PrecisionTrack: Reliable Tracking of Large Groups of Animals Interacting in Complex Environments Over Extended Periods

Vincent Coulombe[1, 2], David-Alexandre Roussel[1], Mohamad Sadegh Monfared[1, 2], Quentin Leboulleux[2], Modesto R. Peralta III[2], Khadijeh Aghel , Benoit Gosselin[1, 2], and Benoit Labonté[2]

[1]sBIOML, Laval University, Quebec, Quebec, Canada
[2]CERVO Brain Research Centre, Quebec City, Canada

October 2024

Abstract

Mice living in complex social groups within ethological environments exhibit a wide range of adaptive behaviors, including individual and group dynamics, often undetected in standard behavioral studies. To better understand the coping and adaptive strategies employed by each member of a social colony, it is necessary to develop tools that enable accurate, long-term monitoring of large groups of animals in a fully automated and unbiased manner. In this work, we introduce PrecisionTrack, a practical solution to the multi-animal pose tracking problem. This solution enables an accurate and reliable tracking of large groups of animal socially interacting within complex environments, even over prolonged periods. Our algorithm builds on a Transformer-CNN hybrid neural network for cross-species classification, detection, and pose estimation, leveraging pose-based matching AI-driven ArUco identification for fast and accurate re-identification. PrecisionTrack demonstrates superior performance in tracking accuracy and latency compared to current gold standards, while sustaining these capabilities over extended periods when tracking large groups of animals in complex environments. Furthermore, we trained PrecisionTrack to identify and monitor over 30 animal species. Overall, PrecisionTrack represents a reliable, accessible step toward adopting more ethological methodologies in behavioral research.

The study of animal behavior provides valuable insights into the mechanisms underlying various pathological processes and conditions [1]. Traditionally, this effort has relied heavily on manual annotation and direct observation, methods that are both time-consuming and prone to error. Automatic approaches often require prohibitively expensive infrastructure or are limited in spatiotemporal accuracy [2, 3]. While recent advancements in computer vision, deep learning, and artificial intelligence (AI) have shown promise for behavioral analysis, they still face important limitations when applied to multi-animal tracking in complex/real-world environments. Limiting their practical application for studying social dynamics.

The integration of computer vision with deep learning tools has potentiated advancements in the field of multi-animal tracking. Tools such as convolutional neural networks (CNNs; [4, 5]) and recurrent neural networks (RNNs; [6]), have shown good efficiency at identifying and distinguishing individual animals in challenging conditions, such as overlapping trajectories, occlusions, or poor lighting. Tools like DeepLabCut and SLEAP leverage these approaches to achieve high level accuracy in animal identification and tracking. However, these approaches still face two major weaknesses which limit their practical use. First, they lack the ability to reliably differentiate between most intra-specie subjects, making it nearly impossible to recover from tracking errors, which propagate over time. Resulting in the well documented propagation of errors [7]. Second, they are not fast enough to track large groups of subjects in real-time [8], resulting in prohibitive computational costs for studies requiring extended monitoring.

Here, we introduce PrecisionTrack a practical social dynamic monitoring system, designed to be reliable over many real-world, ethological scenarios. As such, we present the first real-time multi-animal pose tracking algorithm capable of maintaining unique animal identification over extended periods, even in large groups of animal interacting in complex environments. Taking inspiration of the recent breakthrough advancements in the object detection field, we developed a highly efficient multitask Transformer-CNN hybrid neural network architecture to predict cross-species classification, detection and pose-estimation. We also further refined the current state-of-the-art in multi-object tacking (MOT), improving its robustness and its ability to recover lost trajectories, especially in top-down free space scenarios where the subjects might drastically change their directions and/or velocity within milliseconds. Combined with our novel AI driven ArUco identification plugin based on the Tailtag system [9], PrecisionTrack has demonstrated effective at maintaining the initial tracking accuracy over time. Thus, we are providing a novel solution addressing the challenges imposed by multi-animal pose tracking, including intra-species tracking in high-occlusion scenarios and the ability to track over extended periods.

1 Results

By leveraging a high-recall detection step, an association step that accounts for all detections (Figure 3c), a precise association cost (18), and a dynamic motion predictor (Figure 4a), we were able to outperform the current gold-standard multi-animal tracking algorithms in both the MOTA and the IDF1 metrics. Performance was evaluated on our challenging 20-mice benchmark, which feature 20 mice interacting socially within an enriched environment.

1.1 The MICE dataset and benchmarks

To further advance the field of multi-animal pose tracking, we have compiled a series of comprehensive and challenging datasets aimed at improving the reliability of tracking solutions in real-world scenarios. We argue that the currently available multi-mouse datasets [4] are not sufficiently challenging to meet this objective. To address this gap, we present a more complex pose-estimation dataset featuring 20 mice interacting in an environment enriched with spinning wheels, water bottles, feeding trays, tubes, wooden blocks, and nesting materials. Each mouse's pose, or skeleton, is defined by eight key points: snout, left ear, right ear, centroid, left hip, right hip, tail, and tag (Figure 2). This challenging dataset aims to improve performance's transferability from validation to experimental situations.

Inspired by recent advancements in multi-object tracking [10], which have benefited from the availability of critical datasets [11], we have manually labelled an extensive sequential dataset. This dataset includes over one million locations and poses, and will be freely available for all. We hope this resource will help the community develop more advanced multi-animal pose tracking and action recognition algorithms, driving further progress in the field.

To facilitate the development of reliable systems in challenging real-world scenarios, we are also providing detailed benchmarks to help standardize future multi-mouse algorithm comparisons. These benchmarks consist of approximately three minutes of continuous, manually labelled ground truth data.

1.2 Multitask CNN-Transformer hybrid neural network architecture

Inspired by the work of [12], our model directly regresses each prior's keypoint positions, inherently assembling them into skeletons during detection (Figure 2a). This approach has been shown to be both more accurate and faster than bottom-up and top-down methods [12]. We also leverage the CSPNeXt back-

bone (Figure 1b), which has demonstrated a Pareto-dominant accuracy/latency tradeoff across multiple benchmarks [13]. Additionally, drawing inspiration from advancements in Transformer-based object detection, we incorporated the Attention-based Intra-scale Feature Interaction (AIFI) module from [14] into the final feature map of our feature pyramid. This encoder add-on has been shown to significantly enhance accuracy for smaller objects with only a marginal increase in latency [14]. By combining these incremental architecture improvements, we were able to outperformed [12] across all evaluation metrics (Figure 1a) while maintaining competitive latency performance (Figure 1c).

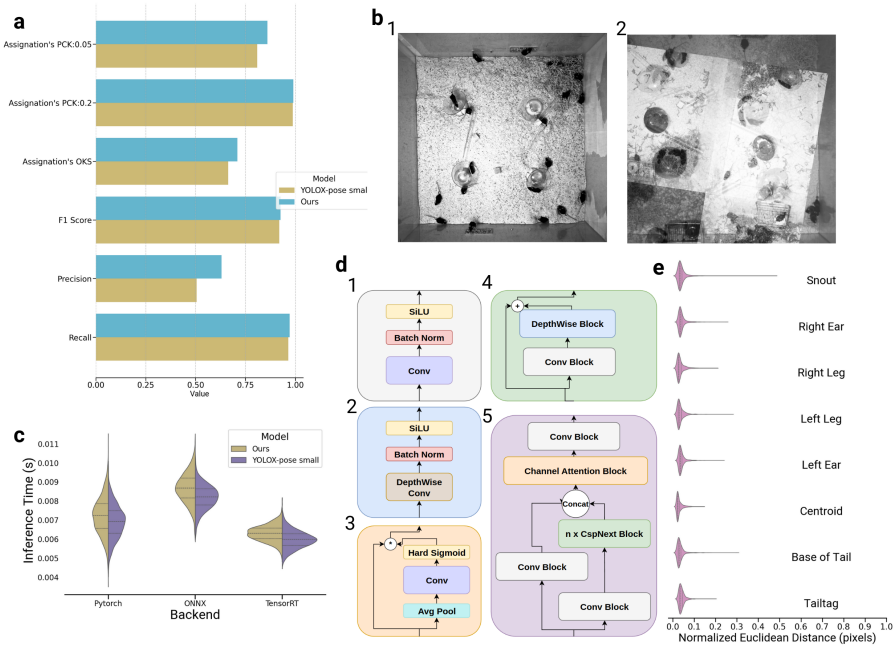

Figure 1: **a**, The Percentage of Keypoint Accuracy (PCK) at 5% and 20% of the animal's size, the Object Keypoint Similarity (OKS) for successfully detected subjects, and the F1 Score, Precision, and Recall of subjects for the YOLOX-pose baseline and our model on MICE's validation split. **b**, Examples of augmented data from the first training stage, incorporating Mosaic and Mixup augmentations along with randomly applied geometric and color space transformations. **c**, Inference delays for the YOLOX-pose baseline and our model across three different backends (PyTorch, ONNX, and TensorRT), including the preprocessing, the logits prediction, and the post-processing steps. All delays are reported at FP16 precision with a batch size of 30 on an RTX-3090 GPU. **d**, Forward propagation through a CSP-block, which includes CSPNeXt-blocks with a depth-wise convolution block, a Channel Attention (CA) block, and standard convolution blocks. **e**, The distribution of keypoint prediction errors for our model on MICE's validation split, with distances normalized by the animal's size.

1.3 Model evaluation

We argue that an ideal object-detector and pose-estimation algorithm developed for real-time object tracking should jointly optimize three key qualities: 1) high detection coverage, detecting as many subjects as possible in each frame, 2) accurate pose inference for each detected object, and 3) low latency. As such,

our evaluation metrics are designed to capture these qualities.

First, we measure each detection's Recall, Precision, and F1 Score against their corresponding labels. Next, we evaluate the matched detections' pose accuracy using the OKS metric (Equation 15) and correctness using the PCK metric at 5% and 20% thresholds. Figure 1a illustrate a comparison between our model's and YOLO-pose's [12] on the MICE validation set. Our pose-tracking metric combine the detection's F1 Score and the assignation's OKS to create a balanced score (equation 5). Additionally, detection latencies are reported for various backends (Figure 1c). Finally, we present the distribution of distances between predicted and labeled keypoint locations, normalized by the size of their corresponding subjects, in Figure 1e.

Model	**Pose-Tracking Metric**
YOLO-Pose-s	79.16
Ours	**84.5**

Table 1: Performance comparison between our approach and YOLO-pose, the current state-of-the-art for the pose-accuracy/latency tradeoff, on our MICE validation set.

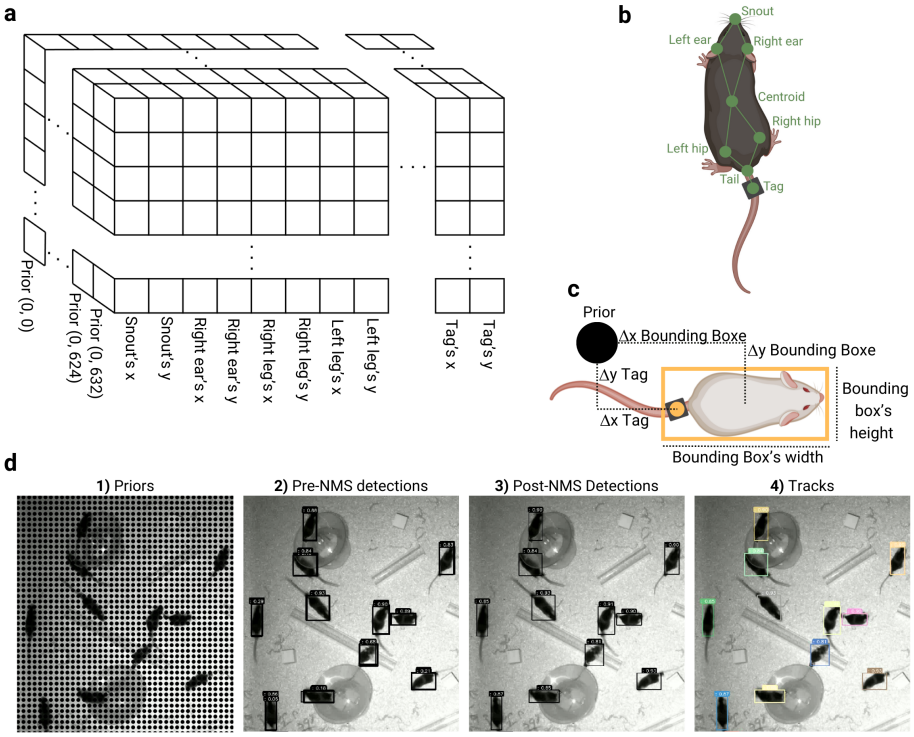

Figure 2: **a**, A breakdown of the decomposed keypoint head convolution block shows how the prior's features are processed in parallel, resulting in 8,400 pose predictions across the three levels of the feature pyramid. These predictions are then refined by multiplying them with the prior's stride and adding the prior's coordinates. **b**, The MICE dataset defines a top-down view of the mouse skeleton with 8 keypoints: snout, left ear, right ear, centroid, left hip, right hip, tail, and tag. **c**, Representation of each prior, which regresses the bounding box's position (center (x, y), width, and height) along with the positions (x, y) of 8 keypoints separately via decoupled heads. These values are then multiplied by the prior's stride and adjusted with the prior's coordinates to correctly position the bounding box and keypoints on the image. **d**, Each prior (1) generates bounding boxes, class probabilities, confidence scores, and keypoints. This process results in a total of 8,400 highly overlapping bounding boxes (2). Using non-max suppression, we remove the redundant detections (3). Our tracking algorithm then associates those detections with tracks (4).

1.4 Multi-animal tracking

Tracking-by-detection (TBD) is currently the most effective paradigm for multi-object tracking (MOT) [15]. Among TBD algorithms, SORT [16] is highly re-

garded for its simplicity, speed, and accuracy, having pioneered the widely used TBD circular processing loop (Figure 3a). When tested on our 20-mice benchmark, our implementation of SORT achieved a MOTA of 94.2%, an IDF1 of 88.8%, and an average processing speed of 84.0 FPS. More recently, ByteTrack [10] has emerged as a Pareto-dominant solution in state-of-the-art MOTA/FPS comparisons [10]. ByteTrack improves upon SORT by incorporating all detections during the association step via matching cascade, rather than relying solely on confident detections. This enhancement allows ByteTrack to better maintain the trajectories of highly occluded subjects (Figure 3c). On our 20-mice benchmark, our ByteTrack implementation achieved a MOTA of 91.8%, an IDF1 of 86.8%, and an average speed of 82.64 FPS. This slight decrease in performance can be attributed to the high degree of occlusion in the benchmark, which leads to a greater number of false positive associations, an 11% increase compared to SORT. Despite this drawback, we argue that the optimal path forward is to improve the robustness of ByteTrack, rather than reverting to SORT, given ByteTrack's theoretically superior capability in maintaining trajectories in challenging scenarios.

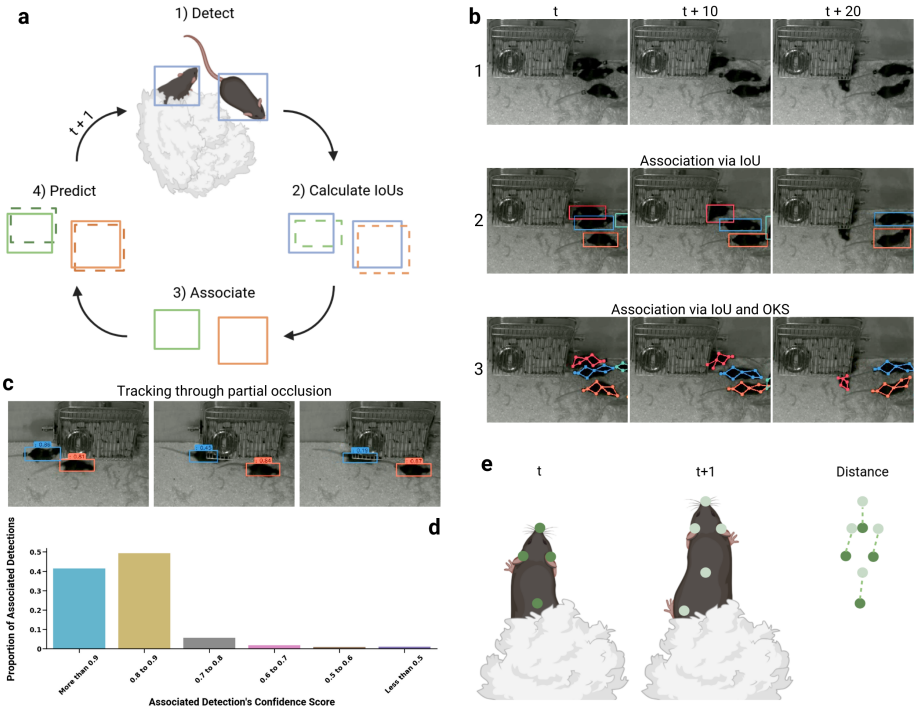

Figure 3: **a**, Processing flow of the original tracking-by-detection algorithm [16]. The current detections are associated with tracks based on their similarity, typically using IoU, with the predicted positions of the tracks from the previous time step. **b**, A real use case demonstrating the benefit of OKS-enhanced association. In this example, the red mouse is partially occluded by the feeder (1). An algorithm relying solely on IoU for association would fail to match the visible head (2, $t + 20$) to the red mouse because the IoU of the visible head does not overlap with the red mouse's last seen position (2, $t + 10$). However, an algorithm using both IoU and OKS would successfully associate them, as the ears of the visible head (3, $t + 20$) are close enough to the red mouse's last seen ears (3, $t + 10$). **c**, Using all detections, not just the confident ones (typically those with a score above 50%), allows the tracking of semi-occluded mice. In this case, the blue mouse is still tracked even though its associated detection has a score of 19%. **d**, In our benchmarks, most of the associated detections have a confidence score above 80%. **e**, A visual representation of the matching pair distances used for the OKS metric calculation.

Both SORT and ByteTrack, originally designed for MOT benchmarks focused on pedestrians and vehicles [11], utilize a unidirectional, constant-velocity Kalman Filter [17] in their association step. While this approach is effective for scenarios such as tracking pedestrians on sidewalks or vehicles in lanes, it is less

suited to multi-animal tracking settings. In such cases, subjects are typically monitored in open fields from a top-down view and exhibit highly unpredictable movement patterns, including rapid changes in direction and speed. For example, a mouse may abruptly start interacting with others or suddenly change its intended destination, as shown in Figure 4a. To address these limitations, we proposed the Dynamic Kalman Filter (DKF), an augmented version of the original Kalman Filter. The DKF incorporates Newton's laws of motion (Equations 8 and 10) to account for changes in both direction and speed. When tested on our 20-mice benchmark, our ByteTrack implementation equipped with our DKF achieved a 95.8% MOTA, 94.5% IDf1 at 92.3 FPS on our 20-mice benchmark.

Another notable limitation of ByteTrack is its exclusive reliance on target position during the association step. We addressed this by leveraging subject pose information alongside position data, thereby improving the robustness of the association step. This enhancement is particularly valuable in semi-occluded scenarios, which are common in practice. For example, as shown in Figure 3b, relying on IoU alone would fail to correctly associate the red mouse's head (2, $t+20$). However, combining IoU with OKS enables successful association, as the ears of the visible head (3, $t + 20$) closely match the red mouse's last observed ears (3, $t + 10$). ByteTrack with pose-sensitive association achieved a MOTA of 95.8%, an IDF1 of 94.4%, and an average speed of 75.8 FPS on our 20-mice benchmark.

Finally, combining the DKF with pose-sensitive association yielded our proposed PrecisionTrack system, which achieved a MOTA of 95.9%, an IDF1 of 94.6%, and an average speed of 83.3 FPS on our 20-mice benchmark.

Another significant challenge in multi-animal pose tracking is the context of the scenes. Multi-animal tracking often occurs in confined environments, such as vivariums, where the number of subjects remains constant. This constraint necessitates the ability to recover lost trajectories whenever occluded subjects reappear. To address this, we developed a novel online trajectory stitching algorithm based on dynamic search zones. These search zones evolve in real time, using a subject's last known velocity and position (Figure 3c) to predict where the subject is most likely to reappear. By dynamically adjusting the search zones, the algorithm efficiently matches newly visible subjects to their lost trajectories (Figure 3b).

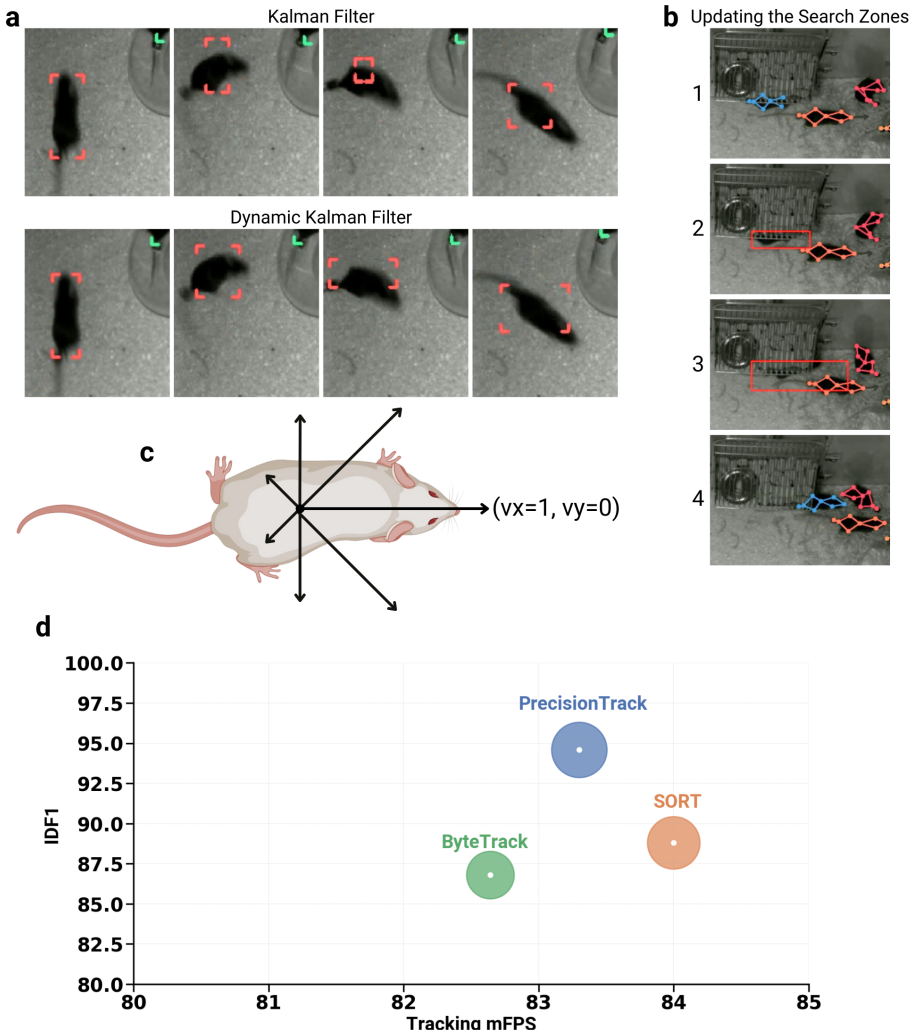

Figure 4: **a**, Side-by-side comparisons between the basic Kalman Filter and our Dynamic Kalman Filter during a sharp turn followed by rapid acceleration. Note that the corners depict the filter's next frame prediction, not the actual tracking. **b**, Real depiction of the evolution of a search zone (2, 3). Given its last seen velocity (1) and a beta of 1, our search-based stitching algorithm recovers the lost mouse's trajectory once a new, unassigned mouse is detected (4). **c**, Visual representation of the velocity biased search zone update values. Each side of the search zone expands relative to its cosine similarity with the velocity vector (vx, vy), this representation assumes an epsilon of 0. **d**, Performance comparison of our method (PrecisionTrack), ByteTrack and SORT on our 20-mice benchmark. The x-axis represents the mean tracking frames per second (mFPS), the y-axis represents IDF1, and the size of the circles corresponds to MOTA (larger is better).

1.5 Comparison with Current Gold Standards

The reported tracking metrics (Figure 4d) comprise the CLEAR metrics [18], specifically MOTA (Equation 30), which measures overall tracking accuracy, and IDF1 (Equation 31), which evaluates the algorithm's ability to preserve correct identities over time. For fair comparisons, all methods were trained on the MICE dataset. To ensure that the reported tracking latencies accurately reflect the real delay between the start and end of tracking, we monitored the performance of entire systems rather than focusing solely on their detection step. Using a batch size of 30, these latencies were recorded on an RTX-3090 GPU paired with an Intel i7-10700k CPU. See the Computational Resource Usage section for more details.

1.6 Sustaining performances over a prolonged period

We validated our pipeline's ability to maintain object identity over extended durations using three 10-minute labeled sequences from our extensive sequential dataset. To evaluate long-term performance, we compared the tracking accuracy of PrecisionTrack alone with PrecisionTrack augmented by our AI-driven detection context, the ArUco validation plugin. This plugin leverages ergonomic, non-constricting fiducial tags called Tailtags, which have been shown to effectively identify individual mice for at least seven days [9].

To assess the system's ability to maintain the correct IDs, the IDF1 metric (Equation 31) was computed every minute to monitor its progression over time. As shown in Figure 5a, PrecisionTrack without validation experienced a rapid decline in IDF1, dropping nearly one-third from 96.4±1.5% to 67.5±4.5% within 10 minutes. This significant decrease was confirmed by one-way ANOVA (p-value = 3.7e-6), with differences emerging as early as the third minute (p-value = 0.049).

These results align with expectations, given the occlusive areas in our vivarium, such as nests and feeders, which exacerbate a key weakness of our tracking solution, its difficulty to distinguish between subjects of the same species. Qualitative observations revealed that most identity switches occurred in these highly occlusive regions. For instance, when two mice similarly and simultaneously approached and hid underneath the same area, they were often misidentified upon reemerging (Figure 5d). Such scenarios occurred frequently, every few minutes, during our experiment. In contrast, PrecisionTrack with the ArUco validation plugin maintained consistent IDF1 scores, showing no significant decline (one-way ANOVA, p-value = 0.903). This consistency was achieved without compromising real-time performance, as the system (PrecisionTrack + ArUco validation) operated at 37 FPS. Figure 5b shows the distribution of correction delays in seconds. Although the aforementioned hiding behaviors skewed the

average ArUco re-identification delay to 14.68 seconds, as mice often remained hidden together for several seconds. The median delay was only 4.76 seconds, indicating that most corrections occurred promptly. Overall, these findings demonstrate the effectiveness of the ArUco validation plugin in preserving PrecisionTrack's accuracy over time by enabling it to reliably differentiate between subjects of the same species.

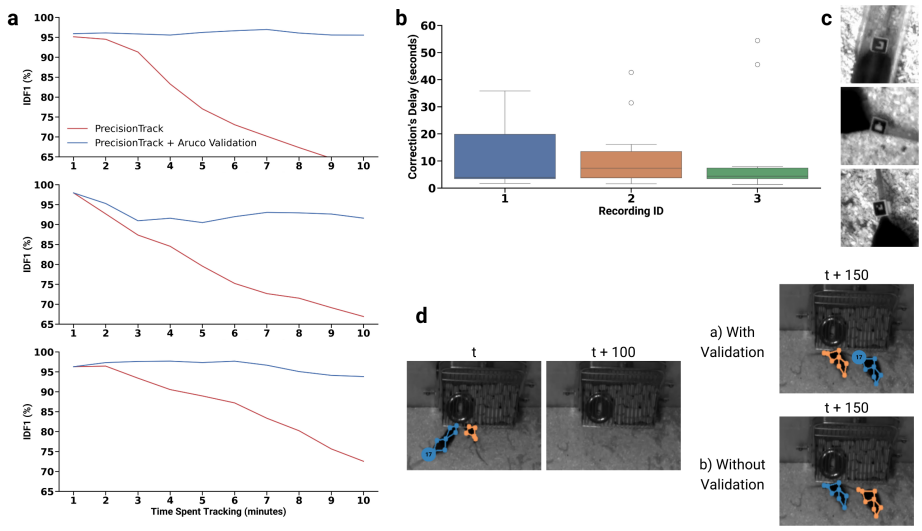

Figure 5: **a**, Visualization of IDF1 evolution during three 10-minutes recordings. Without the ArUco tag validation, tracking mistakes increase over time, causing IDF1 to decline rapidly. The addition of ArUco tail tags mitigates this issue, enabling reliable tracking over extended periods. **b**, Delay distribution between tracking mistakes and their corrections via Aruco validation for our three 10-minute recordings (∼15,000 frames each). A total of 42 corrections were performed, with an average delay of 14.68 seconds and a median delay of 4.76 seconds. **c**, The ArUco detection's context, enabled by tracking the tag's keypoints. These visuals represent a default 120-pixel buffer applied around the keypoint coordinates. **d**, When two mice simultaneously approach the same occlusive area (t) and become fully occluded ($t+100$), tracking errors can occur (b, $t+150$). In contrast, leveraging the ArUco tag validation allows the system to re-identify the correct mouse (a, $t+150$).

1.7 PrecisionTrack's generalizability

In this study, we have focused on demonstrating PrecisionTrack's tracking reliability on mice, the most studied animal in neuroscience. However, there is a growing demand for a practical tool capable of facilitating longitudinal studies

of social dynamics across a wide range of species, including various monkey families, farm and domestic animals, as well as various exotic species. To evaluate PrecisionTrack's ability to meet these needs, we tested its accuracy in detecting and estimating poses for 30 additional species using the AP-10k validation set (Table 2; see Supplemental 1 for species-specific evaluations). Furthermore, we assessed its performance in tracking multiple socially interacting subjects from these species using APTv2's challenging testing dataset of 529 videos (Figure 6; see Supplemental 1 for species-specific evaluations).

Metric	**Score (%)**
Recall	86.8
Precision	82.0
F1 Score	83.9
Assignation's OKS	87.6
Assignation's PCK:0.2	97.5
Assignation's PCK:0.05	80.6

Table 2: Overall pose-tracking metrics performances on the AP-10k's validation set.

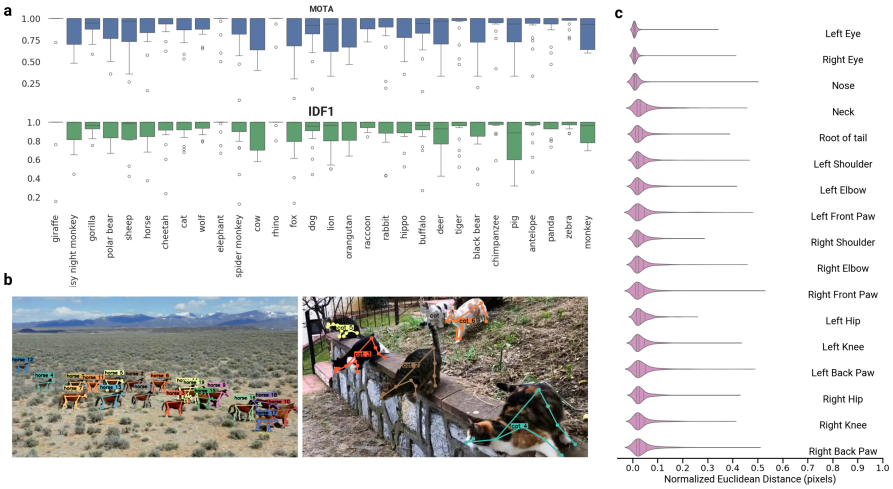

Figure 6: **a**, Distribution of Multi-Object Tracking Accuracy (MOTA) and IDF1 across APTv2's 529 testing clips for the 30 listed species. **b**, Qualitative assessment of pose-estimation precision for cats and horses. **c**, Distribution of keypoint prediction errors for our model on AP-10k's validation set, with distances normalized by animal size.

2 Discussion

Recent research suggests that mice evolving in enriched, social environments exhibit increased neuroplasticity and stress resilience. This raises the question: could study conducted in ethologically relevant conditions yield more reproducible and representative stress responses than those obtained in conventional artificial settings? This can only be tested in standardized, non-intrusive, automated methods to evaluate and compare behavioural responses in enriched environments over prolonged periods [19]. However, current behavioural neuroscience tools lack the capability to meet all necessary engineering requirements. While some approaches [4, 6] come close, none can reliably track multiple animals in complex, enriched and social settings in an online real-time manner while sustaining accuracy over prolonged periods.

Here we introduced PrecisionTrack, a real-time, online multi-animal pose tracking system showing accuracy sustainability over prolonged periods. The unique capabilities of our algorithm originate from important key contributions, including the implementation of a Transformer-CNN hybrid neural network able to predict cross-species classification, detection and pose-estimation in one forward pass. Leveraging large depth-wise separable convolutions, channel attention as well a tuned optimization techniques, RTMDet [13] achieved the current state-of-the-art performance in real-time object-detection and instance segmentation on the competitive Microsoft's COCO benchmark. On the other hand, recent work leveraging the Transformer architecture showed promising results in the field of object-detection RT-DETR [14]. Leveraging the high recall inducing context of RTMDet's anchor-free decoupled head and the fine-grained encoding of the last FPN features provided the RT-DETR's hybrid neck as well as the pose-estimation techniques proposed by [12]. We designed a powerfull yet fast cross-species classification, detection and pose-estimation algorithm, outperforming YOLO-pose [12] in both accuracy and speed on our MICE dataset.

We also proposed a novel, online, and flexible stitching algorithm, combined with significant enhancements to the vanilla Kalman Filter [17], to enable online tracking-by-detection (TBD) algorithms to effectively track a fixed number of subjects in complex environments. The vanilla Kalman Filter has been a cornerstone in state-of-the-art multi-object tracking (MOT) algorithms, particularly for applications such as tracking cars and pedestrians [11]. Since its adoption by early TBD algorithms like SORT [16], the vanilla Kalman Filter has undergone numerous modifications. One prominent adaptation is the Extended Kalman Filter (EKF), which extends the method to handle non-linear dynamics. However, the EKF is computationally expensive due to the calculation of Jacobian matrices and may struggle with highly non-linear dynamics, limiting its practical utility in certain scenarios. To address these limitations, we developed the novel Dynamic Kalman Filter (DKF). This approach explicitly models initial states and covariance for position, velocity, and acceleration based on Newtonian

dynamics. Additionally, it incorporates an Exponential Moving Average (EMA) for velocity updates, effectively reducing sensitivity to measurement noise. By leveraging these innovations, the DKF adapts the vanilla Kalman Filter to better accommodate top-down, real-time multi-animal tracking scenarios, where subjects frequently change both direction and velocity over short time frames.

Multi-object tracking (MOT) involves estimating the trajectories of targets and associating them across frames in a video sequence. The current state-of-the-art paradigm for MOT is tracking-by-detection (TBD), which separates the process into two distinct steps: detection and association [15]. Pose estimation, in contrast, focuses on localizing specific keypoints on a subject's anatomy, such as hands or elbows, in the context of human pose estimation. Pose estimation algorithms generally follow either a top-down or bottom-up approach (as described in Methods). Leveraging recent advancements in pose estimation [12], we overcame the limitations of these methods by inherently learning skeleton assembly through prior assignment during training (see Methods for details). Multi-animal pose tracking, popularized by recent works [5, 4], extends pose estimation to animals, enabling the analysis and tracking of fine-grained movements. This approach provides detailed insights into spatial configurations and behavioural patterns [6]. Unlike MOT, which often tracks pedestrians or vehicles from a side view, multi-animal pose tracking typically involves monitoring a fixed number of subjects, such as mice in a vivarium, from a top-down perspective. By incorporating both pose and localization information, we enhanced the robustness of ByteTrack's [10] association step, enabling reliable tracking of highly occluded objects, even when only a few body parts are visible. Combined with the other contributions described above, this allows us to track multiple individuals with greater accuracy than current latency/accuracy Pareto-dominant MOT algorithms [10], while incurring only a marginal increase in computational complexity. All of which allows us to present PrecisionTrack. We then tested on a challenging multi-animal pose tracking benchmark composed of 20 mice demonstrating systematic improvements in both MOTA and IDF1 tracking metrics compared to the current gold standard in the field [10].

Although these tracking accuracy metrics improvements look good on paper, they are practically useless to achieve our goal, as all three algorithms (PrecisionTrack, SLEAP and DeepLabCut) have the same blatant weakness. They cannot accurately track multiple mice for more than a few minutes, because the subjects are practically physically identical, tracking mistakes are not recovered and propagated through time. After a few minutes, most of the initially tracked trajectories are switched. To solved our problem, we leveraged an ergonomic, non-constricting fiducial tag called the Tailtags which have shown to be able to effectively identify individual mice for at least 7-days [9]. Leveraging these Tailtags and our AI driven ArUco detection context, we were able to maintain our initial level of accuracy through our whole 30 minutes of tests while still maintaining real-time latency, while our PrecisionTrack without such fiducial validation showed a significant IDF1 decline after a few minutes.

PrecisionTrack will be open-sourced and available to the community. It was created to be flexible and configurable, allowing for a multitude of tracking scenarios involving even multiple species at the same time. Simply configure a skeleton for each class and how many should be tracked in a given video. Our system showed well above real-time performances on 20 animals, indicating that the real-time subject upper limit should be much higher. We are also publishing a series of tools to help leverage our training, testing, benchmarking, tracking and saving APIs. We also open-source not only our highly challenging object-detection and pose-estimation MICE dataset, but also a huge sequential dataset containing millions of manually curated poses and actions labels, enabling further advancements in both the field of multi-animal pose tracking and animal action recognition, which we'll next focus on.

3 Methods

3.1 MICE datasets and benchmarks

Our pose-estimation dataset consists of over 1,500 images (1536x1536 resolution) featuring 20 mice. The images are from ethological experiments conducted in the laboratory of B. Labonté at CERVO in 2022. Each localization and skeleton (Figure 2) were manually labelled internally. All animals' procedures followed the Canadian Guide for the Care and Use of Laboratory Animals and were approved by the Animal Protection Committee of Université Laval.

In addition to this dataset, we compiled a sequential dataset comprising 35 minutes of recording at 28 Hz for a total 58,800 frames of 2820x2820 pixels resolution, which correspond to approximately a million localizations, detections, and actions since the experiments were conducted on 20 mice simultaneously interacting in an enriched open field. The labelling of mice localizations follows the COCO standard, with each instance annotated by bounding boxes and keypoints. The bounding boxes are defined by ρ, while the keypoints are represented by 8 (x, y) coordinates corresponding to the snout, right ear, right leg, left ear, centroid, tail base, and tail tag, respectively.

$$\rho = [x1, y1, w, h] \tag{1}$$

To complement these datasets, we provide three manually labelled evaluation benchmarks, each consisting of roughly one minute of footage with varying numbers of mice: 10, 15, and 20. These recordings were captured during additional ethological experiments conducted in B. Labonté's laboratory at CERVO, with a frame rate of 28Hz and a resolution of 1536x1536 pixels.

3.2 Detection step

PrecisionTrack follows the tracking-by-detection (TBD) paradigm, where object-detection serves as the first step [16]. In this stage, raw RBG images are processed by a neural network in order to predict relevant detections and poses (Figure 2c), tasks referred to as object-detection and pose-estimation respectively.

3.2.1 Neural network architecture

Our neural network architecture integrates recent advancements in real-time object detection to balance computational efficiency and accuracy. The CSPNeXt block (Figure 1b), our network's backbone, introduced by [13], utilizes large kernel convolutions to expand the model's effective receptive field, capturing broader spatial context beneficial for dense prediction tasks like object detection and pose estimation [20]. This large kernel approach inherently strengthens the model's capability to understand spatial relationships across larger image regions by increasing its local biases range. However, it also increases the layer's computational demands due to the higher number of multiply-accumulate (MAC) operations it requires over smaller kernels. However, recent studies have demonstrated that depthwise separable convolutions can achieve larger receptive fields efficiently by performing spatial convolutions across channels and resolutions separately [21]. As such, our backbone adopts this approach by using large 5×5 depthwise separable convolutions to maximize the receptive field while maintaining a reasonable computational cost. However, since depthwise separable convolutions involve both depthwise and pointwise operations, stacking them can effectively double the network depth, potentially impeding parallelization. Therefore, the backbone architecture was adjusted by moderately increasing the filter count in each layer, thus widening the network and reducing the stacked block count from 9 to 6. This configuration yields an optimal computation-accuracy trade-off by decreasing latency without compromising detection performance. To further enhance feature selectivity, a Channel Attention layer was added at the end of every block (Figure 1b), refining the model's focus on the most informative features within the newly widened layers output, thereby optimizing both the accuracy and efficiency of the detection pipeline.

Additionally, inspired by RT-DETR [14], we integrated a hybrid encoder comprising an Attention-based Intra-scale Feature Interaction (AIFI). This AIFI layer processes the last or highest-level feature map output from the CSPNeXt backbone through a lightweight Transformer encoder, enriching this high-level representation by capturing connections among semantic concepts. By leveraging AIFI, the model gains a more comprehensive understanding of object relationships, effectively reasoning about complex spatial and semantic interactions within the feature space. This enriched representation is then fused with the two lower-level feature maps using a lightweight FPN neck called Cross-scale Feature Fusion (CCFF). Leveraging this hybrid encoder architecture, paired with a ResNet backbone [22], [14] achieved a state-of-the-art computation-accuracy balance on the COCO dataset.

3.2.2 Multi-subjects object detection

Following recent trends in object detection [23, 13], we implemented an anchor-free detection framework with decoupled heads. The three heads independently regress the detection's bounding boxes ($\frac{P}{2}$, $\frac{P}{2}$, 4), detection scores ($\frac{P}{2}$, $\frac{P}{2}$, 1), and classification scores ($\frac{P}{2}$, $\frac{P}{2}$, C), where P represents the predefined number of priors and C the number of classes. Each head adopts the anchor-free paradigm [24], treating the width (W) and height (H) of each input feature map from the feature pyramid as priors. The feature maps have dimensions (20×20), (40×40), and (80×80), and since the input dimensions are symmetrical, we can express the priors as $\frac{P_f}{2} = H_f = W_f$. Thus, each stage of the feature pyramid produces 400, 1600, and 6400 priors, with strides of 32, 16, and 8, respectively. The network regresses the bounding box coordinates (cx, cy, w, h) relative to each prior's top-left corner (Figure 2c), positioning them within the image by adding the prior's top-left coordinates and scaling based on the prior's stride. This results in a total of 8,400 predictions per image. However, this approach generates numerous redundant and overlapping detections (Figure 2d). To mitigate this, we utilize a fast and configurable Non-Max Suppression (NMS) algorithm, which filters out bounding boxes that overlap by more than 65% (Figure 2d).

3.2.3 Multi-subjects pose-estimation

Pose estimation algorithms typically follow either a top-down or bottom-up approach. Top-down (or two-stage) methods are generally more accurate, as they first detect all relevant subjects in the image and then perform single-individual pose estimation for each detection. However, the computational cost of these methods scales linearly with the number of processed individuals, leading to variable runtime. This makes them unsuitable for scenarios like ours, where the goal is to track large groups of subjects simultaneously and in real-time. In contrast, bottom-up approaches offer constant runtime, as they detect all keypoints in a single pass and then group them into individuals through computationally expensive post-processing steps, such as Part Affinity Fields (PAFs) [25]. To overcome these limitations, we leverage recent advancements in both object detection [24, 23] and pose estimation [12], extending our object detector to support pose estimation while maintaining a constant runtime and without adding additional post-processing. Specifically, we added two decoupled anchor-free heads to our network. These heads, with shapes ($\frac{P}{2}$, $\frac{P}{2}$, $2 \cdot K$) (Figure 2a) and ($\frac{P}{2}$, $\frac{P}{2}$, K), respectively regress the deltas for each keypoint relative to their corresponding prior and predict their confidence levels (Figure 2c). By "flipping the problem on its head," we directly regress each prior's keypoints, inherently linking them. This approach thus directly resolves the keypoint-to-individual problem, eliminating the need for additional post-processing. As a result, we extend our object detector to support pose estimation with only marginal in-

creases in computational cost at runtime.

3.2.4 Model training

During the training process, predictions for each stage of the feature pyramid are matched with ground truth through the SimOTA [23] dynamic label assignment strategy. Each decouple head has its own loss function to optimize: (1) The classification head optimizes the Quality Focal (QFL) [26] Loss (Equation 3). (2) The bounding box regression head optimizes the General Intersection over Union (GIoU) [27] loss (Equation 27). (3) The keypoint regression head optimizes the Object Keypoint Similarity (OKS) [28] loss (Equation 2). (4) Both the detection and keypoints confidence regression heads optimizes the Binary Cross Entropy (BCE) loss. As such, the final loss function is composed of a weighted sum of each decoupled head's loss, as defined in Equation 4.

$$OKS_Loss = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{exp(-\frac{d_i^2}{2 \cdot s^2 \cdot k_i^2}) \cdot p_i}{\sum_{j=1}^{N} p_j} \tag{2}$$

Where:

- N: Total number of keypoints.
- $d_i = \|gt_i - pr_i\|_2$
- $\sigma = \sqrt{h \cdot w}$
- k_i: Weight of the i-th keypoint.
- p_i: 1 if the i-th ground truth is present, else 0.

$$QFL_{Loss} = -|y - \sigma|^{\beta} \cdot ((1 - y)\log(1 - \sigma) + y\log(\sigma)) \tag{3}$$

Where:

- σ: The model's prediction.
- y: The ground truth.
- β: Down-weighting rate's hyperparameter (set to 2).

$$\text{Total Loss} = 2 \cdot QFL_{Loss} + 5 \cdot GIoU_{Loss} + 30 \cdot OKS_{Loss} + 1 \cdot BCE_{Loss} + 1 \cdot BCE_{Loss} \tag{4}$$

Our two-stage training process closely follows [23]'s. Its first stage uses strong data augmentations, Mosaic and MixUp (Figure 2b), to teach the model that relevant subjects could be anywhere and at any scale. This step is crucial for minimizing common biases found in human-taken images, where objects of interest are often centered, non-occluded, and fully in the frame. Leveraging this step, we minimize the risk of the model learning these biases rather than the actual discriminative features necessary to accurately localize relevant subjects in real-world conditions [13]. On the other hand, the second stage leverages weaker geometric augmentations, such as cropping and flipping, to fine-tune the model for the specific task at hand.

We adopt a hyperparameter-free cosine learning rate schedule, preceded by a brief warm-up period. Given our limited computational resources, a single RTX 3090, we utilized the linear learning rate scaling approach proposed by [29]. This allowed us to effectively replicate the performance benchmarks of YOLOX [23] on the COCO dataset [28] while using a fraction of the computational power.

Hyperparameter	**Value**
Optimizer	AdamW
Scaled learning rate	0.00119
Weight decay	0.05
Momentum	0.9
Batch size	76
Learning rate schedule	Cosine
Training epochs	300
Warm up iterations	75
Input size	640×640
EMA decay	0.9998

Table 3: Training hyperparameter settings.

3.3 Model evaluation

An optimal multi-animal pose tracking algorithm's detection step should prioritize three core qualities: 1) high detection coverage—capturing as many subjects as possible in each frame, 2) accurate pose inference for each detected object, and 3) low latency to support real-time tracking. While the standard COCO metrics [28] technically encompass these qualities, they can be less intuitive than other widely used metrics. Therefore, we report additional, flexible metrics with greater interpretability.

Specifically, we evaluated our model based on a weighted sum between the detection's F1 Score and the detected subject's Object Keypoint Similarity (OKS) metric (Equation 15). Subjects are deemed correctly detected when matched to labels using an IoU-based cost function (Equation 24) with the Hungarian Algorithm [30].

For pose-estimation accuracy, we prioritize OKS and PCK over RMSE, commonly used in other methods [4]. As both OKS and PCK are normalized by subject size, reducing the impact of regression errors on larger subjects. However, we still report RMSE scores for interpretability and to facilitate direct comparisons with methods that primarily rely on RMSE [4].

$$\beta \cdot \text{F1 Score} + (1 - \beta) \cdot \text{OKS}, \tag{5}$$

$$\text{where} \quad \beta = 0.5 \tag{6}$$

3.4 Association step

The second step of the tracking-by-detection (TBD) paradigm is the association step [16]. Here, the detected entities from frames are sequentially linked to trajectories, or tracklets, with unique identification. This process is circular, meaning that a detection step will always precede an association step until the end of the video (Figure 3a).

While previous TBD approaches often address the trade-off between true positive and false positive detections by eliminating low-confidence detections, as these could represent noise or background, others argue that Low confidence detection boxes sometimes indicate the existence of objects, the occluded. Consequently, filtering out all low-confidence detections could disrupt the tracking of occluded objects, leading to unnecessary tracking errors [10].

While previous multi-animal pose tracking algorithms favoured the elimination of low-confidence detections, we choose to consider them during our association process. This choice incurs the cost of conducting two matching steps: the first matches confident detections to their respective tracklets, while the second matches the less confident detections to the remaining trajectories. Leveraging this second association step, PrecisionTrack able to track subjects even through partial occlusion (Figure 3c). Empirically, we found that for our experimental setup, particularly the top-down view provided in the MICE dataset, a high-confidence threshold β of 0.5 yielded optimal performance.

3.4.1 Projection Substep

Most current state-of-the-art TBD algorithms split the association step into two substeps: the projection substep and the matching substep. In the projection substep, motion estimation models are leveraged to project tracklet positions into the current frame, facilitating the eventual detections-to-tracklets matching [16]. Among these models, the Kalman Filter [17] remains the preferred choice due to its efficiency, reliability, and low computational cost. Especially in the multi-object tracking (MOT) community, whose benchmarks mainly include pedestrians and vehicles. The Kalman Filter's assumptions of linear motion and constant velocity align well with the typical movement patterns of these objects, such as pedestrians walking along sidewalks or vehicles following lanes.

However, these assumptions break down in multi-animal pose tracking, in which most tracking scenarios involve a top-down view of an open field where animals roam freely, thus often quickly change direction and velocity. We argue that this significant shift in data dynamics imposes a corresponding shift in the leveraged motion projection techniques to better handle the now frequent and abrupt changes in both direction and velocity. To bridge this gap, we developed a simple, yet effective motion algorithm called the Dynamic Kalman Filter (DKF), which leverages Newton's laws of motion to project tracklet dynamics more accurately in case of abrupt changes in direction or velocity.

The DKF updates its parameters at the end of each association step, where sequential detections, here qualified as measurements, allow for a straightforward inference of tracklet velocity based on Newton's first law of motion:

$$v_{t+1} = \frac{x_{t+1} - x_t}{\Delta t} \tag{7}$$

For numerical stability, given the frequent directional changes observed in animals, an exponential moving average (EMA) with a smoothing factor α of 0.1 is applied to the velocity estimate:

$$v_{EMA} = \alpha v_t + (1 - \alpha) v_{t-1}, \tag{8}$$

$$\text{where} \quad \alpha = 0.1 \tag{9}$$

This smoothed velocity estimate is then used to infer tracklet acceleration using Newton's second law of motion:

$$a_t = \frac{v_{t+1} - v_t}{\Delta t} \tag{10}$$

To further accommodate the complexities of animal tracking, we also adjusted the Kalman Filter's noise estimation. We set the initial state of covariance matrix P to 1000 times the identity matrix, representing high uncertainty in the absence of prior knowledge about the tracklet's state. P is updated at every time step using Equation 13. The process noise covariance Q is scaled by 0.1 and dynamically adjusted based on the time step Δt. This models the position noise growing quadratically and the velocity noise linearly in respect to eventual delays between measurements, allowing us to account for potential broken trajectories. The measurement H and measurement noise covariance R are both set to diagonal matrices with values of 1.0, assuming equal noise in both the x- and y-axes.

$$S = H \cdot P \cdot H^T + R \tag{11}$$

$$K = P \cdot H^T \cdot S^{-1} \tag{12}$$

$$P = (I - K \cdot H) \cdot P \tag{13}$$

By incorporating Newtonian dynamics and refining the Kalman Filter's noise estimation methods, the DKF enables more precise trajectory projections in open-field, top-down view setups, as demonstrated in our experiments (Figure 4a). This approach also allows us to add velocity and acceleration information to our tracklet's parametrization Ω for both the x- and y-axes:

$$\Omega = [cx, cy, \dot{cx}, \dot{cy}, \ddot{cx}, \ddot{cy}] \tag{14}$$

3.4.2 Matching Substep

The matching substep involves linking the projected tracklet locations to the current detections. To achieve this, we employ the Hungarian algorithm, a widely used method in such scenarios. Typically, matching costs are defined by the Intersection over Union (IoU) (Equation 24) between the bounding boxes of detections [27]. However, we argue that relying solely on bounding boxes overlooks critical information that could lead to tracking errors. For instance, two tracklets may overlap and exhibit similar IoU scores with newly acquired detections; however, one subject may be facing left while the other is facing right. Incorporating additional pose information can enhance the robustness of the association step, as finer-grained pose data allows for better differentiation between closely interacting tracklets. To address this, we utilize a slightly modified version of the Object Keypoint Similarity (OKS) metric [28], which is originally designed to evaluate pose estimation by measuring the proximity between predicted and labelled keypoints. Our implementation of OKS normalizes the Euclidean distances d (Equation 16) by the areas of the tracklets A.

The exponential term in the OKS calculation is also multiplied by the weight of each detection w.

$$OKS(p,t) = \frac{1}{K}\sum_{i=1}^{k} w_i \cdot exp(-\frac{d_i}{\sqrt{A_i}}) \tag{15}$$

$$\text{where} \quad d_i = \|p_i - t_i\|_2 \tag{16}$$

$$\text{and} \quad A_i = h_i \cdot w_i \tag{17}$$

By combining a weighted sum of the IoU and OKS metrics with a *lambda* of 0.2, we achieved more precise matching costs (Figure 3e) while retaining the advantages of the IoU metric, such as the ability to easily identify non-overlapping tracklet-detection pairs.

$$C = (1 - \lambda \cdot IoU) + (\lambda \cdot OKS) \tag{18}$$

3.5 Tracklet stitching

Tracklet, or trajectory, stitching is an optional plugin in the tracking pipeline. It allows the reconstruction of sparse trajectories for specific classes, for which all the subjects remains within the camera's field of view throughout the whole recording. For example, if a user wishes to track n marmosets (*Callithrix jacchus*) in their home-cage while also counting the number of new researchers entering the camera's field of view, they can configure the stitching plugin to only consider the n marmosets. This ensures that whenever a new marmoset is detected (e.g., emerging from a hideout), the algorithm attempts to link it to a previously existing marmoset trajectory instead of considering it as a new tracklet, preventing an incorrect increment of the marmoset count.

While offline methods such as DeepLabCut leverage the fact that they can perform tracklet stitching at the end of the entire recording by employing clever techniques such as using the network flow minimization algorithm on a directed acyclic graph representing the trajectories to elegantly link them over time. On the other hand, real-time and online methods like PrecisionTrack do not have the luxury of knowing the global trajectories for the whole recording. Instead, whenever a tracklet is lost, they must rely solely on past information to try to stitch it to an eventual new detection on the fly.

Working within this constraint, we developed a fast and flexible heuristic algorithm called Search-Based Stitching (SBS) that delivers strong practical

performance. SBS defines a dynamic search zone around the last known position of a lost tracklet. The search zone expands over time according to the lost tracklet's last recorded EMA velocity. To minimize the risk of excessively expanding the search zone, which could lead to a higher rate of false-positive matches, the rate of expansion decreases according to a configurable exponential decay (Equation 19).

$$v_{t+1} = v_t \cdot e^{-\beta x} \tag{19}$$

Additionally, the search zones expand asymmetrically, guided by each side cosine similarity with the lost tracklet's last recorded EMA 2D direction (Equation 23).

$$\cos\theta = \frac{\text{sides} \cdot \text{velocity}}{\text{norm}_{\text{sides}} \cdot \text{norm}_{\text{velocity}}} \tag{20}$$

$$\text{where} \quad \text{norm}_{\text{sides}} = \|\text{sides}\|_2 \tag{21}$$

$$\text{and} \quad \text{norm}_{\text{velocity}} = \|\text{velocity}\|_2 \tag{22}$$

$$\text{cosines} = \frac{\cos\theta + 1}{2} \tag{23}$$

This biases the search towards the area the lost tracklet is most likely heading for (Figure 4b). To account for the frequent use cases where multiple search zones are active and/or multiple detections occurs at the same time step, we cast the search zone-to-new detection linking as an assignment problem, which we solve using the Hungarian algorithm [30]. The cost matrix for the association is based on our implementation of a normalized Generalized Intersection over Union (GIoU) (Equation 29) between each active search zone and each new relevant detection.

$$\mathrm{IoU}(A, B) = \frac{\mathrm{Area}(A \cap B)}{\mathrm{Area}(A) + \mathrm{Area}(B) - \mathrm{Area}(A \cap B) + \epsilon} \tag{24}$$

$$\text{where} \quad A = [\mathrm{acx} - \frac{\mathrm{aw}}{2}, \mathrm{acy} - \frac{\mathrm{ah}}{2}, \mathrm{acx} + \frac{\mathrm{aw}}{2}, \mathrm{acy} + \frac{\mathrm{ah}}{2}] \tag{25}$$

$$\text{and} \quad B = [\mathrm{bcx} - \frac{\mathrm{bw}}{2}, \mathrm{bcy} - \frac{\mathrm{bh}}{2}, \mathrm{bcx} + \frac{\mathrm{bw}}{2}, \mathrm{bcy} + \frac{\mathrm{bh}}{2}] \tag{26}$$

$$\mathrm{GIoU}(A, B) = \mathrm{IoU}(A, B) - \frac{\mathrm{Area}(\text{enclosing box}) - \mathrm{union}(A, B)}{\mathrm{Area}(\text{enclosing box}) + \epsilon} \tag{27}$$

$$\text{where} \quad \text{enclosing box} = \min(A_1, B_1), \max(A_2, B_2) \tag{28}$$

$$\text{normalized GIoU} = \frac{\mathrm{GIoU}(A, B) + 1}{2} \tag{29}$$

3.6 ArUco validation

Previous research has demonstrated that detecting ArUco markers at full image resolution is computationally expensive and limits the ability to achieve real-time latency. To address this issue, we leveraged the fact that we are already tracking the poses and developed an AI-driven detection context for ArUco identification, which is utilized in our ArUco validation plugin in conjunction with an ergonomic, non-constricting fiducial tag called the Tailtags.

In this system, AI-detected tag positions are added to a priority queue, sorted by the confidence levels of each tracklet's detections. Only tag positions with a higher confidence level than the γ threshold are processed. The algorithm then iterates through either all those confident tag positions, or halts if the computational time exceeds a predetermined timeout threshold μ. During each iteration, the algorithm calculates the cropped ArUco detection's context by selecting 120-pixels (configurable) around the predicted tag's position, then read the encompassed ArUco marker (Figure 5c). Classification metrics for each marker ID on each tracklet are dynamically recorded to assess reliability online. This approach ensures that only genuine switches between tracklets and not sporadic detection errors are flagged, as the system waits until it as gathered sufficient evidence before indicating a tracking error. The validation system then recursively corrects these tracking mistakes, enabling online validation and correction of potential switches. A total of 42 corrections were performed accross the three 10-minute recordings, with a median error-to-correction delay 4.76 seconds (Figure 5b).

Paired with this ArUco validation plugin, PrecisionTrack is able to maintain reliable tracking performances over extended periods (Figure 5a), while remaining computationally efficient enough to ensure that processing the whole pipeline remains real-time for at least 20 concurrently tracked subjects using our hardware (see Computational Resource Usage for more details).

3.7 Evaluation

Tracking performances are assessed using the CLEAR metrics [18], which have been the standard for multi-object tracking (MOT) evaluations for over a decade. Specifically, we employ the Multiple Object Tracking Accuracy (MOTA) metric (30), which quantifies three main types of errors that can occur during tracking:

1. False Positives (FP): Instances where the algorithm incorrectly tracks irrelevant objects.

2. False Negatives (FN): Instances where the algorithm fails to track relevant objects.

3. Identity Switches ($IDSW$): Instances where the algorithm misidentifies two trajectories, resulting in swapped identities.

As such, MOTA focuses on assessing the overall quality of the tracking algorithm:

$$\text{MOTA} = 1 - \frac{\sum_{t=1}^{T}(FP_t + FN_t + IDSW_t)}{\sum_{t=1}^{T} O_t} \tag{30}$$

Where:

- FP_t: Total number of false positives at time t.
- FN_t: Total number of false negatives at time t.
- $IDSW_t$: Total number of identity switches at time t.
- O_t: Total number of visible objects at time t.
- T: Total number of frames.

Additionally, we utilize the IDF1 score (31), which focuses solely on evaluating the algorithm's ability to maintain the correct identities of tracked objects over time:

$$\text{IDF1} = 1 - \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{31}$$

3.8 Computational Resource Usage

All tests were conducted on a machine equipped with 32 GB of SSD storage, an Intel i7-10700k CPU, and an RTX-3090 GPU. The PrecisionTrack algorithm requires approximately 4 GB of RAM and 2.5 GB of VRAM for execution. Additionally, saving the tracking results incurs a disk space cost of about 7.3 MB per minute when monitoring 20 subjects.

3.9 Acknowledgements

This work is made possible by the MMLab's open-source MMPose toolkit [31]. We also thank Arturo Marroquín Rivera and Sam Mansouri for their comments on both the methods and the manuscript.

References

[1] Elizabeth A. Buffalo, J. Anthony Movshon, and Robert H. Wurtz. "From basic brain research to treating human brain disorders". In: *Proceedings of the National Academy of Sciences* 116.52 (2019), pp. 26167–26172. DOI: 10.1073/pnas.1919895116. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1919895116. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1919895116.

[2] J. P. Bennett and W. Akram. "A review of RFID technology and its applications in animal tracking". In: *The Veterinary Journal* 198.1 (2013), pp. 1–9. DOI: 10.1016/j.tvjl.2013.05.003.

[3] B. König, A.K. Lindholm, and P.C. Lopes. "A System for Automatic Recording of Social Behavior in a Free-living Wild House Mouse Population". In: *Animal Biotelemetry* (2015). DOI: 10.1186/s40317-015-0069-0.

[4] J Lauer, M Zhou, and S et al. Ye. "Multi-animal pose estimation, identification and tracking with DeepLabCut." In: *Nature Methods* 19 (2022), pp. 496–504. DOI: https://doi.org/10.1038/s41592-022-01443-0.

[5] Talmo D Pereira et al. "SLEAP: A deep learning system for multi-animal pose tracking". In: *Nature Methods* 19 (2022), pp. 486–495. DOI: https://doi.org/10.1038/s41592-022-01426-1. URL: https://www.nature.com/articles/s41592-022-01426-1.

[6] Markus Marks et al. "Deep-learning-based identification, tracking, pose estimation and behaviour classification of interacting primates and mice in complex environments". In: *Nature Machine Intelligence* 4.4 (Apr. 2022), pp. 331–340. ISSN: 2522-5839. DOI: 10.1038/s42256-022-00477-5. URL: https://doi.org/10.1038/s42256-022-00477-5.

[7] James D. Crall et al. "BEEtag: A Low-Cost, Image-Based Tracking System for the Study of Animal Behavior and Locomotion". In: *PLOS ONE* 10.9 (Sept. 2015), pp. 1–13. DOI: 10.1371/journal.pone.0136487. URL: https://doi.org/10.1371/journal.pone.0136487.

[8] Scott W. Wolf et al. "NAPS: Integrating pose estimation and tag-based tracking". In: *Methods in Ecology and Evolution* 14.10 (2023), pp. 2541–2548. DOI: https://doi.org/10.1111/2041-210X.14201. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.14201. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.14201.

[9] Vincent Coulombe et al. "The Tailtag System: Tracking Multiple Mice in a Complex Environment Over a Prolonged Period Using ArUco Markers". In: *bioRxiv* (2024). DOI: 10.1101/2024.11.07.622536. eprint: https://www.biorxiv.org/content/early/2024/11/08/2024.11.07.622536.full.pdf. URL: https://www.biorxiv.org/content/early/2024/11/08/2024.11.07.622536.

[10] Yifu Zhang et al. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. 2022. arXiv: 2110.06864 [cs.CV]. URL: https://arxiv.org/abs/2110.06864.

[11] Laura Leal-Taixé et al. *MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking*. 2015. arXiv: 1504.01942 [cs.CV]. URL: https://arxiv.org/abs/1504.01942.

[12] Debapriya Maji et al. *YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss*. 2022. arXiv: 2204.06806 [cs.CV]. URL: https://arxiv.org/abs/2204.06806.

[13] Chengqi Lyu et al. *RTMDet: An Empirical Study of Designing Real-Time Object Detectors*. 2022. arXiv: 2212.07784 [cs.CV]. URL: https://arxiv.org/abs/2212.07784.

[14] Yian Zhao et al. *DETRs Beat YOLOs on Real-time Object Detection*. 2024. arXiv: 2304.08069 [cs.CV]. URL: https://arxiv.org/abs/2304.08069.

[15] Yu-Hsiang Wang et al. *SMILEtrack: SiMIlarity LEarning for Occlusion-Aware Multiple Object Tracking*. 2024. arXiv: 2211.08824 [cs.CV]. URL: https://arxiv.org/abs/2211.08824.

[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: http://dx.doi.org/10.1109/ICIP.2016.7533003.

[17] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf. URL: https://doi.org/10.1115/1.3662552.

[18] Keni Bernardin and Rainer Stiefelhagen. "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics". In: *EURASIP Journal on Image and Video Processing* 2008.1 (2008), p. 246309. ISSN: 1687-5281. DOI: 10.1155/2008/246309. URL: https://doi.org/10.1155/2008/246309.

[19] David Grimm. "The happiness project". In: *Science* 359.6376 (2018), pp. 624–627. DOI: 10.1126/science.359.6376.624. eprint: https://www.science.org/doi/pdf/10.1126/science.359.6376.624. URL: https://www.science.org/doi/abs/10.1126/science.359.6376.624.

[20] Wenjie Luo et al. *Understanding the Effective Receptive Field in Deep Convolutional Neural Networks*. 2017. arXiv: 1701.04128 [cs.CV]. URL: https://arxiv.org/abs/1701.04128.

[21] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: https://arxiv.org/abs/1704.04861.

[22] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[23] Zheng Ge et al. *YOLOX: Exceeding YOLO Series in 2021*. 2021. arXiv: 2107.08430 [cs.CV]. URL: https://arxiv.org/abs/2107.08430.

[24] Zhi Tian et al. *FCOS: Fully Convolutional One-Stage Object Detection*. 2019. arXiv: 1904.01355 [cs.CV]. URL: https://arxiv.org/abs/1904.01355.

[25] Zhe Cao et al. *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2017. arXiv: 1611.08050 [cs.CV]. URL: https://arxiv.org/abs/1611.08050.

[26] Xiang Li et al. *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection*. 2020. arXiv: 2006.04388 [cs.CV]. URL: https://arxiv.org/abs/2006.04388.

[27] Gong Cheng and Junwei Han. "A survey on object detection in optical remote sensing images". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 117 (July 2016), pp. 11–28. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2016.03.014. URL: http://dx.doi.org/10.1016/j.isprsjprs.2016.03.014.

[28] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

[29] Priya Goyal et al. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour". In: *arXiv preprint arXiv:1706.02677* (2017).

[30] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2 (1955), pp. 83–97. DOI: https://doi.org/10.1002/nav.3800020109.

[31] MMPose Contributors. *OpenMMLab Pose Estimation Toolbox and Benchmark*. https://github.com/open-mmlab/mmpose. 2020.