Table S1. Pose-derived kinematic and postural feature-extraction routines
The table lists the main kinematic features extracted across different time windows in the custom FishFeatureExtractor, a description of its computation (centre column), and the pose-estimation landmarks used (right column). All body parts refer to the full landmark set specified in (n = 18). Specific labels (e.g., Bodypart2, Bodypart16) follow the project's DeepLabCut naming scheme. Lionfish-specific added features are marked by an asterisk (*).

Extraction Methods	Description	Landmarks Used
calc_movement	Computes the Euclidean displacement between consecutive frames for each body part and sums these to yield a "Summed_movement" measure.	All body parts (from self.bp_names)
calc_X_relative_to_Y_movement	Computes the difference between the x-axis and y-axis displacements for each body part, then aggregates these differences.	All body parts (from self.bp_names)
calc_velocity	Calculates velocity as a rolling sum of the movement (over a 1-second window based on fps) for each body part, then averages these values.	All body parts (from self.bp_names)
calc_acceleration	Computes acceleration as the difference between current and shifted velocity values over multiple rolling windows.	All body parts (from self.bp_names)
calc_rotation	Determines the fish's orientation by calculating the angle between two key points and converting this angle to compass directions.	**Bodypart2** (head) and **Bodypart16** (tail tip used for angle calculations)

calc_angular_dispersion	Computes the consistency (or dispersion) of heading by using cumulative sine and cosine of the orientation.	Derived from rotation (thus indirectly from **Bodypart2** and **Bodypart16**)
calc_N_degree_direction_switches / **calc_45_degree_direction_switches**	Detects and counts sharp directional switches (e.g. 90° and 180° changes) in compass direction over time.	Derived from orientation computed using **Bodypart2** and **Bodypart16**
calc_distances_between_body_part	Computes pairwise Euclidean distances between every combination of body parts, then derives rolling statistics (mean, std, skew, kurtosis) for these distances.	All body parts (from self.bp_names)
pose_confidence_probabilities	Aggregates detection confidence (probabilities) for each body part and computes counts of low-confidence detections at several thresholds.	All body parts (using detection probabilities in self.p_cols)
calc_rhythmic_patterns*	Uses autocorrelation and spectral analysis (Welch's method) on the movement signal to capture periodicity and extract dominant frequency components.	All body parts (from self.bp_names)
calc_turning_metrics*	Computes turning rate from changes in orientation and identifies turning bouts (duration, frequency) based on these rates.	Uses orientation computed from **Bodypart2** and **Bodypart16**
calc_energy_metrics*	Computes a kinetic energy proxy (0.5 × velocity^2) for each body part and aggregates these values to estimate overall energy expenditure.	All body parts (from self.bp_names)

calc_complexity_metrics*	Calculates the entropy of the movement distribution over rolling windows to capture movement variability (complexity).	All body parts (from self.bp_names)
calc_path_metrics	Computes path tortuosity and swimming efficiency by comparing the cumulative movement (path length) to the straight-line distance traveled.	**Bodypart16** is used as a reference for tail movement.
calc_body_curvature*	Derives segment angles between consecutive triplets of body parts and sums them to compute overall body curvature and its changes.	All body parts (order defined in self.bp_names)
analyze_swimming_bouts	Detects bouts of active swimming by thresholding the summed movement and computes bout statistics (duration, frequency, etc.).	Aggregated information from all body parts
calc_forward_movement_projection*	Projects the center's displacement onto the current heading (obtained from rotation) to capture forward propulsion. Aggregated over multiple time windows.	**Bodypart14** (center); orientation from **Bodypart2** and **Bodypart16**
calc_vertical_displacement_features	Computes vertical (up–down) displacement by taking the difference in the center's y-coordinate between frames, with rolling aggregates over specified windows.	**Bodypart14** (center)
calc_enhanced_turning_dynamics*	Derives a "turn sharpness" metric by multiplying turning rate and angular acceleration, then aggregates mean and standard deviation over specified time windows.	Derived from turning features computed via **Bodypart2** and **Bodypart16**

calc_occlusion_indicators	Monitors sudden drops in the mouth detection probability as an occlusion indicator, and counts these events over time windows.	**Bodypart2** (mouth detection probability)
calc_relative_landmark_order*	Checks the horizontal (x-coordinate) order between the mouth, center, and tail to verify expected arrangement (used to infer reordering during turns).	**Bodypart2** (mouth), **Bodypart14** (center), and **Bodypart16** (tail)
calc_composite_swimming_metric*	Integrates forward movement projection, vertical displacement, and turn sharpness into a single composite score, with aggregated mean and std over time windows.	Derived from features of the **center** (bodypart14) and turning features (bodypart2 & bodypart16)
calc_inter_body_coordination	Computes the rolling correlation between the average movement of the head group and tail group, with aggregated statistics over specified windows.	**Head group:** e.g. bodypart1, bodypart2, bodypart13; **Tail group:** e.g. bodypart15, bodypart16
calc_jerk_and_angular_acceleration*	Calculates jerk (the third derivative of position) for the center's velocity and computes angular acceleration from changes in turning rate, aggregated over time windows.	**Bodypart14** (center for jerk) and turning derived from **Bodypart2** & **Bodypart16**
calc_frequency_domain_features	Applies Welch's method to the center's movement signal to extract the dominant frequency component and its power.	**Bodypart14** (center)

calc_lateral_symmetry	Computes the lateral width (difference between maximum and minimum x-coordinates) across all body parts and calculates the standard deviation to assess symmetry.	All body parts (using x-coordinates from self.bp_names)
calc_fractal_dimension*	Estimates the fractal dimension of the center's trajectory via a box-counting method, reflecting the complexity of the movement path.	**Bodypart14** (center trajectory)
calc_spatial_occupancy	Measures the displacement of the center from its starting position and computes rolling variance in x and y to assess spatial occupancy.	**Bodypart14** (center)

a) Hover

	Predicted 0	Predicted 1
Actual 0	923	54
Actual 1	168	74

b) Rest

	Predicted 0	Predicted 1
Actual 0	1062	65
Actual 1	60	32

c) Swim

	Predicted 0	Predicted 1
Actual 0	978	78
Actual 1	24	139

d) Hunting

	Predicted 0	Predicted 1
Actual 0	386	111
Actual 1	28	694

Figure S2. Confusion matrices for lionfish behaviour classifiers The figure presents confusion matrices for four binary classifiers trained to detect distinct lionfish behaviours: hover, rest, swim, and hunting. In each matrix, rows represent the actual behaviour labels obtained from human observers, and columns indicate the classifier's predictions, where 0 corresponds to behaviour absence and 1 to behaviour presence. The top-left and bottom-right cells represent correct classifications, including true positives and true negatives, while the top-right and bottom-left cells indicate misclassifications, including false positives and false negatives. Blue shading corresponds to correct predictions, with darker blue indicating higher counts. Red shading reflects incorrect predictions, with darker red representing a greater number of errors.

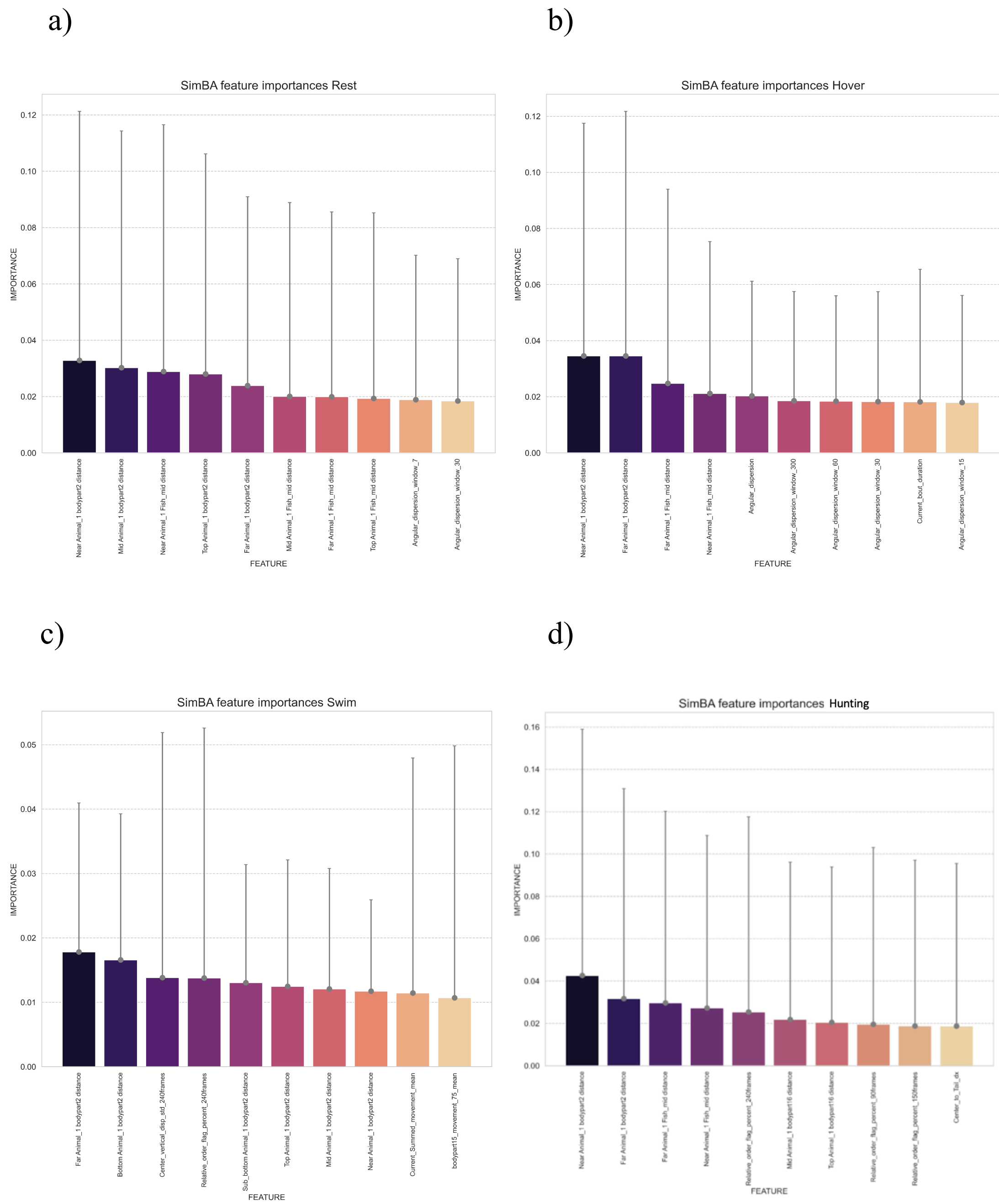

Figure S1. Top-10 Feature Importances for Rest, Hover, Swim, and Hunting Classification Bars show the ten most predictive features for each behavior class, ranked by mean importance in the Random Forest models implemented in SimBA. Error bars indicate the standard deviation of feature importance across cross-validation folds, these are large due to variations in experimental setup (e.g., lighting, camera angle, lens distortion).