Fast, versatile and quantitative annotation of complex images

**Supplemental Information**

**Supplemental Methods**

**Example ‘Wurm Paint’ app**

For our main example app, ‘Wurm Paint’, we used video frames from *C. elegans* behavior data as our image set. We identified frames where worms were partially self-occluded (see Worm Tracking section below), as the postures of worms in these images are often ambiguous and these postures occur during reorientation. Reorientation is often a behavior of interest for *C. elegans* researchers. To efficiently cover our large set of worm reorientation images, we chose to present images from this set to users in a random order. For one of our other example apps that we used to trace corn and rice roots, we presented images sequentially to allow users to use temporal context to help them annotate.

Our worm tracing example app, ‘Wurm Paint’, can be found for free on the [Google Play Store](https://play.google.com/store/apps/details?id=com.caden.drawing.wurmpaint). This version of our app also makes use of Google Play Services to facilitate login via user’s Google accounts and to implement gaming utilities for users. The source code as well as setup instructions for creating new versions of the app with arbitrary image sets can be found at <https://github.com/jiangshen/WurmPaint>. The source code can be used under an MIT license.

**Beta testing of the app**

App annotations used in this study were collected from 6 beta-testers, including both *C. elegans* experts and non-experts recruited from Georgia Tech and authors’ acquaintances, who are either authors on this paper or acknowledged in the acknowledgements. Several Morningside Elementary School (Atlanta, GA) students tested the usability of the app by drawing worm shapes with their finger. Combined, students annotated approximately 30 worm images, none of which were used in worm posture analysis (see below). All children who annotated using the app did so with the verbal consent of their parents, and no demographic or other information was collected from them.

Annotations from users of our example app published on Google Play are not included in this study, but we inform users on our Google Play site, on our app information site (<https://sites.google.com/view/wurm/app-privacy-policy>), and within the app itself what information we collect: user emails so that they may establish an account; annotations they produce; timestamps of when each annotation is updated. No demographic information is collected from users and we do not contact users via their email or share their email addresses.

To familiarize non-expert users with typical worm movement and shapes, we assembled a brief tutorial <https://sites.google.com/view/wurm/tutorial>. As general guidelines, we asked users to draw a continuous contour along the midline of the worm, starting at one end of the worm to the other end, so that the contour did not contain sharp corners, rather smooth bends along the length of the worm.

**Post-processing of annotated worm images**

Although the current version of the app allows us to upload the coordinate trajectories of user annotations, the initial version that much of the data presented here originates from only allowed us to upload the annotation superimposed on the source image. Thus, to extract annotations and reconstruct trajectories from uploaded images, some post-processing of annotated images was required. Briefly, to identify annotations, we found non-grey pixels in each image. We then binarized the annotation alone and skeletonized the image, followed by removal of branch points if branch points existed. We then checked the curvature of each line segment to ensure it fell in a reasonable range – if it did not, we broke the segment at its point of maximum curvature. Using the resulting line segments, we attempted to reconnect them to each other using both the proximity of segment endpoints and local segment slope. Once segments had been reconnected, the worm’s midline was reconstructed using the projections onto the first five eigenvectors as described previously(12). Average speed of this post-processing was 0.0597 s/ frame (n = 1000). This process is illustrated in **Supplemental Figure 1a**, and code for these steps is available in our GitHub repository. However, we emphasize that other app users need not perform any post-processing of images. Instead, coordinate trajectories can be accessed by parsing JSON files that are downloadable from our Firebase database.

**Supplemental Figures**

****

**Supplemental Figure 1 |** Worm annotation characterization(**a**)Reconstruction pipeline for worm midlines.Worm backbone annotations used in the main text were collected as images superimposed on the source image, so midlines must be reconstructed. Newer versions of the app save the drawn trajectory directly, so reconstruction is unnecessary. We found non-gray pixels (annotations) within the annotated images and binarized the annotation, followed by breaking the annotation at points of intersection or extreme curvature. Then we used local curvature and distance metrics to predict which line segments were connected and reconstructed the midline and image. (**b**) Probability density of similarity scores for ambiguous posture predictions compared to ambiguous posture consensus annotations. Red dashed line is threshold used for consensus generation. The broader similarity score distribution compared to unambiguous annotation to unambiguous ground truth comparison is caused partially by user variability and lower user accuracy and partially by the predictive nature of the state-of-the-art algorithm that sometimes leads to incorrect solutions. N = 449 (**c**) First four eigenvectors (‘eigenworms’) of the *C. elegans* posture space computed from four annotated videos (>37,000 frames). Computing eigenworms from only unambiguous postures or both ambiguous and unambiguous postures resulted in little difference, as reported in other work. Compared to eigenworms reported in other work, ours are similar, but with different eigenworms capturing a greater fraction of the total postural variability.

**Supporting Information Captions**

**S1 Movie**. **Demonstration of example app usage.** Users simply trace features of interest with a finger or stylus and upload them to the cloud. Users can also report images that don’t match the anticipated pattern (in this case, images that don’t look like worms). Movie is in real time.

**S2 Movie**. **Representative example reconstruction of worm behavior.** Segmented images from raw data are on left, with purely algorithmic reconstruction in center, and consensus annotation-based reconstruction on right (only for complex or ambiguous frames). Although algorithmic reconstruction is visually more accurate in many frames, annotation-based reconstruction produces accurate results even in frames that are inaccurately segmented or in cases where the algorithm performs poorly. Movie is at half speed from original 30 fps.