Supplementary Material 1

Baseline logistic regression model

Tracking. Automatic segmentation was performed using Cellpose[1] (model 'cyto3') based on the green channel as follows. Because some images contained regions of zero intensity that were causing artifacts during segmentation, we added noise to pixels with intensity $I_0(p)$ below 20 (in range 0-255), as:

$$I(p) = \begin{cases} I_0, & I_0(p) > 20 \\ q_{20} + \varepsilon(p), & I_0(p) \leq 20 \end{cases}$$

with q_{20} the 20% quantile of unique I_0 values, and $\varepsilon(p) \sim N(0,5)$. Segmentation was performed using Cellpose (v3.1.1.1, flow_threshold = 0.4, cellprob_threshold = 2, max_size_fraction = 0.08, using default normalization) in python (v3.12.2). Centroids were generated from segmentation masks using scikit-image[2] (v0.25.0), and tracks were automatically generated using trackpy (v0.6.4, search range = 20, memory = 30 – essentially allowing all links within the limited-size patch). We did not extensively optimize these settings as the goal was to evaluate how off-the-shelf tracking performs in this context. In addition to automated tracking results, we also obtained manually generated tracks for each video-patch.

Track processing. Automatically or manually generated tracks can contain "gaps": intermediate frames for which no cell is detected. Before feature extraction, these gaps were detected and positions interpolated using celltrackR[3] (v1.2.0, R v4.4.1, method interpolateTrack with how = "spline", https://ingewortel.github.io/celltrackR/). For some video-patches, manual or automated tracking resulted in multiple tracks as there were multiple cells in the video-patch. In those cases, we selected the track that was closest to the video-patch midpoint (25,25) in the middle frame of the image sequence. "Tracks" with less than 3 coordinates were discarded, since a "change in behavior" is undefined when there is only one movement step (2 coordinates).

Feature extraction. The following track features were extracted from each of the selected processed tracks using default methods from celltrackR[3] (v1.2.0, R v4.4.1): speed, meanTurningAngle, outreachRatio, displacementRatio, straightness, asphericity, and displacement (see package documentation for details). Because exploratory data analysis on the training data (Fig S1) showed that the features straightness and outreachRatio were bimodal in class 0, with class 1 unimodal somewhere in between, we transformed these feature values as: $v_{new} = |v_{orig} - b|$, with b = 0.55 for the outreachRatio and b = 0.45 for the straightness based on manual inspection of the distributions (Fig S1). In addition, we included two additional "standard" features: $N_{coordinates}$ in the track (as tracks of very few coordinates may indicate that there was no real cell, therefore pointing to class 0), and the track's distance d_{center} to the frame center (25, 25) at frame 10 (where again a large distance/absence of a cell in that frame may indicate that the video-patch was not focused on a cell and therefore class 0).

In addition to the abovementioned "basic" features, which used little information about the specific task, we designed two hand-crafted features to detect (sharp) changes in direction based on the step displacement vectors $\vec{v}_t = \vec{x}_{t+1} - \vec{x}_t$ (where $\vec{x}_t$ is the cell's position at time t). The first feature, $\Delta\theta$,

uses the angle θ_t (w.r.t. the positive x-axis) of each step $\vec{v}_t$, taking the difference in average direction before and after the middle of the video t_{mid} (frame 10):

$$\Delta\theta = |\langle\theta_t\rangle_{t\geq t_{mid}} - \langle\theta_t\rangle_{t<t_{mid}}|$$

The second feature instead uses the normalized average displacement vectors before and after t_{mid}:

$$s_{norm} = \| \frac{\vec{v}_{t<t_{mid}}}{\|\vec{v}_{t<t_{mid}}\|} + \frac{\vec{v}_{t\geq t_{mid}}}{\|\vec{v}_{t\geq t_{mid}}\|} \|,$$

$$where: \vec{v}_{t<t_{mid}} = \frac{1}{N_{t<t_{mid}}} \sum_{t=t_{min}}^{t_{mid}} \vec{v}_t \ and \ \vec{v}_{t\geq t_{mid}} = \frac{1}{N_{t\geq t_{mid}}} \sum_{t=t_{mid}}^{t_{max}-1} \vec{v}_t$$

where t_{min} and t_{max} are the first and last time point of the track, respectively. The value of s_{norm} is small when the two vectors point in opposite directions (cancelling each other out), and attains its max value of 2 for straight, ballistic motion. Note that the features d_{center}, $\Delta\theta$, and s_{norm} are only defined when there is a cell observed at $t = t_{mid}$. When this is not the case, we consider this as evidence that there is no focus cell in the video-patch, and therefore set these features to "class 0-like" values (Fig S1): $d_{center} = 50$, $\Delta\theta = 0$, $s_{norm} = 2$.

Logistic regression. To compare the performance of logistic regression using automated versus manual tracks and using all features versus only those not specifically crafted for the task, we trained four logistic regression models (Table 1) using scikit-learn (v1.6.1) in python (v3.12.2), using L2 regularization (C=200). In cases where all features were missing because no track was observed, all features were set to -1 before training and prediction (and an extra, binary feature was added to indicate track missingness, which was used for all models in Table 1).

Model	**Tracks**	**Trained on:**
auto-all	Automated	All features
auto-basic	Automated	All features except $\Delta\theta$ and s_{norm}
manual-all	Manual	All features
manual-basic	Manual	All features except $\Delta\theta$ and s_{norm}

Table 1. Overview of logistic regression models.

Model evaluation. All models were evaluated on the held-out test data (validation and test datasets). To further assess generalization ability, we performed 5-fold cross-validation on the training data (ensuring that video-patches from the same original video only occurred in the training or test fold, but not both). Because of the large variation observed in test accuracy between folds, we repeated this process 5 times with different splits to obtain a better performance estimate. All performances are reported as balanced accuracy (from scikit-learn) and overall score.

Supplementary Material 2 - Description of the participating algorithms

GIMR - Garvan Institute of Medical Research

The GIMR team developed TrajNet, a track-based approach designed for analyzing and classifying CBVCC data. It integrates multiple components, including cell tracking, feature extraction using a convolutional neural network (CNN), an attention-based track selection module, and a multilayer perceptron (MLP) classifier.

The first step involves tracking cells across video frames. A retrained Cellpose model (based on cytotorch0) is employed for segmentation[1], combined with trackpy/laptrack for track generation[4]. To optimize performance and minimize noise, only the green channel of each video frame is processed. This results in multiple trajectories per video, each representing a cell's movement over time. To mitigate the effect of segmentation and tracking errors, only high-intensity (>50) and large (>50 pixels) cells are considered.

Trajectories are then zero-padded to a fixed 3 × 20 size, representing (x, y, t) coordinates over 20 frames. A shallow two-layer CNN is employed for feature extraction to capture motion characteristics. The network consists of two 1D convolutional layers with ReLU activation and batch normalization, followed by max-pooling and average-pooling operations. This simple architecture is motivated by the insight that motion features, such as speed and turning angles, can be effectively captured through first- and second-order differential operations.

To focus on the most informative trajectories in the video-patch, an attention mechanism assigns weights to each trajectory's features allowing the model to aggregate the most representative motion patterns through weighted averaging. This enables the model to prioritize salient motion patterns, akin to human visual assessment. The aggregated features are then fed into an MLP classifier, which is trained jointly with the attention module for end-to-end classification.

Overfitting is mitigated through reduced model complexity, dropout layers with a rate of 0.5, weight decay, and data augmentation techniques such as rotation, scaling, time reversal, speed adjustments, and interpolation. Final predictions are obtained via model ensembling by averaging the outputs of the top three performing models.

The source code for TrajNet is available at https://github.com/lxfhfut/TrajNet.

LRI Imaging Core - Cleveland Clinic

The LRI team developed the CB pipeline, a track-based approach based on an extension of the LabGym tool[5,6], designed specifically for analyzing and classifying CBVCC data. Since multiple cells can be present in a video-patch, each exhibiting different behaviors over time, the CB pipeline ensures that all cells within the video-patches are tracked, and their behaviors are classified at every frame. Specifically, the authors categorized cell behaviors into three predefined motion patterns: inplace (minimal movement), linear (consistent movement in one direction), and orient (sudden directional

change). The pipeline integrates three main components: cell tracking, feature extraction using CNNs and long-short-term-memory (LSTM) layers[7], and an MLP classifier.

Cell tracking is performed using LabGym's Detector module, which is based on Detectron2[8]. To train the Detector, 475 manually segmented frames were augmented to produce 9850 training images. The tracking method assigns a unique identity to each cell and links them across frames using Euclidean distance calculations, ensuring reliable trajectory continuity.

For behavior classification, a model was trained on the three predefined motion patterns using manually annotated data for supervised learning. The classification pipeline consists of three components: an Animation Analyzer, a Pattern Recognizer, and a Decision Maker. The Animation Analyzer processes video-patches, cropped around each cell using segmentation masks, with 2D convolutional and LSTM layers[7]. The Pattern Recognizer extracts spatial features using 2D convolutional layers from a 2D image projection of each cell's boundary across all frames. The Decision Maker then combines these outputs through an MLP with a SoftMax function to predict behavior probabilities.

For the CBVCC classification task, the objective is to distinguish video-patches into two classes. A locational filter is applied to focus only on cells passing through the center of the frame. Predictions from frames 14, 15, and 16 are used, with weights assigned as follows: 0.1429 for linear behavior, 0.2857 for inplace behavior, and 0.5714 for orient behavior. The final summary prediction score determines whether a video belongs to class 0 or class 1.

To mitigate overfitting, the model incorporates dropout layers with a 0.5 rate, and extensive data augmentation techniques, including rotation, flipping, and brightness adjustments. Training optimization follows a categorical cross-entropy loss function with an adaptive learning rate schedule.

The source code for the CB pipeline is available at https://github.com/yujiahu415/ CBVCC.

QuantMorph - University of Toronto

The QuantMorph team developed a track-based approach for analyzing and classifying CBVCC data. This method uses the pre-trained Segment Anything Model 2 (SAM2)[9] for cell segmentation and tracking, followed by a custom LSTM-based neural network for classifying the cell tracks[7].

For data preprocessing, only the green channel of each frame was used. Cell segmentation and tracking were performed using the pretrained SAM2, which segments and tracks objects in a video starting from the centroid in the first frame they appear. Centroids were initially identified in each frame using a Laplacian of Gradients blob detector[2]. The segmentation process was terminated if SAM2 failed to produce a segmentation within 10 pixels of the previous frame's centroid or if the segmented area exceeded 500 pixels.

In the training set, manual pruning was applied to class 1 videos to ensure only class 1 tracks were used for training. Class 0 videos, containing only class 0 objects, retained all tracks for training. The centroid coordinates across frames were used to construct a 2×20 matrix representing the (x, y) coordinates of each object's trajectory. If any object was not segmented across all frames, missing data was filled by assigning the centroid from either the first or last segmented frame.

The classifier architecture consists of a 1D convolutional layer followed by five LSTM layers[7], and a final fully connected layer with SoftMax activation for classification.

During inference, a video-patch was classified as class 1 if any segmented object track within the video-patch was classified as class 1. Videos with no predicted class 1 object tracks, or with no successfully segmented objects, were classified as class 0. The object with the highest class 1 probability was used as the probability for classifying the entire video-patch.

To mitigate overfitting, the model includes dropout layers with a rate of 0.7, along with data augmentation by training on both forward and reverse orientations of centroid tracks. The training optimization uses a categorical cross-entropy loss function with class weights to address class imbalance, along with an adaptive learning rate schedule.

dp-lab - USI

The dp-lab team developed an end-to-end deep learning (DL) approach for analyzing and classifying CBVCC data. This approach is based on a 3D CNN for video classification tasks, which simultaneously processes spatial and temporal components[10]. No explicit cell segmentation or tracking was performed.

The classification model consists of three convolutional blocks, each containing two 3D convolutional layers with a progressively decreasing number of filters. The first block has 128 filters, the second has 64, and the third has 32. Each convolutional layer is followed by batch normalization and ReLU activation. After each block, a max pooling layer is applied to downsample the feature maps. The output from the final max pooling layer is flattened and passed through a MLP with two fully connected layers, each followed by ReLU activation and dropout layers. The final classification is performed using a fully connected output layer with a SoftMax activation function, producing a probability distribution over the two classes.

To mitigate overfitting, the model incorporates reduced complexity, dropout layers with a 0.2 rate, and data augmentation techniques, including random rotation, zooming, vertical and horizontal flipping, shifting, and contrast stretching. The training optimization uses a categorical cross-entropy loss function with class weights to address class imbalance.

UWT-SET - University of Washington

The University of Washington (UWT-SET) team developed the Enhanced Swin-Tiny Model, an end-to-end DL framework specifically designed for analyzing and classifying CBVCC data. This approach integrates two primary components: a standard Video Shifted Window Transformer Tiny (Swin-Tiny)[11] and a motion-guided data augmentation pipeline.

The input video frames are resized to a 224 × 224 resolution. The classification model, Swin-Tiny, is a parameter-efficient variant of the original Swin Transformer, designed to balance computational efficiency with the capacity to capture subtle motion. Its hierarchical architecture consists of four stages with layer depths of [2, 2, 6, 2], an embedding dimension of 96, and multi-head self-attention layers featuring 3, 6, 12, and 24 heads across successive stages. The model employs a patch size of [2, 4, 4] and a drop path rate of 0.2, with layer normalization and GELU activations applied to each block.

To mitigate overfitting, training is augmented with a motion-guided data augmentation pipeline designed to enhance both cell movement and imaging conditions. Frame differencing identifies high-motion regions, while HSV-based filtering selects cells from these regions by discarding irrelevant pixels based on color and brightness. Local affine transformations are applied to these high-motion patches to simulate variations in cell trajectories. Additionally, the augmentation pipeline applies global transformations across the entire frame, such as color jitter and Gaussian blur, to replicate variations in imaging conditions.

The model is trained using the AdamW[12] optimizer with a weight decay of 0.02 and a cosine annealing learning rate schedule. To address class imbalance, class weighting is incorporated into the cross-entropy loss function.

Computational Immunology - Radboud University

The Computational Immunology team developed an ensemble of a track-based method and an end-to-end DL image-based method. The final prediction was obtained by averaging the outputs of both models. The track-based method was built on a logistic regression model, as described in Supplementary Materials 1, trained on all features extracted from automatically generated tracks. The end-to-end DL image-based method is based on[13] and employs a 3D convolutional encoder with contrastive learning to learn a low-dimensional feature representation, which is then fed to an MLP classifier, as described below.

Contrastive learning compares reference training input videos with 1 positive anchor (y^+) and multiple negative anchors ($y^-_{1\dots n}$) to learn an embedding space in which similar samples (x, y^+) are pulled towards each other and dissimilar ones (x, y_i^-) are pushed apart. Following[13], training data X was first ordered by class. During training, a batch of (b = 8) reference videos are sampled and mapped to the 2-dimensional latent space ($f(x)$, dimension 8x2). The corresponding b = 8 positive anchors are sampled with an offset of one: $\forall\, x_k \in X \;\rightarrow y^+ = x_{k+1}$. Negative anchors $y^-_{1\dots b}$ are sampled uniformly from $X \setminus \{x,\ y_+\}$. A Siamese encoder network $f(x)$ and $f'(y)$ is optimized to map reference x and anchors y to the shared low dimensional space. The encoder architecture consists of two 3D convolutional layers followed by max-pooling layers, which are then passed through two fully connected layers. Using cosine similarity ψ and writing $\psi\big(f(x), f'(y)\big)$ as $\psi(x, y)$ for simplicity, the following contrastive loss[13] is minimized:

$$L_{contrastive} = E_{x,y_+,y_{-,1\dots n}}\left[-\psi(x, y^+) + log \sum_{i=1}^{b} e^{\psi(x,y_i^-)}\right]$$

The learned low-dimensional features are then fed to an MLP for classification, which consists of an elementwise activation layer with Tanh, followed by a fully connected Softmax layer for the final classification output. The entire network is optimized jointly on the representation ($L_{contrastive}$) and classification (L_{BCE}, cross entropy) objectives with equal weights: $L_{contrastive} + L_{BCE}$, using Adam optimization[14] (250 epochs, learning rate 3 x10^{-4}). No data augmentation was performed for this method.

BioVision – University of Central Florida

The BioVision team developed an end-to-end DL approach using the Video Shifted Window Transformer (Swin) model for analyzing and classifying CBVCC data[11]. The model was pretrained on the Something-Something v2 (STHv2) dataset, which focuses on human-object interactions from an egocentric perspective[15]. This pre-training helps the model capture fine-grained, localized actions, making it well-suited for analyzing dynamic cell movements and filament interactions in microscopy videos. Further fine-tuning on the CBVCC training dataset allowed the model to adapt to domain-specific features.

Input video frames were resized to 224 × 224 resolution and normalized using the standard preprocessing values of the STHv2 dataset. No explicit cell segmentation or tracking was performed.

The classification model is based on a Swin backbone with a hierarchical architecture consisting of four stages with depths of [2, 2, 18, 2]. It uses an embedding dimension of 128 and multi-head self-attention with 4, 8, 16, and 32 heads across stages. The model employs a patch size of [2, 4, 4], a window size of [16, 7, 7], a drop path rate of 0.2, and patch normalization for improved learning efficiency. After feature extraction, 3D adaptive average pooling is applied to reduce dimensionality while preserving key spatial and temporal features. The final classification is performed using a fully connected output layer with a softmax activation function.

Training optimization follows a categorical cross-entropy loss function with an adaptive learning rate schedule.

References

1. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. Cellpose: a generalist algorithm for cellular segmentation. *Nat Methods* 18, 100–106 (2021).
2. der Walt, S. *et al.* scikit-image: image processing in Python. *PeerJ* 2, e453 (2014).
3. Wortel, I. M. N. *et al.* CelltrackR: An R package for fast and flexible analysis of immune cell migration data. *ImmunoInformatics* 1–2, 100003 (2021).
4. Allan, D. B., Caswell, T., Keim, N. C., van der Wel, C. M. & Verweij, R. W. soft-matter/trackpy: Trackpy v0. 5.0. *Zenodo repository* Preprint at (2021).
5. Hu, Y. *et al.* LabGym: Quantification of user-defined animal behaviors using learning-based holistic assessment. *Cell Reports Methods* 3, (2023).
6. Goss, K. *et al.* Quantifying social roles in multi-animal videos using subject-aware deep-learning. *bioRxiv* (2024).
7. Hochreiter, S. Long Short-term Memory. *Neural Computation MIT-Press* (1997).
8. Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y. & Girshick, R. Detectron2. Preprint at (2019).
9. Ravi, N. *et al.* Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714* (2024).
10. Ji, S., Xu, W., Yang, M. & Yu, K. 3D convolutional neural networks for human action recognition. *IEEE Trans Pattern Anal Mach Intell* 35, 221–231 (2012).
11. Liu, Z. *et al.* Swin transformer: Hierarchical vision transformer using shifted windows. in *Proceedings of the IEEE/CVF international conference on computer vision* 10012–10022 (2021).

12. Loshchilov, I. & Hutter, F. Decoupled Weight Decay Regularization. (2017).
13. Schneider, S., Lee, J. H. & Mathis, M. W. Learnable latent embeddings for joint behavioural and neural analysis. *Nature* 617, 360–368 (2023).
14. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. (2014).
15. Mahdisoltani, F., Berger, G., Gharbieh, W., Fleet, D. & Memisevic, R. On the effectiveness of task granularity for transfer learning. *arXiv preprint arXiv:1804.09235* (2018).

Supplementary figures

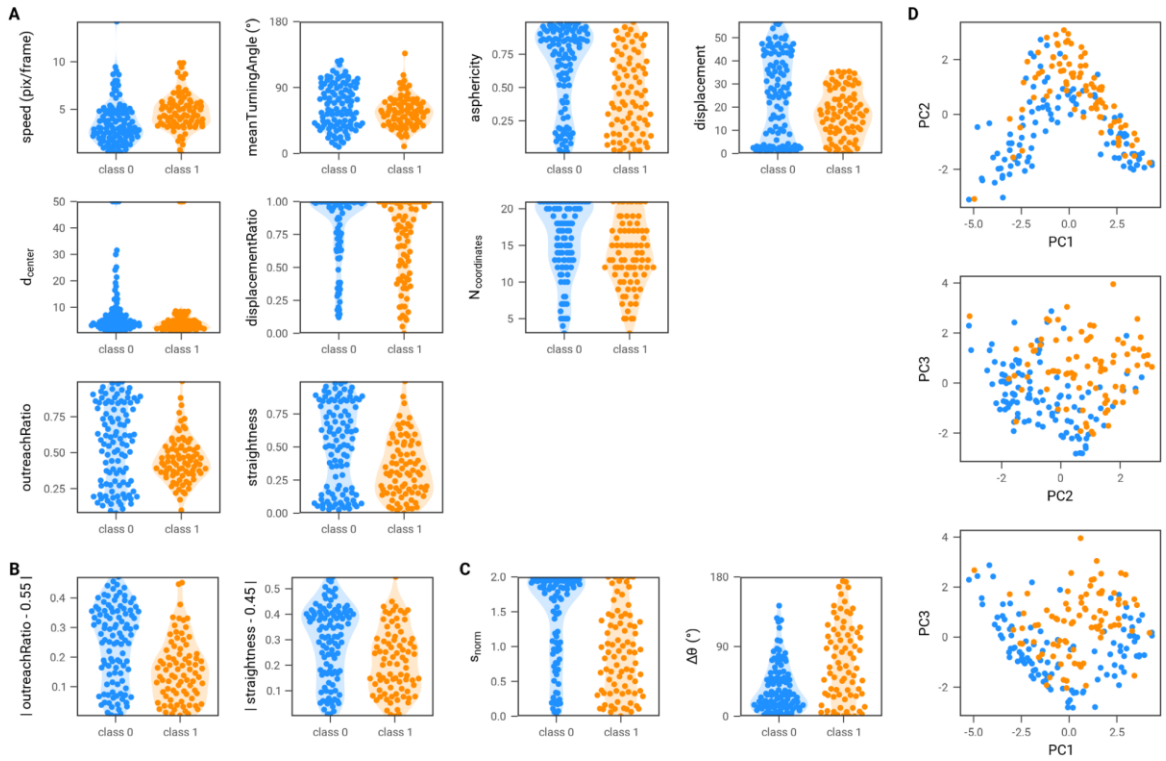

Figure S1. Exploratory data analysis on manual track features (training data). A. "Basic" features. **B.** Transformed versions of outreachRatio and straightness used for logistic regression. **C.** Hand-crafted features. **D.** PCA projections along the first three principal components using all features.

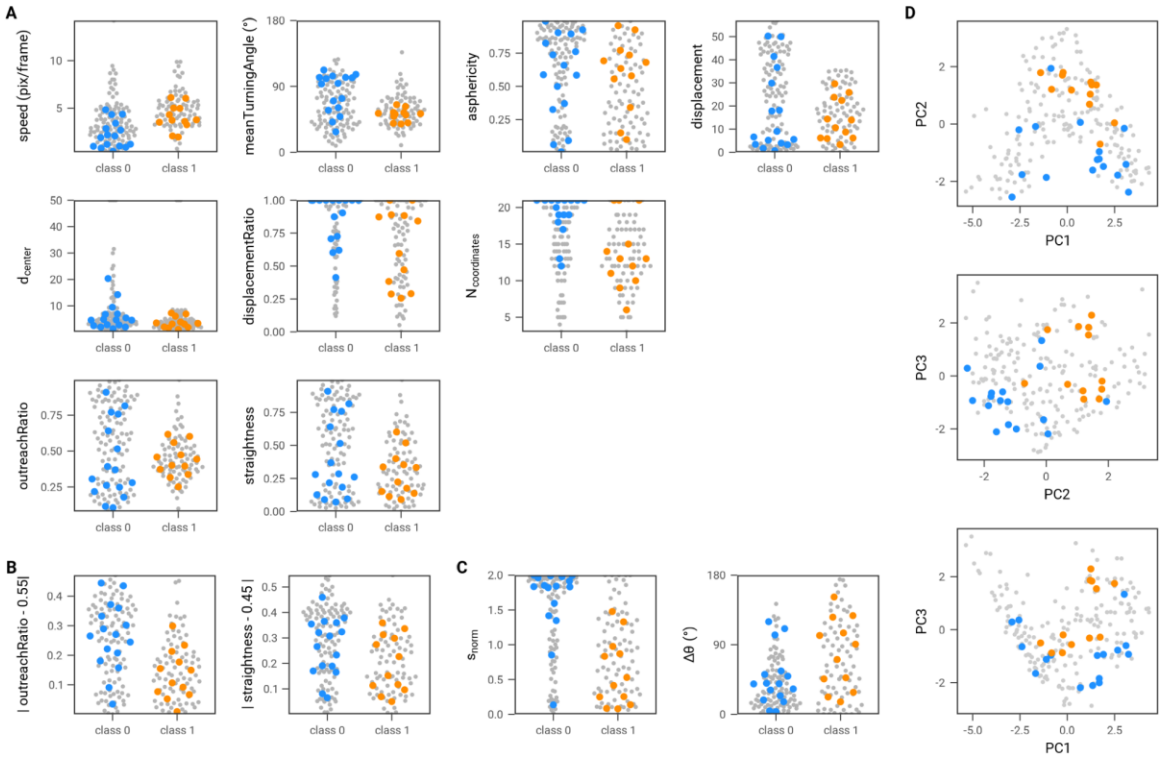

Figure S2. Exploratory data analysis on manual track features (validation set). Gray dots: training data from Figure S1. **A.** "Basic" features. **B.** transformed versions of outreachRatio and straightness used for logistic regression. **C.** Hand-crafted features. **D.** PCA projections along the first three principal components using all features.

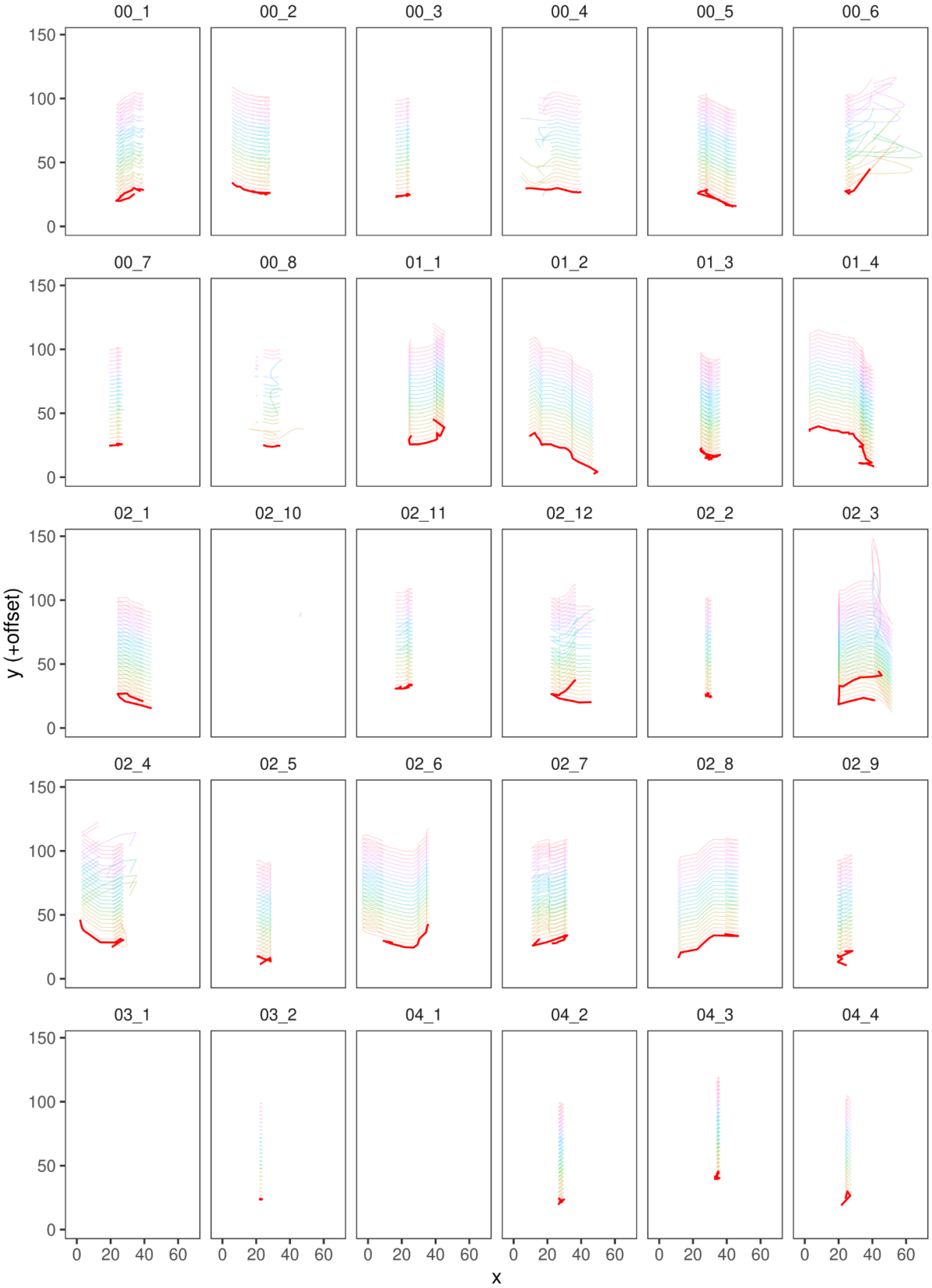

Figure S3. Variation in automated tracking results on the validation set. Thick, red line: tracks used for evaluation. Thin colored lines: results from 25 replicates of the tracking pipeline with different random seeds.

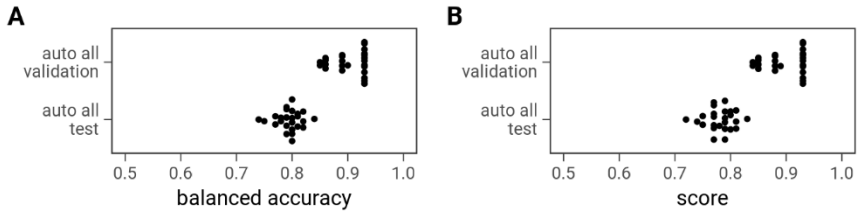

Figure S4. Effect of errors in automated tracking on logistic regression performance. The automated tracking pipeline was repeated 25 times to generate different tracks for validation and test sets (Fig S3), used to evaluate a single trained model ("auto all" from Figure 6, trained on the full training set). The figure shows evaluation results across those 25 tracking replicates (dots). Differences in generated tracks translated to substantial variation in model performance – even though both the trained model and the videos used for validation were fixed.