and simple overlaps between animals or self-intersection during coiling remained challenging for longer. Alternative methods were developed for pose estimation in overlapping worms[13,14] or individual coiling worms[15,16] but most of these methods are computationally intensive and less appropriate for high-throughput screening with large numbers of worms.

More recently, advances in deep learning have extended the applicability of pose estimation to more challenging applications including non-uniform backgrounds and animals with complex body plans[17–19]. Several instance segmentation methods have been applied to or developed for segmenting worms specifically[20–22]. A model called DeepTangle was recently developed to track swimming worms which takes into account temporal information and explicitly learns a worm shape model to produce skeletons directly from videos[23]. Swimming worms adopt simpler lower-dimensional postures than crawling worms, rarely coil, and tend to have simpler overlaps. Here we extend DeepTangle to the more complex postures and long overlaps of crawling worms using a new training dataset and show that the new model, DeepTangleCrawl (DTC), can track coiling and overlapping worms, achieving a new state-of-the-art on challenging worm tracking data compared to two instance segmentation approaches. The resulting data is accurate, and tracks are longer and more complete (fewer gaps). The improved tracking enables the analysis of behaviour in previously inaccessible conditions and improves the signal-to-noise ratio in phenotypic screens.

## Results

### *Data and model training*

All the data we used for training and evaluation were collected using megapixel camera arrays with a resolution of 12.4 µm/pixel and a frame rate of 25 frames/second[24]. We selected subsets of data from diverse experiments including data from disease models[25], worms treated with pharmaceuticals[24], and worms treated with insecticides and nematicides[26]. The

data were collected by different operators over several years. The goal was to have a training dataset that included variation in worm behaviour and morphology as well as technical variation, day-to-day variation, and seasonal variation. The current training data does not include data from different imaging setups or different labs.

The original DeepTangle model was trained on synthetic data. Simulating swimming worms is simpler than crawling worms for biological and technical reasons. First, worm behaviour during swimming is simpler than during crawling[5,27,28]. Second, imaging worms in liquid tends to result in a more uniform background than worms crawling on the surface of an agar plate seeded with a lawn of bacteria. Third, the challenging tracking examples we are most interested in solving—such as worms in thick bacterial lawns or tightly overlapping worms with correlated locomotion and an interface that is complicated optically—are the cases where producing reliable synthetic images would be the most challenging. Given these considerations, we constructed our training data from recordings of actual worms.

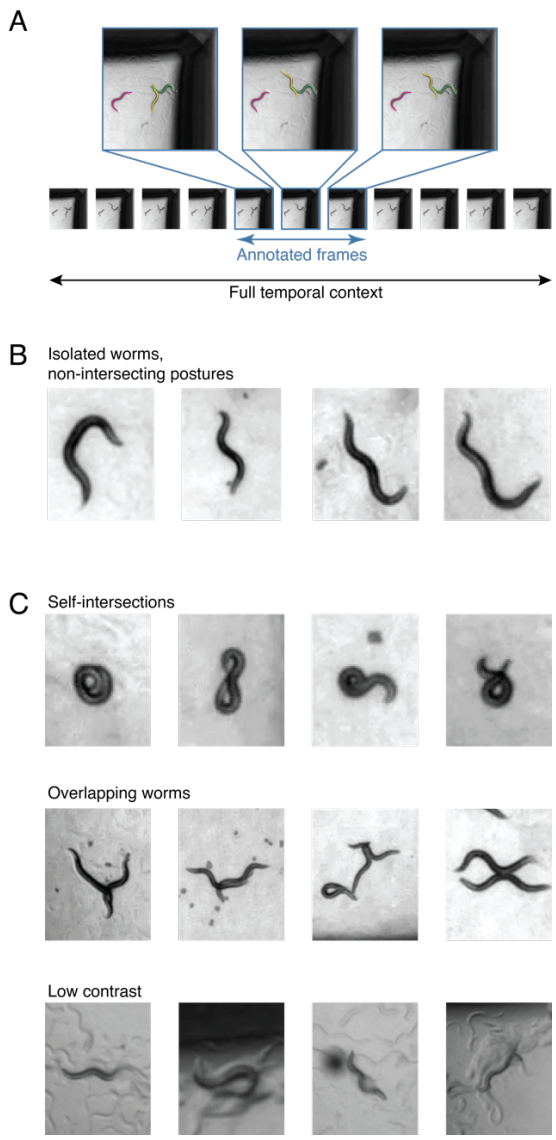


**Figure 1: Training data is a mix of straightforward and challenging cases.** (A) DeepTangle's input consists of clips with central frames annotated. Other models were simply trained on individual annotated frames. (B) Isolated non-intersecting worms can be tracked using Tierpsy's existing algorithm. (C) There are several categories of more challenging cases where simple skeletonization algorithms fail including self-intersecting worms, multiple overlapping worms, and worms with complex non-uniform backgrounds. Clips for these cases were manually annotated.

DeepTangle uses information from adjacent frames to improve segmentation. Therefore, all annotations were done on short clips consisting of 11 frames with all worms in the middle three frames annotated manually (Figure 1A). All 11 frames from each clip are input to the model during training. We first compiled an extensive training data set consisting

of non-coiling non-overlapping worms that were accurately tracked using our baseline algorithm Tierpsy (Figure 1B). We then created a synthetic dataset by overlaying pre-segmented and skeletonised worms with random orientations to create training data with overlaps. These simple cases and synthetic overlaps were then supplemented with manual annotations from more challenging cases (Figure 1C). Most instance segmentation methods require only single annotated frames. To train these models we simply used the annotated frames individually.

Compared to the original DeepTangle model, we increased the dimension of the late space representing worm shapes from 12 to 72. Because crawling behaviour has diverse timescales ranging from fast locomotion to long periods of dwelling, the 11-frame clips in the training data were uniformly sampled from 2, 4, and 8 second recordings so that the model could learn postural changes across several time scales.

Before both training and inference, we subtracted the background from recordings using singular value decomposition. Specifically, we performed SVD on videos that were subsampled 1:400 in time and used the single highest energy mode as a background image that was then subtracted from all video frames. This first mode consistently corresponds to the background without any worms. This step made all of the data, including the synthetic overlapping worm clips, directly comparable without a background outside of the worms.

### *Pose estimation accuracy*

We evaluated DTC on a set of held out data that were sampled across the same experiments as the training data but that were not seen by any model during training. The median root mean squared deviation between the predicted pose and the manually annotated pose (or Tierpsy pose in the case of simple non-overlapping shapes) is 2.2 pixels (Figure 2A). At the resolution used here, this corresponds to an RMSD of approximately half a worm body width. To compare the performance of DTC to the current state of the art, we used the same data to train Omnipose and a landmark-based tracker similar to SLEAP[29] we refer to here as PAF for part affinity field[30]. PAF and Omnipose have a lower modal RMSD than DTC for the cases where a prediction is made, and all of the models have a lower modal RMSD than the inter-annotator RMSD of greater than 4 pixels. However, there are also cases where no prediction is made and this failure rate is different across models (Figure 2B). On this measure, DTC performs the best, making no predictions less frequently than other models.

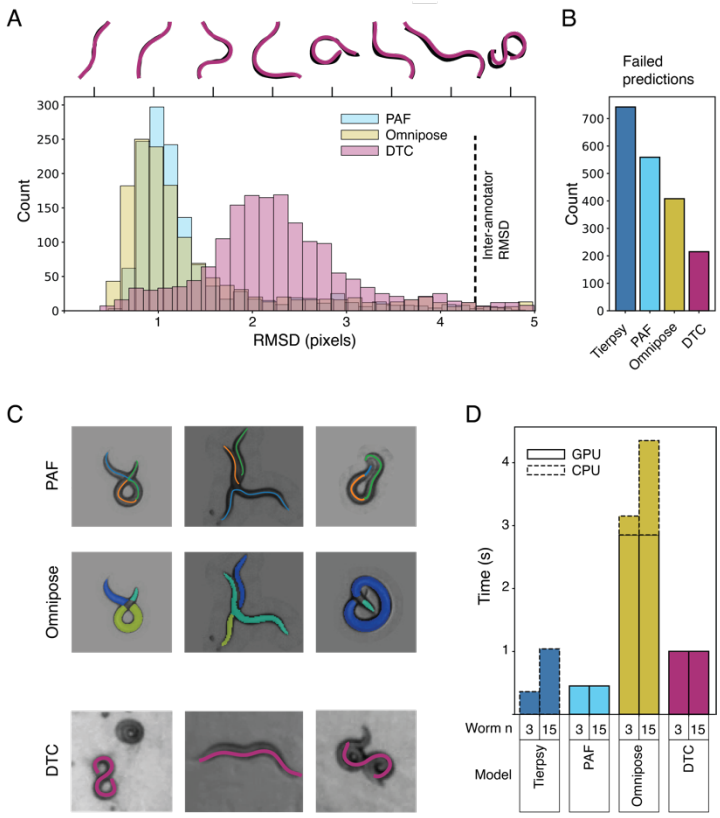


**Figure 2: Pose estimation accuracy.** (A) Histograms of root mean square deviation between held-out skeletons and predicted skeletons for each tested model. The vertical dashed line shows the RMSD between three manual annotators. Skeletons above the histogram are examples that illustrate the corresponding RMSD visually. (B) Bar chart showing the number of cases in the held-out test data where a model fails to make a prediction (e.g. Tierpsy fails on coiled worms or a neural network model does not identify any worms above a confidence threshold). (C) Examples of the kinds of errors each model makes. PAF and Omnipose often over-segment worms. DTC fails when worms are persistently tightly coiled or in tight parallel contact for an extended time. (D) Computation time per input frame for the different models as a function of worm number/well. Each camera records 16 wells in a 96-well plate so these correspond to 48 and 240 worms per video. Tierpsy only uses CPU computation while Omnipose uses GPU and CPU because we use Tierpsy's skeletonization algorithm to convert segmented regions to skeletons.

Not only are the rates of failure different, the types of cases each model fails on are different. For example, Omnipose takes worm width as a parameter and is good at separating parallel worms that are in contact along their length but tends to over-segment coiled worms or crossing worms (Figure 2C, top row). PAF makes some similar errors on coiled and crossing worms and struggles to separate parallel worms (Figure 2C, middle row). DTC performs better on coiled and crossing worms, most likely because it is able to take earlier and later frames into account to resolve the most likely poses. DTC does still fail on worms that form long-lasting overlaps or tight coils with little motion (Figure 2C, bottom row). Because the worms do not move, DTC is unable to take advantage of temporal information to resolve the coiled shape. When these slow-moving coilers do straighten within a clip available to DTC, it is often able to estimate the pose in the coiled frames. All models currently fail in cases of complex overlaps with multiple worms (Figure 2C, bottom row, right).

In addition to their accuracy, computation time is an important metric to consider for any model that will be used for large scale phenotypic screens. We compared the inference time for each model separated into CPU and GPU time (Figure 2D). We only included the time to generate skeletons, not the time to connect them into tracks because the tracking

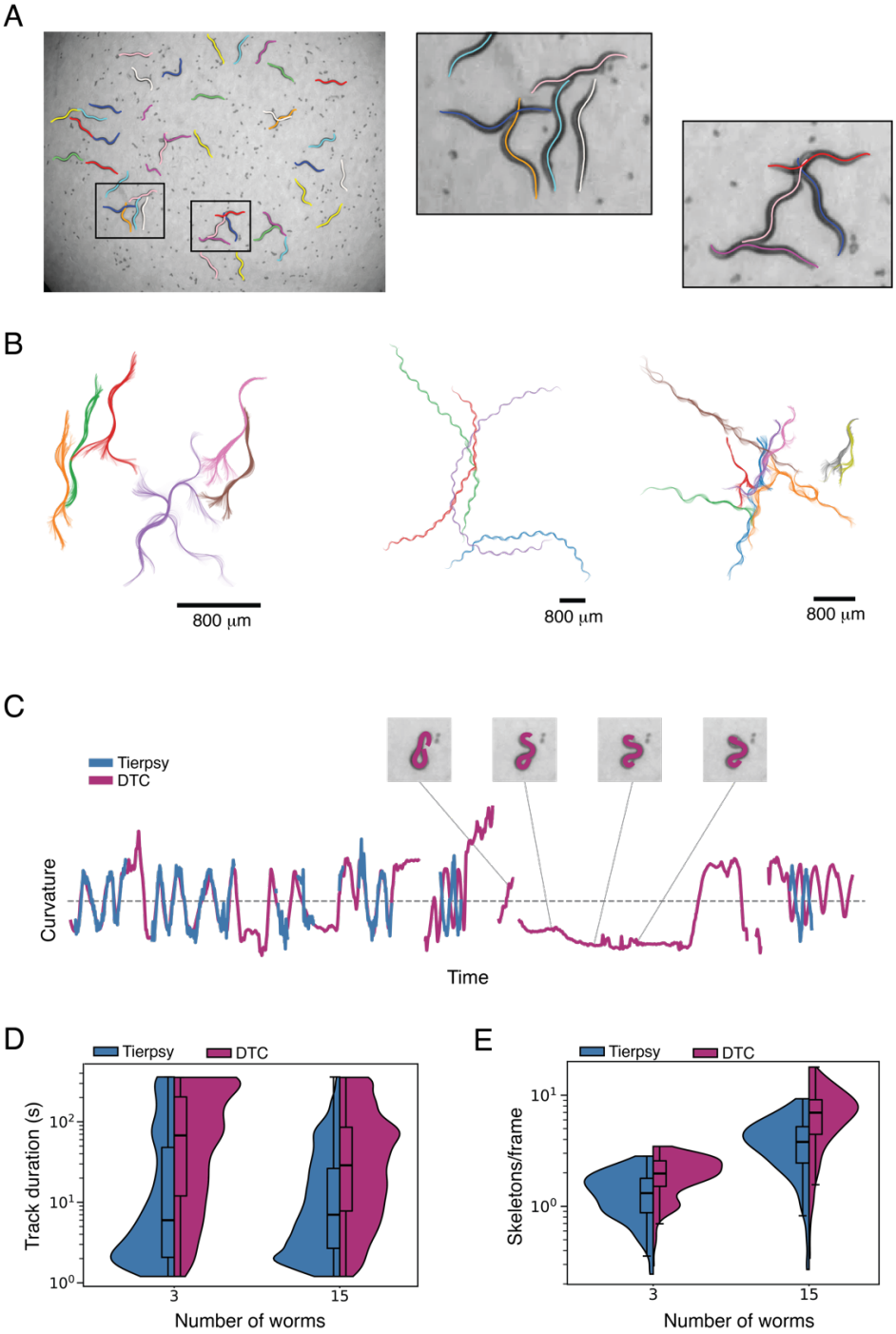


**Figure 3: Tracking in challenging conditions.** (A) Multiple overlapping worms with high density of eggs. Inset images show higher magnification images of two sets of overlapping worms where all individuals are successfully skeletonised. (B) Examples of continuous tracks that preserve worm identify during and through collisions. (C) For highly curved worms that form long-lived coiling shapes, long gaps in the data can be present in Tierpsy-derived data. Here, there is a long gap with a high curvature that is recovered using DTC. (D) Improved skeletonization leads to longer tracks from DTC compared to Tierpsy. Note the log scale. The duration is longer for videos with 3 and 15 worms per well. (E) The number of skeletons/frame averaged over a video. DTC tracking produces numbers closer to the nominal number of worms per well.

algorithm could be varied and is not a core feature of each approach. The baseline model Tierpsy is run only on CPUs while PAF and DTC run only on GPUs. Omnipose outputs a

mask for each worm and so requires a second skeletonization step. Here we apply Tierpsy's skeletonization algorithm and do this step on CPUs. The neural network methods process all pixels and so processing time does not depend on the number of worms. Tierpsy skeletonises each segmented object separately and so processing time increases with worm number. To speed up inference in cases where there are a small number of worms, we down-sampled videos 1:3 in time and then used a three-dimensional smoothing spline to interpolate the missing data. This reduced total computation time with only a small decrease in performance, but the time advantage is modest with 15 worms per well (Figure S1).

Given its state-of-the-art performance in number of worms skeletonised and its acceptable accuracy and computation time, we focus on characterising DTC for the remainder of the paper. The results below were calculated using the DTC+spline algorithm but the results are broadly similar with DTC applied to each frame.

### *DTC improves tracking in challenging conditions*

The data presented so far were collected in conditions with relatively sparse worms on thin food lawns that were optimised for tracking using Tierpsy. Tierpsy is more sensitive than DTC to thicker food lawns, the presence of eggs or other background objects, as well as more frequent overlaps or coiling. The difference in performance in more challenging conditions is therefore larger. When there is a higher density of worms and eggs on the lawn, DTC is still able to skeletonise the majority of worms accurately (Figure 3A and Supplementary Movie 1) whereas Tierpsy frequently fails (Supplementary Movie 2). Because skeletonization is maintained for overlapping worms, DTC is also able to accurately maintain worm identity through collisions (Figure 3B).

When worms perform sharp turns, their self-intersecting poses cannot be resolved by Tierpsy, leading to gaps in data. For strong coiler mutants or worms treated with drugs that cause coiling, these gaps can be long and the poses that are not resolved are precisely those that are most characteristic so missing them limits accurate phenotyping. DTC is often able to resolve these shapes if there is some worm motion (Figure 3C and Supplementary Movie 3) although tight coils with little to no motion remain challenging, as mentioned previously (Figure 2C). Quantitatively, we see an increase in both trajectory duration and the number of skeletons per frame between Tierpsy and DTC for both 3 and 15 worms per well (Figure 3D).

### *Improved signal to noise ratio in phenotypic screens*

170 To determine whether improved tracking quality leads to improved sensitivity in the context
171 of a large phenotypic screen, we reanalysed a previously published dataset consisting of
172 videos of worms treated with diverse insecticides and nematicides[26]. Using Tierpsy tracker,
173 we have previously shown that we could detect the effects of insecticides and also predict
174 their mode of action based only on their behavioural effects. As noted in the previous section,
175 some of these treatments led to worms that adopted self-intersecting postures that were not
176 tracked by Tierpsy leading to significant missing data. We extracted the same behavioural
177 features from videos tracked using Tierpsy and DTC.

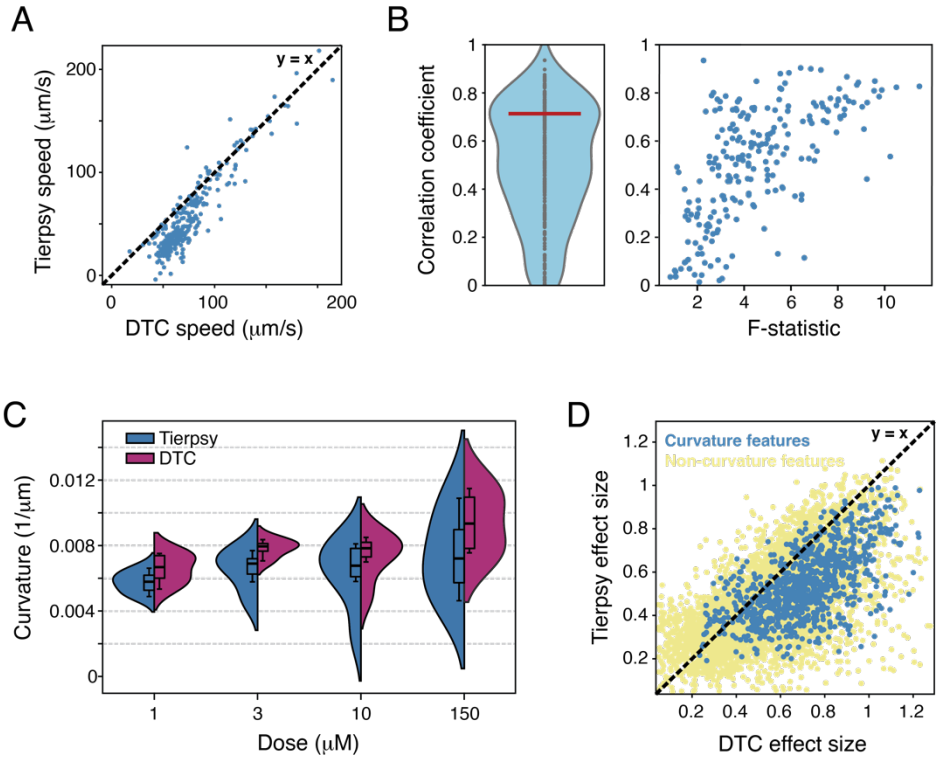


**Figure 4: DTC improves the signal to noise ratio in a phenotypic screen. All data from ref. [26].** (A) Speed calculated from Tierpsy skeletons and from DTC skeletons for a random sample of 290 wells from a previously published drug screen. The dashed line is y = x. (B) Correlation coefficients for each of the Tierpsy 256 behavioural features for the data from ref. [26] (left). The red line indicates the modal value of 0.71. Correlation coefficient plotted against the F-statistic for each feature calculated over the entire dataset. (C) Tail curvature as a function of dose for worms treated with a spiroindoline known to cause coiling. (D) A comparison of the mean absolute value of Hedge's d effect size calculated using features derived from Tierpsy and DTC tracking data. Each point is the mean Hedge's d across all doses of a drug compared to DMSO controls for a single feature. Any feature with 'curvature' in the name except for time derivatives of curvature are shown in blue. All other features are shown in yellow.

178 As expected for relatively clean data that was collected specifically to work with
179 Tierpsy, the results are broadly comparable using both tracking methods. For example, the
180 median midbody speed calculated using skeletons from Tierpsy is highly correlated to the
181 speed calculated using skeletons from DTC (Figure 4A). Looking across the Tierpsy256
182 features, the modal correlation coefficient is 0.71 (Figure 4B, left). Some features show little

correlation between the two tracking methods. Many of these poorly correlated features also have low F-statistic calculated across drug classes suggesting that they are simply noisy features in this dataset (Figure 4B, right).

We next focussed specifically on a spiroindoline drug (SY1713) that causes strong coiling where we expect a difference between Tierpsy and DTC. For each dose, we find a higher curvature estimate from DTC than for Tierpsy, consistent with the expectation that DTC skeletonises more frames with coiling (Figure 3C). The distribution of curvature values is also narrower for DTC-calculated curvatures, corresponding to a larger signal-to-noise ratio comparing drug treatments to DMSO controls for feature data derived from DTC tracking. Based on this observation we quantified the effect size for all features and all samples using features derived from Tierpsy and DTC tracking (Figure 4D). Effect sizes are correlated but are larger (below the diagonal in Figure 4D) for DTC-derived features.

## Discussion

DTC tracks more worms in challenging conditions including complex backgrounds and overlaps than the other tested models. It also has acceptable accuracy, with an RMSD that is better than the variation between human annotators, although the current version is less accurate than Omnipose and PAF for the cases where these models make any prediction. DTC therefore opens the door to experiments that were previously difficult or impossible. For example, thicker bacterial lawns that are closer to natural worm habitats obscure worm outlines and lead to more worm-like tracks that are difficult for some algorithms to distinguish from worms themselves. Perhaps more interestingly, a quantitative study of worm-worm interactions including their dependence on other behavioural variables like posture and speed is within reach. Although we do not expect to be able to track worms in large dense aggregates from brightfield imaging alone, studying the early stages of aggregate formation involving two- and three-worm interactions may already yield insight into aggregate formation. Because this analysis is compatible with large-scale imaging, it will be possible to look for mutants and natural variants that affect the early interactions and may reveal novel genes involved in worm behaviour.

Even in conditions optimized for classic segmentation and skeletonization algorithms, DTC increases the fraction of frames that yield reliable posture data, minimising gaps and preserving identity through overlaps and coils. This improvement directly translates into more data points per imaging run and increased effect sizes in phenotypic screens. The increased power comes from a better ability to detect subtle differences in behaviour as well

as the ability to detect more extreme phenotypes such as the persistent high curvature seen in Figure 3C which is effectively never observed in control worms.

Despite these advances, DTC still has limitations. For example, the long-lived close interactions that are characteristic of mating remain a challenge for DTC. We expect that a large-scale study of mating would become possible with more mating-specific training data. More generally, a significant increase in training data will likely be required to make a model that generalises to data collected on other trackers. We have included code for an annotator that makes generating new training data relatively straightforward and would welcome a collaborative approach to worm annotation to generate multi-lab training data. A complementary approach would be to generate better fully synthetic training sets that capture complex behaviours like mating.

For a modest increase in GPU time, DTC increases the value of data that is expensive to generate in terms of reagents (for drug screens) and labour.

## Materials and Methods

### *Training data*

Initially, isolated worm clips from Tierpsy were organized into sequences of 2, 4, or 8 seconds with the background subtracted. Each training clip consisted of a stack of 11 frames, each frame being 512x512 pixels. In total, we created 50,000 such clips. To simulate overlapping, we paired and stacked two clips on top of each other. Importantly, for each epoch, the dataset was randomly shuffled so that at every epoch, different pairs of clips were stacked together. This approach introduced variability and diversity to the training data. As a result, the effective number of unique clips used for training was 25,000 after stacking. The Tierpsy-derived data were supplemented with manual annotations for challenging cases included coils, complex overlaps, and non-uniform backgrounds. A total of 1000 clips were annotated using a custom graphical user interface (GUI) (https://github.com/WeheliyeHashi/Worm_annotation). To create new manually annotated training data, we have also included a script that converts videos to clips that can be loaded and annotated using the GUI. For testing, we used 2000 skeletons, which are available on Zenodo (https://doi.org/10.5281/zenodo.14517883).

### *DTC training and hyperparameters*

Training code was adapted from ref. [23]. Training was conducted on an NVIDIA RTX A6000 with 48 GB of RAM. The model was implemented using JAX, with the Adam optimization

algorithm, a learning rate of 0.001, and a modified ResNet (average pooling instead of max pooling in the first layer to break translational invariance) as the backbone architecture. The latent dimension was set to 8, and the maximum number of skeletons per grid was also set to 8. A cutoff distance of 48 pixels was applied during training, and the Principal Component Analysis dimension was set to 72.

The PCA transformation matrix was constructed using skeletons from both the training clips and supplemented with single-frame annotation data. As the model is based on a YOLO algorithm, it produces skeleton predictions, confidence scores, and latent space outputs. The output shape for the skeleton predictions was (32, 32, 8, 3, 72), corresponding to a 32x32 grid, with each cell measuring 16x16 pixels. Each cell contains 8 predicted skeletons across three central frames of the input stack, along with 72 PCA components. To convert these PCA components into real-space coordinates, we multiplied them by the PCA transformation matrix derived from the training data and single-frame annotations. The output shape for confidence scores was (32, 32, 8), and for latent space outputs, it was (32, 32, 8, 8). For inference, input images were padded so that the image resolution was a multiple of 16.

### *Fitting 3D splines*

For small numbers of worms, we found that it was faster and only slightly less accurate (Figure S1) to down-sample videos 1:3 in time before applying DTC and then using a 3D smoothing spline to interpolate over missing frames[31]. The comparisons shown in Figures 3 and 4 were done using this down-sampling and interpolation procedure.

### *Omnipose training and hyperparameters*

The Omnipose code was adapted from ref. [22]. Training was conducted on an NVIDIA RTX A6000 GPU with 48 GB of memory. The model was implemented using PyTorch, with the Adam optimization algorithm, a learning rate of 0.1, and UNET as the backbone architecture. During training, we used the average cell diameter method as input, providing the estimated diameter for each worm. However, during inference, the actual worm diameter in pixels was used to enhance prediction accuracy. The training data, the trained model, and the inference code are available on Zenodo (https://doi.org/10.5281/zenodo.14517883).

### *PAF training and hyperparameters*

We implemented the 'Part Affinity Fields' (PAF) model, based on ref. [30]. The PAF model uses vector fields to represent connections between body parts. We adopted a bottom-up approach, where all body parts are detected simultaneously in the image, unlike top-down methods that first detect individuals and then estimate their poses. The model predicts the location of landmark points along the worm body and PAFs to determine the connections between them. To link two landmarks, the algorithm calculates a confidence score by integrating the PAF values along the line segment connecting the two points. Higher scores indicate stronger connections, allowing the model to accurately link the correct body parts and construct worm skeletons. This method considers both the spatial positions and the relationships between the body parts to ensure coherence. The implementation was carried out in PyTorch, using the Adam optimizer with a learning rate of 0.001 and a UNET-based backbone architecture. To reduce under-segmentation of closely aligned worms, the spreads of both the part affinity fields (edges) and confidence maps were set to match the diameter of the worms.

### *RMSD calculation and failed predictions*

To calculate the Root Mean Square Deviation (RMSD), we measure the minimum distances between the predicted skeleton points and the labelled points. Before calculating the RMSD, we ensure that the first point of the skeleton data corresponds to the head and the last point corresponds to the tail. This alignment is achieved using the head-tail algorithm in Tierpsy[12]. Points are interpolated to be equally spaced before calculating the RMSD.

'Failures' occur when no prediction is made. For Tierpsy, this decision is based on a series of heuristics. For example, the segmented area to be skeletonised must have exactly two points of curvature above a threshold (the head and tail) and the maximum and minimum width of the worm after skeletonization must not differ more than a preset threshold (helps to detect coiled shapes with an incorrect skeleton). For DTC, failure occurs when the model does not predict any skeleton above a confidence score of 0.5. For PAF, we check the number of detected landmark points. If a skeleton has fewer than 14 points (the expected number for a complete skeleton), we classify it as a half or broken skeleton and call it a failed prediction. For Omnipose, we adopt the same approach as Tierpsy since we use the Tierpsy skeletonisation algorithm to go from segmented objects to skeletons.

### *Phenotypic screen analysis*

We used the Pearson correlation coefficient to evaluate the linear association between the features obtained by Tierpsy and DTC. We used the F-statistic to quantify the signal in a given feature, calculated across all drugs and doses. Hedge's d was used to quantify the standardized difference in effect sizes between the two methods. Hedge's d is an adjusted version of Cohen's d designed to correct for small sample bias. These values were converted to absolute values, disregarding the direction of the effect, and then averaged across all drugs and doses for each feature.

## Acknowledgements

We thank Bonnie Evans, Eleanor Warren, Riju Balachandran, and Tom O'Brien for help annotating data. This work was supported by the Medical Research Council through grant MC-A658-5TY30.

## References

1. Yemini, E., Jucikas, T., Grundy, L. J., Brown, A. E. X. & Schafer, W. R. A database of Caenorhabditis elegans behavioral phenotypes. *Nat. Methods* **10**, 877–879 (2013).
2. Restif, C. *et al.* CeleST: Computer Vision Software for Quantitative Analysis of C. elegans Swim Behavior Reveals Novel Features of Locomotion. *PLoS Comput. Biol.* **10**, e1003702 (2014).
3. Nagy, S., Goessling, M., Amit, Y. & Biron, D. A Generative Statistical Algorithm for Automatic Detection of Complex Postures. *PLOS Comput. Biol.* **11**, e1004517 (2015).
4. Swierczek, N. A., Giles, A. C., Rankin, C. H. & Kerr, R. A. High-throughput behavioral analysis in C. elegans. *Nat. Methods* **8**, 592–598 (2011).
5. Pierce-Shimomura, J. T. *et al.* Genetic analysis of crawling and swimming locomotory patterns in C. elegans. *Proc. Natl. Acad. Sci.* **105**, 20982–20987 (2008).
6. Baek, J.-H., Cosman, P., Feng, Z., Silver, J. & Schafer, W. R. Using machine vision to analyze and classify Caenorhabditis elegans behavioral phenotypes quantitatively. *J. Neurosci. Methods* **118**, 9–21 (2002).
7. Geng, W., Cosman, P., Baek, J.-H., Berry, C. C. & Schafer, W. R. Quantitative classification and natural clustering of Caenorhabditis elegans behavioral phenotypes. *Genetics* **165**, 1117–1126 (2003).
8. Feng, Z., Cronin, C. J., Wittig, J. H., Jr, Sternberg, P. W. & Schafer, W. R. An imaging system for standardized quantitative analysis of C. elegans behavior. *BMC Bioinformatics* **5**, 115 (2004).
9. Stephens, G. J., Johnson-Kerner, B., Bialek, W. & Ryu, W. S. Dimensionality and Dynamics in the Behavior of C. elegans. *PLoS Comput. Biol.* **4**, e1000028 (2008).
10. Tsibidis, G. D. & Tavernarakis, N. Nemo: a computational tool for analyzing nematode locomotion. *BMC Neurosci.* **8**, 86 (2007).
11. Sznitman, R., Gupta, M., Hager, G. D., Arratia, P. E. & Sznitman, J. Multi-Environment Model Estimation for Motility Analysis of Caenorhabditis elegans. *PLoS ONE* **5**, e11631 (2010).
12. Javer, A. *et al.* An open-source platform for analyzing and sharing worm-behavior data. *Nat. Methods* **15**, 645–646 (2018).
13. Fontaine, E., Burdick, J. & Barr, A. Automated Tracking of Multiple C. Elegans. in 3716–3719 (IEEE, 2006). doi:10.1109/IEMBS.2006.260657.
14. Roussel, N., Morton, C. A., Finger, F. P. & Roysam, B. A computational model for C. elegans locomotory behavior: application to multiworm tracking. *IEEE Trans. Biomed. Eng.* **54**, 1786–1797 (2007).
15. Broekmans, O. D., Rodgers, J. B., Ryu, W. S. & Stephens, G. J. Resolving coiled shapes reveals new reorientation behaviors in *C. elegans*. *eLife* **5**, (2016).
16. Hebert, L., Ahamed, T., Costa, A. C., O'Shaughnessy, L. & Stephens, G. J. WormPose: Image synthesis and convolutional networks for pose estimation in C. elegans. *PLOS Comput. Biol.* **17**, e1008914 (2021).
17. Mathis, A. *et al.* DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nat. Neurosci.* **21**, 1281–1289 (2018).

18. Graving, J. M. *et al.* DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife* **8**, e47994 (2019).
19. Pereira, T. D. *et al.* Fast animal pose estimation using deep neural networks. *Nat. Methods* **16**, 117–125 (2019).
20. Deserno, M. & Bozek, K. WormSwin: Instance segmentation of C. elegans using vision transformer. *Sci. Rep.* **13**, 11021 (2023).
21. Banerjee, S. C., Khan, K. A. & Sharma, R. Deep-worm-tracker: Deep learning methods for accurate detection and tracking for behavioral studies in C. elegans. *Appl. Anim. Behav. Sci.* **266**, 106024 (2023).
22. Cutler, K. J. *et al.* Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation. *Nat. Methods* **19**, 1438–1448 (2022).
23. Alonso, A. & Kirkegaard, J. B. Fast detection of slender bodies in high density microscopy data. *Commun. Biol.* **6**, 754 (2023).
24. Barlow, I. L. *et al.* Megapixel camera arrays enable high-resolution animal tracking in multiwell plates. *Commun. Biol.* **5**, 253 (2022).
25. O'Brien, T. J., Barlow, I. L., Feriani, L. & Brown, A. E. Systematic creation and phenotyping of Mendelian disease models in C. elegans: towards large-scale drug repurposing. Preprint at https://doi.org/10.7554/eLife.92491.2 (2024).
26. McDermott-Rouse, A. *et al.* Behavioral fingerprints predict insecticide and anthelmintic mode of action. *Mol. Syst. Biol.* **17**, (2021).
27. Berri, S., Boyle, J. H., Tassieri, M., Hope, I. A. & Cohen, N. Forward locomotion of the nematode *C. elegans* is achieved through modulation of a single gait. *HFSP J.* **3**, 186–193 (2009).
28. Hassinan, C. W. *et al.* Dimensionality of locomotor behaviors in developing C. elegans. *PLOS Comput. Biol.* **20**, e1011906 (2024).
29. Pereira, T. D. *et al.* SLEAP: A deep learning system for multi-animal pose tracking. *Nat. Methods* **19**, 486–495 (2022).
30. Cao, Z., Simon, T., Wei, S.-E. & Sheikh, Y. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 1302–1310 (IEEE, Honolulu, HI, 2017). doi:10.1109/CVPR.2017.143.
31. Song, A., Uhlmann, V., Fageot, J. & Unser, M. Dictionary Learning for Two-Dimensional Kendall Shapes. *SIAM J. Imaging Sci.* **13**, 141–175 (2020).

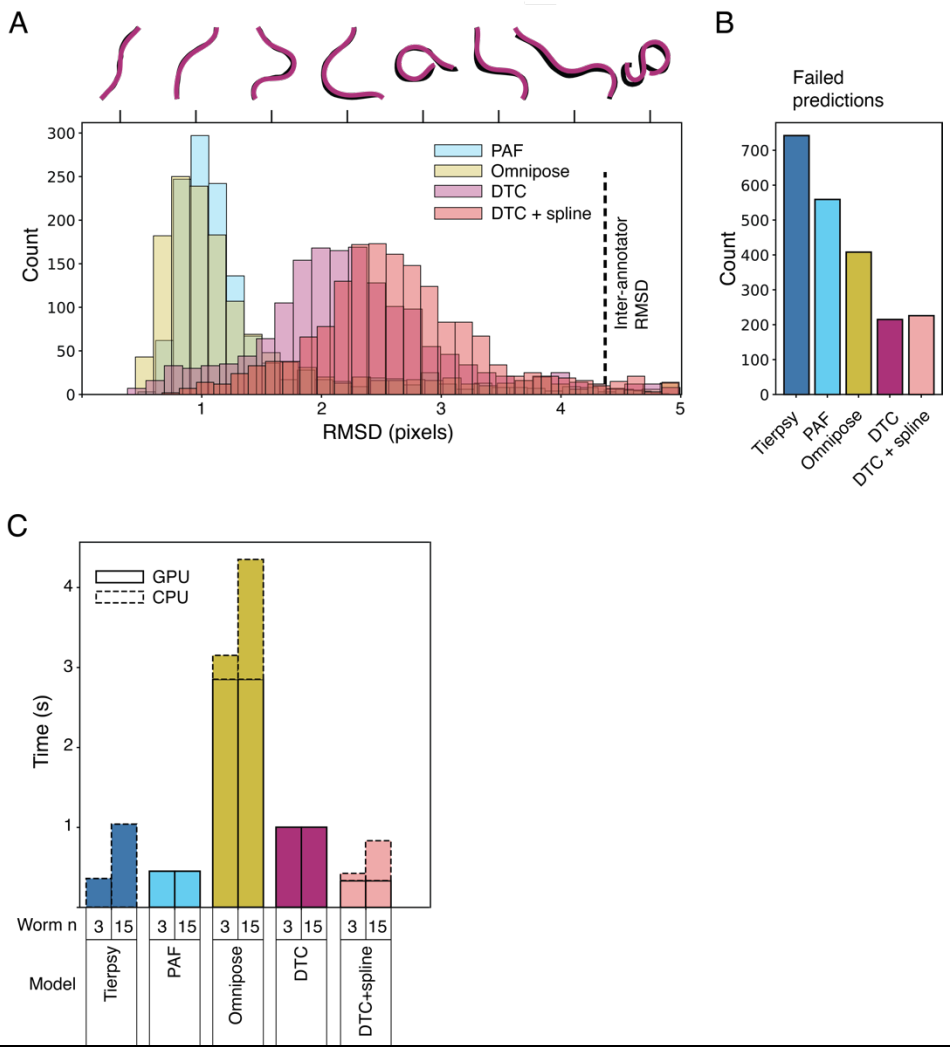


**Figure S1: Skipping frames and interpolating skeletons with a spline achieves similar accuracy and faster computational time.** (A) Histograms of root mean square deviation between held-out skeletons and predicted skeletons for each tested model, here including 'DTC + spline' in which only every third frame is tracked using DTC and skeletons for intermediate frames are interpolated using a 3D smoothing spline. The vertical dashed line shows the RMSD between three manual annotators. Skeletons above the histogram are examples that illustrate the corresponding RMSD visually. (B) Bar chart showing the number of cases in the held-out test data where a model fails to make a prediction (e.g. Tierpsy fails on coiled worms or a neural network model does not identify any worms above a confidence threshold). (C) Computation time per input frame for the different models as a function of worm number. Tierpsy only uses CPU computation while Omnipose uses GPU and CPU because we use Tierpsy's skeletonization algorithm to convert segmented regions to skeletons. DTC + spline uses both GPUs for predicting skeletons on the subsampled frames and CPUs for fitting smoothing splines to interpolate the missing frames. The advantage is modest for videos with 15 worms/well (which is nominally 240 worms/video since each camera records 16 wells on a 96 well plate).

398