Inferring cognitive strategies from groups of animals in natural environments

Ines Aitsahalia[*,1,2], **Thomas L. Botch**[*,1,3], **Shijie Gu**[*,1,4], **Thomas C. O'Connell**[*,1,5], **Rebecca S Siegel**[1], **Ralph E Peterson**[⋆,1], **Dmitry Batenkov**[⋆,1], **Emily L Mackevicius**[⋆,1]

[1]Basis Research Institute, [2]Columbia University, [3]Dartmouth College,
[4]UC Berkeley and UCSF, [5]Skidmore College
[*] co-first authorship, [⋆]co-senior authorship
[†]correspondence to `emily@basis.ai`

Abstract

Understanding multi-agent behavior in natural environments requires world models that jointly capture agent interactions, environmental structure, and the cognitive strategies that shape collective dynamics. Existing approaches typically focus on isolated components — such as tracking, simulation, or inference — without integration into a unified pipeline for reasoning about relationships between behavior, environment, and cognition. Here, we introduce new behavioral data from groups of birds foraging in outdoor 3D environments, as well as an open-source framework for interpreting this data. The framework combines detection and tracking of multiple animals, semantic 3D environment reconstruction, multi-agent simulation, and graph neural network–based inference. By bringing together behavior, context, and predictive modeling, this paper lays the groundwork for investigating latent cognitive strategies across species and environments, and identifies key challenges to interpreting models of these systems.

1 Introduction

Understanding how agents coordinate their movements and make decisions in real-world environments is a persistent challenge spanning the fields of neuroscience, behavioral ecology, and artificial intelligence. Collective behaviors such as flocking and foraging emerge from local interactions among individuals and between individuals and their surroundings. These processes embody distributed cognition: no single agent or environmental feature dictates the outcome, yet coordinated patterns arise that allow the group to navigate, exploit resources, and avoid threats. Developing a holistic account of such dynamics requires requires models that jointly represent animal movement, environmental context, and the latent behavioral cognitive strategies that include internal agent world models and inter-agent communication patterns.

Deciphering the rules and neural substrates underlying such group behaviors *en natura* is difficult given the unconstrained nature of behavior and environmental complexity in the wild. In absence of experimental control typically afforded by laboratory experimentation, how does one 1.) quantify structure in animal behavior and 2.) understand that behavior in the context of an animal's surroundings? Recent advances in machine learning have dramatically expanded the toolkit available for studying collective behavior in natural settings. Deep learning models now enable relatively robust multi-agent tracking from video data in varied conditions [27, 21, 58]. In parallel, rendering techniques allow for high-resolution reconstruction of complex 3D environments from sparse visual data [23, 64, 56, 33], offering new opportunities to study how agents interact with their surroundings in a spatially grounded manner.

Despite this important progress on isolated techniques, there remains no framework that unifies real-world tracking, environmental reconstruction, and predictive modeling to reason about collective decision-making in context (Figure 1). Statistical mechanics has revealed general principles of collective motion across species [24], GPS-based studies of wild primates have tied decision-making to habitat structure [50, 49], and laboratory neuroethology has shown how hippocampal maps encode sociospatial structure [35]. At the same time, AI research is beginning to build foundation models of ecosystems and global environmental dynamics [39, 7], as well as animal vocalizations [38, 14, 16]. Yet these advances remain siloed — they either prioritize behavior at the expense of environment, or focus on environment without resolving behavior. Our framework integrates these AI/ML domains into a single pipeline, increasing the scalability and richness with which it is possible to study collective intelligence *en natura*. This integration allows us to ask how behavior emerges from the interplay of social interaction and environmental affordances. As shown in a recent application to urban rat collectives [29], such integrative analyses can reveal latent principles of social cognition that would be invisible to approaches in isolation.

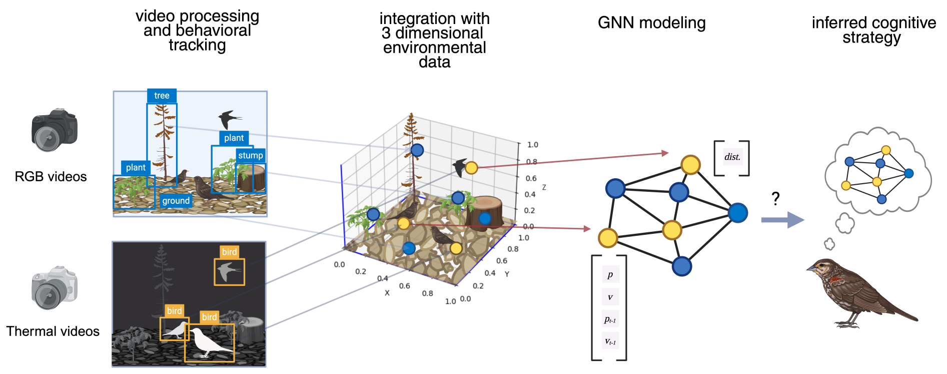

video processing and behavioral tracking
integration with 3 dimensional environmental data
GNN modeling
inferred cognitive strategy
RGB videos
Thermal videos
tree
plant
stump
ground
bird
dist.
?
p
v
p_{t-1}
v_{t-1}

Figure 1: **Our approach for inferring movement strategies.** Pipeline schematic. We capture multimodal behavioral data of group foraging to reconstruct a 3-dimensional mesh representation of the environment into which we can project the tracked data. This allows us to analyze multi-agent tracking in natural environments. We use this data (along with simulations of classic flocking models) to fit a graph neural network to estimate local interaction rules.

To model social group behavior, we used graph neural networks (GNNs), a class of deep learning model designed to operate on graph-structured data, where entities are represented as nodes and their relationships as edges[68, 62]. Unlike traditional neural networks that assume fixed input structures, GNNs flexibly model interactions and dependencies across arbitrary topologies, making them well-suited for systems where relational dynamics drive behavior. This framework is particularly relevant to our study of bird behavior, where collective actions such as flocking, foraging, and evasion emerge from decentralized interactions among individuals and with their environment. GNNs have previously been used to model flocking behavior [30, 67], but to our knowledge, not to distinguish between different flocking mechanisms through the attention mechanism. Other related works utilizes GNNs in various agent-based environments, [51, 2, 8, 70, 19, 42].

This manuscript introduces an open-source methodological pipeline that integrates real-world multi-agent tracking, semantic 3D environment reconstruction, and graph-based inference into a unified pipeline for studying collective behavior. To demonstrate the pipeline, we acquired a dataset of animal groups foraging in complex natural environments using synchronized thermal and RGB imaging, and developed a cross-species detection and tracking pipeline capable of resolving individuals across camera views. We further generated semantic 3D reconstructions via Gaussian splatting to capture spatial environmental context, and integrated real and artificial agents to probe the interplay between collective dynamics and environmental constraints. Finally, we constructed a GNN model with an attention mechanism to uncover latent interaction structure in multi-agent systems. We tested the GNN models on simulated flocks of birds with different foraging strategies, identifying key challenges and opportunities for interpreting such systems.

2 Results

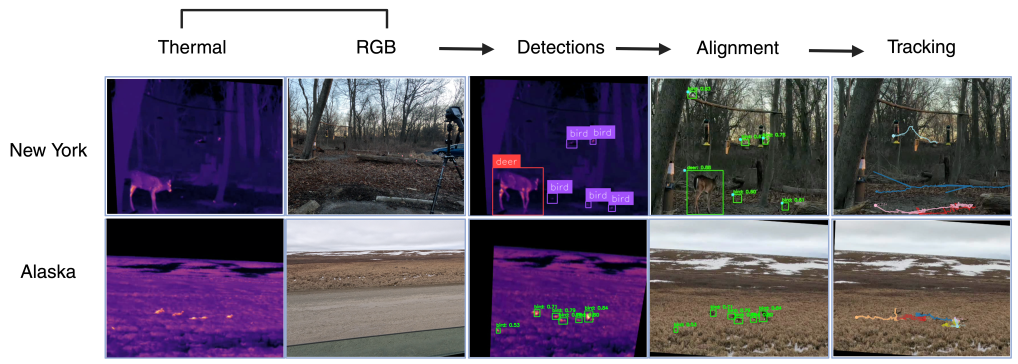


Figure 2: **Multi-animal detection and tracking in real world environments.** Tracking pipeline for example fieldwork videos of birds in Marshlands Conservatory, Westchester County (above) and Toolik, Alaska (below). Behavior was filmed with RGB and thermal cameras. These videos were used to train computer vision object detection models (YOLOv11, RF-DETR Base). Due to discrepancies between the field of view of different camera types, we developed a semi-automatic alignment pipeline to allow the bounding boxes to be cross-validated across the video modalities. These detections were then tracked with the ByteTrack algorithm.

2.1 Multi-animal detection and tracking in real world environments

We set out to observe the foraging behavior of unconstrained wild birds in a diversity of outdoor environments, including urban and suburban parks, wildlife refuges, and the Arctic tundra. We selected a range of locations in New York (Central Park, New York County; Marshlands Conservatory, Westchester County) and Alaska (Creamer's Field Migratory Waterfowl Refuge, Fairbanks AK; Toolik Lake in the Arctic Circle). Tripods were positioned approximately 5-100 meters from groups of birds, and recording sessions lasted as long as many birds remained in frame, typically between 5 minutes and 45 minutes. Next, images of the scene from many different camera angles were acquired using a GoPro camera mounted to a 9 ft carbon fiber pole. In each location, birds foraged for food in multi-species groups. We counted at least 32 unique species of birds across our datasets, as well as at least 6 species of mammals (listed in supplemental information). Outdoor environments can be challenging for object detection and tracking algorithms, so in addition to RGB videos, we collected thermal videos, where warm birds pop out against a cold background (Figure 2, left).

In order to extract the trajectories of animals within a scene, we first extracted bounding boxes of animal locations on each frame, aligning thermal and RGB field of views (Figure 2, middle). Multi-animal object detection models were trained on thermal videos using two architectures: YOLOv11 (Fast) [15] and RF-DETR (Nano) [40]. Both models were trained and evaluated on datasets hosted and annotated via the Roboflow platform [41], with manual bounding box annotations available. YOLOv11 was selected as the primary model due to its strong performance, efficient inference speed, and widespread adoption in real-time detection tasks. To validate this choice, we conducted a comparative evaluation against RF-DETR, a transformer-based architecture that was among the top-performing models at the time of experimentation. Models were fine-tuned on a dataset split into training (1,750 images), validation (276 images), and test (211 images) sets. Preprocessing steps included auto-orientation, resizing, and contrast stretching. Data augmentation was applied with two outputs per training example, incorporating randomized hue shifts and saturation adjustments. Fine-tuning increased performance substantially, with the fine-tuned YOLOv11 model achieving 97.4% precision, 90.9% recall, and 90.9% mAP@50, on held-out data. Detection performance on our datasets across different model architectures is summarized in Table 1.

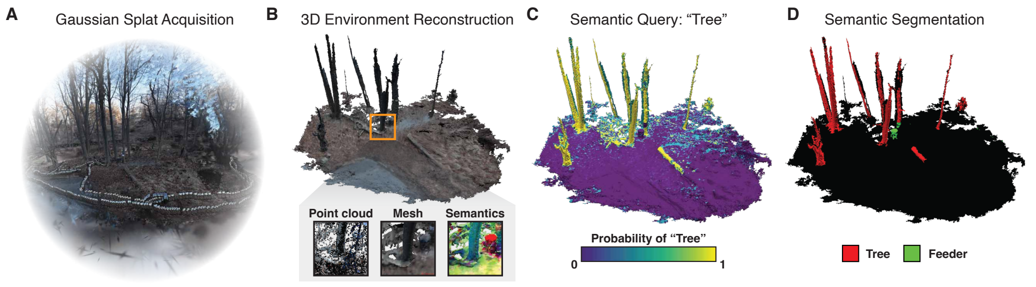


Figure 3: **Environment modeling.** (A) Rasterized view of a trained 3-D Gaussian splat (3DGS). RGB video frames used for model training are superimposed on the rasterized image. (B) Mesh surfaces of the environment were extracted from the 3DGS model. Inset: extracted environmental features including point-cloud, mesh, and semantic information. (C) Semantic representations enable queries of the environment through natural language. (D) Semantic queries provide the ability to segment the environment into discrete categories.

Bounding boxes were used as input to multi-object tracking performed using the ByteTrack algorithm [66], which associates detections across frames to generate unique track IDs and trajectories for each individual. Thermal and RGB field of views are aligned using manual keypoints, so that bounding boxes locations can be translated from thermal frames to RGB frames. Example tracks, superimposed on an RGB frame, are shown in Figure 2, right.

2.2 Environmental modeling with Gaussian splats and semantic segmentation

We used 3-D Gaussian splatting (3DGS) to generate a model of the environment at each fieldwork site (Figure 3A; [17]). 3DGS models provide an explicit representation of the scene by fitting parameters for a number of Gaussian spheres (e.g., position, color). We integrated recent advances in 3DGS rasterization, such as depth and normal map calculation, to create a smooth mesh surface of the scene [55, 63]. We also trained the 3DGS model to learn semantic features originally extracted from a vision-language model [32, 10]. We integrated these features into a multi-view representation of the environment (i.e., color, depth, semantics) and used the associated normal maps to map these representations to a smooth mesh surface (Figure 3B; [22]).

Critically, these semantic features allowed us to quantify the structure of the environment through natural language. Specifically, we used the original vision-language model to extract a text representation of each semantic query (e.g., tree, feeder). We then calculated the cosine similarity between each query vector and the semantic feature vector associated with each point within the environment. This process yields a continuous probability map (soft-max similarities) of the environment for each query (Figure 3C). Lastly, we applied semantic segmentation to the environment by thresholding each query map and spatially clustering these probabilities using DBSCAN (Figure 3D). Together, these semantic representations enabled us to capture environmental affordances – such as trees, feeders, and obstacles – directly within the spatial reconstruction, providing a substrate for modeling agent-environment interactions.

2.3 Integrating animal movements with environment representations

We wondered whether it was possible to integrate animal tracks and identities with the representation of their surrounding environment. To this end, we used scene structural information extracted during the COLMAP stage of the Gaussian splat training (i.e., camera poses, intrinsics) and localized the static cameras within the 3-D environment reconstruction [44, 47, 46]. Next, we validated the position of the camera by comparing the camera image of the mesh to the ground-truth video (Figure 4). We found moderately high levels of spatial and perceptual similarity ($SSIM = 0.718$; $LPIPS = 0.237$), suggesting that the mesh environment model provides a decent approximation of the scene. Lastly, we combined the previously extracted tracks with the localized camera position to estimate the position of the tracks on the mesh surface. Along with the mesh environment, these position estimates provide the primitives for understanding and simulating the behaviors of animals within their respective environments.

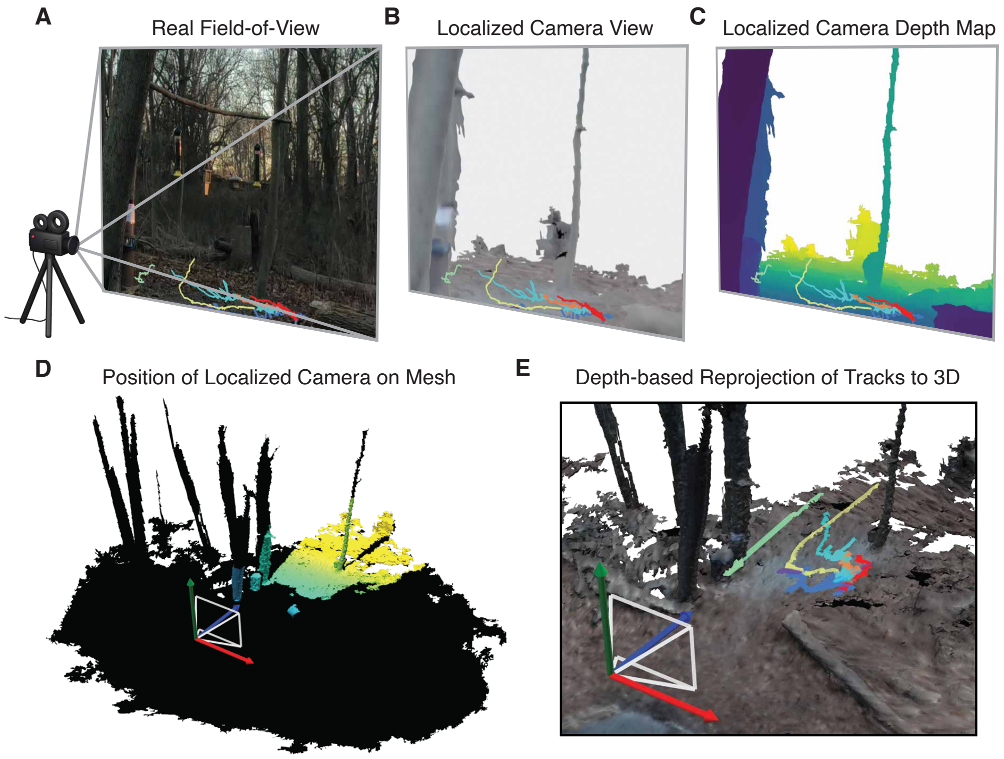
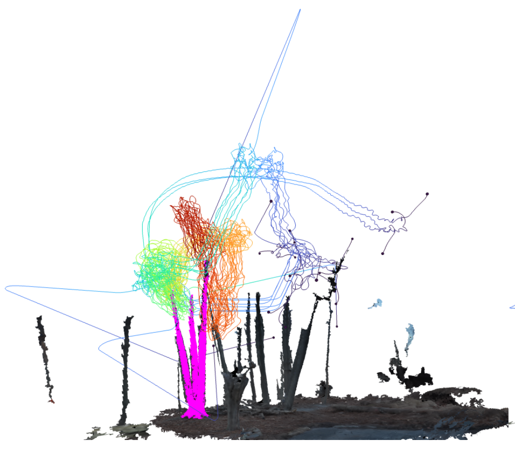
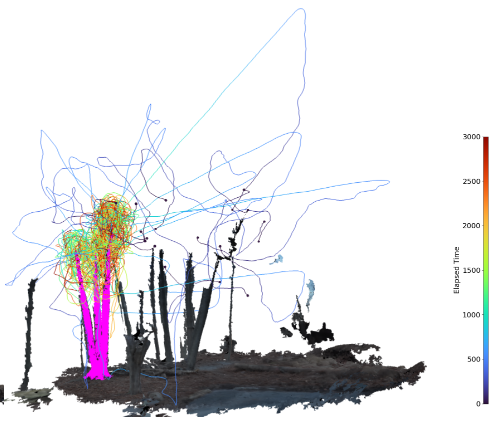

A
Real Field-of-View
B
Localized Camera View
C
Localized Camera Depth Map
D
Position of Localized Camera on Mesh
E
Depth-based Reprojection of Tracks to 3D

Figure 4: **Integrating tracking and environment models.** We localized the position of the tracking camera within the 3-D environment. (A) Ground-truth image from the perspective of the the RGB tracking camera. (B) View of mesh from the perspective of the localized camera. (C) Depth map indicating mesh distance from the position of the localized camera. (D) Position of the tracking camera on the mesh. Colors indicate distance of each mesh vertex from the view of the camera. Black indicates camera vertices not within the camera view. (E) Re-projected tracks within the environment. Depth information was integrated with 2-D image positions of each animal to estimate 3-D positions within the environment. Across all panels, colored lines denote tracks from different animals.



(a) Trajectory of Boids agents.



(b) Trajectory of independent agents.

Figure 5: **Simulating groups of agents within 3D natural scenes.** The trajectories are shown for (A) Boid agents and (B) agents that act independently of other agents. The simulation was run for 3000 time steps, with the target appearing after 500 time steps. The colors of the tracks move from blue to red as time progresses. The pink tree with three trunks is the target.

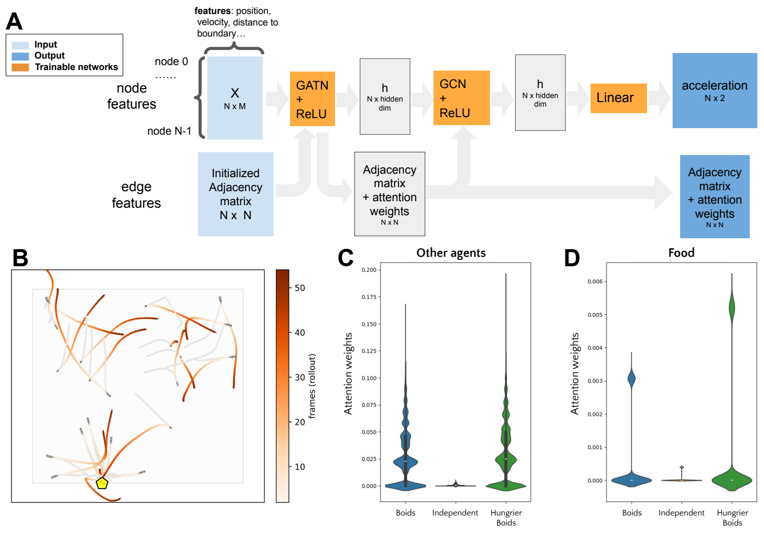


Figure 6: **Inferring movement strategies with graph neural networks. A**, Schematic of GNN model architecture, showing inputs, outputs, and trainable network layers. **B**, 52 frames of a 2D Boid simulation with walls and a food source (yellow pentagon), used to train a GNN model. Boid simulation shown in gray. GNN roll-out shown in red. The first 5 frames (dark gray) serve as input to the GNN rollout. **C**, Distributions attention paid to other agents in GNNs fit on different simulated models. **D**, Distributions of attention paid to food for GNNs fit on different simulations.

2.4 Generating simulated trajectories to test inference methods

We built a multi-agent simulator using Gymnasium [54] and Open3D [69] to generate synthetic data for testing inference methods. Synthetic data provides a more controlled testing platform because the agents' strategies are known and controlled with a small finite set of parameters. By choosing a standard, open source, reinforcement learning platform like Gymnasium, the simulator is capable of generating data of many types of agent strategies. In our initial simulations, agents move either independently or by following a variant of the Boid model [36] for collective movement (details in Methods). The simulator uses a 3D reconstruction of the environment derived from Gaussian splats (Figure 3). This 3D reconstruction constrains and influences the agents' movements, providing obstacles to avoid and targets to move towards. Figure 5 shows the tracks of agents who are drawn toward a particular tree in the environment (i.e. "food"), with Figure 5a showing agents following Boid rules and Figure 5b showing agents following a random walk. Both types of agents have a weak attraction to the target tree starting 500 steps into the simulation. The independent agents are much more scattered initially and move toward the target more quickly once the target becomes active. Since Boids cluster in groups, constraints of group movements keep the Boids away from the target for longer. Even after finding the target, the Boids have a tendency to be pulled away from the target by the group behavior. This is partly the result of our collision avoidance constraints, which move the agents away from any part of the scene if they get too close. As some Boids get pushed away from the tree, other Boids will follow because the strength of their attraction to neighboring Boids is greater than the strength of their attraction to the target. Toward the goal of inferring agents' cognitive strategies, a key open question is how to distinguish between groups of agents that exchange information (e.g. Boids), compared to groups that act individually, but have common goals.

2.5 Inference of strategies with graph neural networks (GNNs)

It is challenging to assess whether or how agents influence each others' behavior in cases when they may have correlated behavior, e.g. due to approaching a common food source. We wondered whether we could capture the degree to which agents pay attention to each other by fitting a GNN with an attention mechanism. The GNN is trained to predict each agent's acceleration as a function of the previous velocities and positions of other agents, as well as environmental features such as a food source (Figure 5). For simplicity, we test the GNN on ground-truth data simulated in 2D, with a single food source (Figure 6B). Three types of simulated agents are considered: Boids, independent agents, and hungrier Boids, which have stronger attraction to the food source.

In order to assess the GNN fits, the GNNs were initialized with 5 frames of simulated data, then used to iteratively roll out subsequent frames of data. The rollout initially appears to match the true data, but often diverges substantially within 20 frames (Figure 6B). Since Boid models are frequently considered a system which is unpredictable over moderate time scales [13], it is not surprising that the rollouts diverge from the ground truth relatively quickly. In fact, a small perturbation of initial conditions yields similar divergence in simulated data (Figure 7 in supplemental info, section 6.4). Machine-learning and deep-learning prediction of chaotic systems is an active area of research, cf. [9, 43, 34, 48, 26, 31] and references therein.

We wondered to what extent we could recover properties of the ground-truth simulation from examining the GNN attention weights. GNNs fit on independent agents were compared to GNNs fit on Boid data. Indeed, the GNNs fit on Boid data exhibited stronger agent-to-agent attention weights than GNNs fit to independent agents (Figure 6C, $p < 0.0001$, Rank Sum Test comparing the distributions of attention weights across simulations). The distributions of GNN attention paid to food are concentrated very close to zero, but stronger values were observed for GNNs fit on hungrier Boids (Figure 6D). The near-zero values may be explained by the limited visual range of the agents, and/or by the fact that the food is static in this simulation, so could be learned in absolute coordinates, or by following other agents, and mostly ignoring the food node in the GNN graph. Overall, it was possible to recover some sensible differences between GNNs fit on the different simulated datasets, with some caveats, namely that the overall GNN fits diverged from the data on relatively short timescales, and that the attention to food was mostly near zero.

Finally, we wondered whether a GNN could capture aspects of real bird foraging behavior. We trained a GNN on one of our datasets (Sandhill Cranes foraging in Alaska), and this yields qualitatively reasonable rollout trajectories, compared to a GNN trained on Boids, which looks quite different from the real data (Figure 8 in supplemental info, section 6.5).

2.6 Code and data availability

We release our code as open-source in two separate repositories. Additional input videos, meshes and other data files are available upon reasonable request from the authors.

1. `https://github.com/BasisResearch/collab-splats/` - splatting and meshing generation and analysis (Figure 3).
2. `https://github.com/BasisResearch/collab-environment/` - video alignment and animal tracking (Figure 2), camera alignment and track reprojection into 3D (Figure 4), simulations (Figure 5), GNN modeling (Figure 6) and perturbation (Figure 7), real-data fit (Figure 8).

3 Discussion

3.1 Key contributions

Our work explores the possibility of uncovering strategies that subserve collective behavior by building a computational framework to go from raw videos of natural animal movement *en natura* to graph-based neural network representations of collective motion. The first step in this pipeline is multi-agent tracking, where it is difficult to achieve high-precision tracking of multiple agents in a scene. This is particularly challenging with standard RGB camera recordings of animals in the wild, where animals are often well-camouflaged with their environment and undetectable by computer vision algorithms. We circumvented this issue by recording thermal videos, where relatively warm animals contrast well against their lower temperature environment (Figure 2).

Understanding behavior in the wild requires not only robust tracking of individual agents, but also a detailed understanding of the environment in which the behavior occurs. Did a behavioral change occur because of information communicated from another agent, because of something in the environment (e.g. presence of food), or some combination of the two? A central challenge to realizing this vision lies in capturing behavior and quantifying environment in the wild with sufficient fidelity. Even with state-of-the-art AI tools, these remain challenging problems. Neural radiance fields (NeRFs) and their derivatives provide a powerful approach for reconstructing 3D structure from sparse views, but they struggle with dynamic scenes, fine semantic detail, and scalability to large outdoor environments [56]. Here we apply 3D Gaussian splatting and semantic segmentation of the environment to build a quantitative map of the natural world. This involved integrating a diverse set of algorithms for splatting, meshing, semantic segmentation, and alignment of multi-animal tracks in 3D space (Figure 3, Figure 4). Creating high-fidelity 3D meshes from 3D Gaussian splats was a particularly challenging step. Our codebase is designed modularly, to allow flexible swap-outs as new tracking or segmentation algorithms are developed.

In addition to computer-vision approaches to process multi-agent behavior in the wild, we developed GNN methods for interpreting collective behavior. The GNN methods were tested in simulation. Given the high sensitivity to initial conditions of the Boid dynamics, we should not expect exact fits, but the GNNs recovered some key features of ground-truth simulations, such as which simulated agents pay more attention to other agents, or to food. Taken together, the pipeline presented here can begin to address questions of how environment affects collective movement.

3.2 Limitations and future directions

One major limitation of this work is that it is a preliminary proof-of-concept of a data acquisition and analysis pipeline. As such, substantially more data and experiments are needed in order to support major scientific claims about what strategies different species of animals use in different environments, and the behavioral affordances of different environments. Larger scale experiments, including longer recording sessions, as well as recordings covering larger spatial scales, would be very beneficial for constraining models.

On the computer vision side, there are also several limitations that could be improved in future work. Despite the relative success of the tracking algorithms investigated, tracking quality still suffers from occasional dropouts due to agent-agent or agent-environment occlusions. Future work can integrate tools that are more robust to occlusion, like Omnimotion [59]. The current 3D environment models we analyze are static, but exciting recent advances allow '4D' dynamic Gaussian splats, which could capture a dynamically changing environment [61, 20]. This would likely require modified data acquisition, with more cameras. With the current camera setup, it was possible to estimate 3D position when animals were walking on flat areas of the environment, but near-impossible to estimate positions when animals flew or interacted with more complex geometric features such as tree branches and hanging feeders. Future work would benefit from integrating across many cameras and microphones. An array of microphones could enable audio sound source location [37, 28].

Our tests on simulated data identified some important limitations to consider in modeling multi-agent systems. Boid agent trajectories are highly sensitive to slight perturbations in initial conditions. Since real data will invariably contain some noise, predicting precise trajectories over a long timescale may prove impossible. However, GNN models can be trained to fit aspects of short-timescale data, and potentially provide interpretable/explainable results [3], providing insight about what strategies underlie behavior. This manuscript provides some promising results distinguishing simulated datasets governed by different rules, but further work on simulated data is needed to validate these approaches across a broader range of possible model classes, and capture real-world complexity.

Reasoning about multi-agent cognition is critical for social neuroscience, and has broad societal implications. The framework laid out here for integrating multi-agent dynamics with semantic scene understanding could benefit many domains, including human crowd management, traffic flow, drone coordination in disaster response, and creation of immersive ecological simulations for conservation and education.

4 Acknowledgments

ELM received support through the Simons Society of Fellows, as well as the NIH NINDS (award numbers 1K99NS131256-01 and 4R00NS131256-03). We are grateful for funding support to Basis Research Institute from private donations. We thank Matt Levine and Kim Stachenfeld for helpful discussions. Much appreciation to Toolik Field Station personnel, especially naturalist Seth Beaudreault, and the Teachers at Toolik Arctic LTER Education program.

References

[1] Kelsey R Allen et al. *Graph network simulators can learn discontinuous, rigid contact dynamics*. 2023. URL: `https://proceedings.mlr.press/v205/allen23a.html`.

[2] Cédric Allier et al. *Decomposing Heterogeneous Dynamical Systems with Graph Neural Networks*. July 2024. DOI: `10.48550/arXiv.2407.19160`. arXiv: `2407.19160 [cs]`. (Visited on 09/02/2025).

[3] Kenza Amara et al. *GraphFramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks*. May 2024. DOI: `10.48550/arXiv.2206.09677`. arXiv: `2206.09677 [cs]`. (Visited on 08/29/2025).

[4] ARC. *Thermal Bird Dataset*. `https://universe.roboflow.com/arc-ejbpg/thermal-bird`. Open Source Dataset. visited on 2025-08-13. June 2023. URL: `https://universe.roboflow.com/arc-ejbpg/thermal-bird`.

[5] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[6] Shaked Brody, Uri Alon, and Eran Yahav. *How Attentive are Graph Attention Networks?* 2022. arXiv: `2105.14491 [cs.LG]`. URL: `https://arxiv.org/abs/2105.14491`.

[7] Christopher F Brown et al. "AlphaEarth Foundations: An embedding field model for accurate and efficient global mapping from sparse label data". In: *arXiv preprint arXiv:2507.22291* (2025).

[8] Siji Chen et al. "Learning Decentralized Flocking Controllers with Spatio-Temporal Graph Neural Network". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. May 2024, pp. 2596–2602. DOI: `10.1109/ICRA57147.2024.10610627`. (Visited on 09/02/2025).

[9] Seung Whan Chung and Jonathan B. Freund. "An Optimization Method for Chaotic Turbulent Flow". In: *Journal of Computational Physics* 457 (May 2022), p. 111077. ISSN: 0021-9991. DOI: `10.1016/j.jcp.2022.111077`. (Visited on 09/04/2025).

[10] Xiaoyi Dong et al. *MaskCLIP: Masked Self-Distillation Advances Contrastive Language-Image Pretraining*. 2023. arXiv: `2208.12262 [cs.CV]`. URL: `https://arxiv.org/abs/2208.12262`.

[11] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[12] Matthias Fey et al. "PyG 2.0: Scalable Learning on Real World Graphs". In: *Temporal Graph Learning Workshop @ KDD*. 2025.

[13] John Harvey, Kathryn Merrick, and Hussein A. Abbass. "Application of Chaos Measures to a Simplified Boids Flocking Model". In: *Swarm Intelligence* 9.1 (Mar. 2015), pp. 23–41. ISSN: 1935-3812, 1935-3820. DOI: `10.1007/s11721-015-0103-0`. (Visited on 09/04/2025).

[14] Benjamin Hoffman and Grant Van Horn. *Behind the Scenes of Sound ID in Merlin*. `https://www.macaulaylibrary.org/2021/06/22/behind-the-scenes-of-sound-id-in-merlin/`. Macaulay Library, Cornell Lab of Ornithology. 2021.

[15] Glenn Jocher and Jing Qiu. *Ultralytics YOLO11*. Version 11.0.0. 2024. URL: `https://github.com/ultralytics/ultralytics`.

[16] Stefan Kahl et al. "BirdNET: A deep learning solution for avian diversity monitoring". In: *Ecological Informatics* 61 (2021), p. 101236.

[17] Bernhard Kerbl et al. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. 2023. arXiv: `2308.04079 [cs.GR]`. URL: `https://arxiv.org/abs/2308.04079`.

[18] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]`. URL: `https://arxiv.org/abs/1412.6980`.

[19] Thomas Kipf et al. "Neural Relational Inference for Interacting Systems". In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, July 2018, pp. 2688–2697. (Visited on 08/26/2025).

[20] Zhan Li et al. "Spacetime gaussian feature splatting for real-time dynamic view synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 8508–8520.

[21] Alexander Mathis et al. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". In: *Nature neuroscience* 21.9 (2018), pp. 1281–1289.

[22] MeshLib. *Geometry Processing Library*. Version 2.3.1. 2025. URL: `https://meshlib.io`.

[23] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[24] Thierry Mora and William Bialek. "Are biological systems poised at criticality?" In: *Journal of Statistical Physics* 144.2 (2011), pp. 268–302.

[25] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. en. arXiv:2304.07193 [cs]. Feb. 2024. DOI: `10.48550/arXiv.2304.07193`. URL: `https://arxiv.org/abs/2304.07193` (visited on 08/21/2025).

[26] Elise Özalp, Georgios Margazoglou, and Luca Magri. "Physics-Informed Long Short-Term Memory for Forecasting and Reconstruction of Chaos". In: vol. 10476. 2023, pp. 382–389. DOI: `10.1007/978-3-031-36027-5_29`. arXiv: `2302.10779 [cs]`. (Visited on 09/04/2025).

[27] Talmo D. Pereira et al. "SLEAP: A deep learning system for multi-animal pose tracking". In: *Nature Methods* (2022). DOI: `10.1038/s41592-022-01426-1`.

[28] Ralph Peterson et al. "Vocal Call Locator Benchmark (VCL) for localizing rodent vocalizations from multi-channel audio". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 106370–106382.

[29] Ralph Emilio Peterson et al. "Computational Urban Ecology of New York City Rats". In: *bioRxiv* (2025), pp. 2025–07.

[30] Jan von Pichowski and Sebastian von Mammen. "Engineering Surrogate Models for Boid Systems". In: *The 2023 Conference on Artificial Life*. ALIFE 2023. MIT Press, 2023. DOI: `10.1162/isal_a_00636`. URL: `http://dx.doi.org/10.1162/isal_a_00636`.

[31] Jason A. Platt et al. *Constraining Chaos: Enforcing Dynamical Invariants in the Training of Recurrent Neural Networks*. Apr. 2023. DOI: `10.48550/arXiv.2304.12865`. arXiv: `2304.12865 [cs]`. (Visited on 09/04/2025).

[32] Ri-Zhao Qiu et al. *Feature Splatting: Language-Driven Physics-Based Scene Synthesis and Editing*. 2024. arXiv: `2404.01223 [cs.CV]`. URL: `https://arxiv.org/abs/2404.01223`.

[33] Ri-Zhao Qiu et al. "Feature splatting: Language-driven physics-based scene synthesis and editing". In: *arXiv preprint arXiv:2404.01223* (2024).

[34] Vasista Ramachandruni et al. *Using Machine Learning and Neural Networks to Analyze and Predict Chaos in Multi-Pendulum and Chaotic Systems*. Apr. 2025. DOI: `10.48550/arXiv.2504.13453`. arXiv: `2504.13453 [cs]`. (Visited on 09/04/2025).

[35] Saikat Ray et al. "Hippocampal coding of identity, sex, hierarchy, and affiliation in a social group of wild fruit bats". In: *Science* 387.6733 (2025), eadk9385.

[36] Craig W. Reynolds. "Flocks, herds and schools: A distributed behavioral model". In: *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), pp. 25–34. ISSN: 0097-8930. DOI: `10.1145/37402.37406`. URL: `https://doi.org/10.1145/37402.37406`.

[37] Tessa A Rhinehart et al. "Acoustic localization of terrestrial wildlife: Current practices and future opportunities". In: *Ecology and Evolution* 10.13 (2020), pp. 6794–6818.

[38] David Robinson et al. *NatureLM-audio: an Audio-Language Foundation Model for Bioacoustics*. 2025. arXiv: `2411.07186 [cs.SD]`. URL: `https://arxiv.org/abs/2411.07186`.

[39] David Robinson et al. "Naturelm-audio: an audio-language foundation model for bioacoustics". In: *arXiv preprint arXiv:2411.07186* (2024).

[40] Isaac Robinson, Peter Robicheaux, and Matvei Popov. *RF-DETR*. `https://github.com/roboflow/rf-detr`. SOTA Real-Time Object Detection Model. 2025.

[41] Roboflow. *Roboflow Python Package*. URL: `https://github.com/roboflow/roboflow-python`.

[42] Joao F. Rocha et al. *STAGED: A Multi-Agent Neural Network for Learning Cellular Interaction Dynamics*. July 2025. DOI: `10.48550/arXiv.2507.11660`. arXiv: `2507.11660 [cs]`. (Visited on 08/26/2025).

[43] Matteo Sangiorgio, Fabio Dercole, and Giorgio Guariso. "Forecasting of Noisy Chaotic Systems with Deep Neural Networks". In: *Chaos, Solitons & Fractals* 153 (Dec. 2021), p. 111570. ISSN: 0960-0779. DOI: `10.1016/j.chaos.2021.111570`. (Visited on 09/04/2025).

[44] Paul-Edouard Sarlin et al. "From Coarse to Fine: Robust Hierarchical Localization at Large Scale". In: *CVPR*. 2019.

[45] Paul-Edouard Sarlin et al. "SuperGlue: Learning Feature Matching with Graph Neural Networks". In: *CVPR*. 2020.

[46] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[47] Johannes Lutz Schönberger et al. "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *European Conference on Computer Vision (ECCV)*. 2016.

[48] Christof Schötz et al. *Machine Learning for Predicting Chaotic Systems*. Mar. 2025. DOI: 10.48550/arXiv.2407.20158. arXiv: 2407.20158 [cs]. (Visited on 09/04/2025).

[49] Ariana Strandburg-Peshkin et al. "Habitat and social factors shape individual decisions and emergent group structure during baboon collective movement". In: *eLife* 6 (2017), e19505. DOI: 10.7554/eLife.19505. URL: https://elifesciences.org/articles/19505.

[50] Ariana Strandburg-Peshkin et al. "Shared decision-making drives collective movement in wild baboons". In: *Science* 348.6241 (2015), pp. 1358–1361.

[51] Andrea Tacchetti et al. "Relational forward models for multi-agent learning". In: *arXiv preprint arXiv:1809.11044* (2018).

[52] Matthew Tancik et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*. SIGGRAPH '23. ACM, July 2023, pp. 1–12. DOI: 10.1145/3588432.3591516. URL: http://dx.doi.org/10.1145/3588432.3591516.

[53] Jordi Torrents, Tiago Costa, and Gonzalo G de Polavieja. "New idtracker.ai: rethinking multi-animal tracking as a representation learning problem to increase accuracy and reduce tracking times". In: (Aug. 2025). DOI: 10.7554/elife.107602.1. URL: http://dx.doi.org/10.7554/eLife.107602.1.

[54] Mark Towers et al. *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. 2024. arXiv: 2407.17032 [cs.LG]. URL: https://arxiv.org/abs/2407.17032.

[55] Matias Turkulainen et al. *DN-Splatter: Depth and Normal Priors for Gaussian Splatting and Meshing*. 2024. arXiv: 2403.17822 [cs.CV]. URL: https://arxiv.org/abs/2403.17822.

[56] Matias Turkulainen et al. "Dn-splatter: Depth and normal priors for gaussian splatting and meshing". In: *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2025, pp. 2421–2431.

[57] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[58] Tristan Walter and Iain D. Couzin. "TRex, a fast multi-animal tracking system with markerless identification, and 2D estimation of posture and visual fields". In: *eLife* 10 (2021). DOI: 10.7554/eLife.64000.

[59] Qianqian Wang et al. "Tracking everything everywhere all at once". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19795–19806.

[60] Zhou Wang et al. "Image quality assessment: From error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.

[61] Guanjun Wu et al. "4d gaussian splatting for real-time dynamic scene rendering". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 20310–20320.

[62] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2020.2978386. (Visited on 09/04/2025).

[63] Baowen Zhang et al. *RaDe-GS: Rasterizing Depth in Gaussian Splatting*. 2024. arXiv: 2406.01467 [cs.GR]. URL: https://arxiv.org/abs/2406.01467.

[64] Baowen Zhang et al. "Rade-gs: Rasterizing depth in gaussian splatting". In: *arXiv preprint arXiv:2406.01467* (2024).

[65] Richard Zhang et al. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. arXiv: 1801.03924 [cs.CV]. URL: https://arxiv.org/abs/1801.03924.

[66] Yifu Zhang et al. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. 2021. DOI: 10.48550/ARXIV.2110.06864. URL: https://arxiv.org/abs/2110.06864.

[67] Bocheng Zhao et al. "Graph-based multi-agent reinforcement learning for large-scale UAVs swarm system control". In: *Aerospace Science and Technology* 150 (2024), p. 109166.

[68] Jie Zhou et al. “Graph Neural Networks: A Review of Methods and Applications”. In: *AI Open* 1 (2020), pp. 57–81. ISSN: 26666510. DOI: `10.1016/j.aiopen.2021.01.001`. (Visited on 08/26/2025).

[69] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).

[70] Siyu Zhou et al. “Clone Swarms: Learning to Predict and Control Multi-Robot Systems by Imitation”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE, Nov. 2019, pp. 4092–4099. ISBN: 978-1-7281-4004-9. DOI: `10.1109/IROS40897.2019.8967824`. (Visited on 08/26/2025).

5 Supplementary Information

5.1 Performance of object-detection models

Table 1: Performance comparison of object detection models on validation and test sets.

Metric	YOLOv11 (Fine-tuned)	RF-DETR (Fine-tuned)	YOLOv11 (Non-fine-tuned)
Validation Set			
mAP@50	90.6%	70.0%	51.6%
Precision	97.7%	95.2%	68.0%
Recall	89.0%	68.4%	47.4%
Test Set			
mAP@50	90.9%	83.6%	55.5%
Precision	97.4%	87.3%	77.8%
Recall	90.9%	79.8%	53.3%

5.2 List of species observed

We counted at least 32 unique species of birds across our datasets (Snow Goose, Greater White-fronted Goose, Brant, Cackling Goose, Canada Goose, Mallard, Northern Pintail, American Wigeon, Greater Scaup, Lesser Scaup, Bufflehead, Wild Turkey, Sandhill Crane, Mourning Dove, Red-bellied Woodpecker, Downy Woodpecker, Hairy Woodpecker, Blue Jay, Black-capped Chickadee, Tufted Titmouse, White-breasted Nuthatch, American Robin, House Sparrow, House Finch, American Goldfinch, Red-winged Blackbird, Northern Cardinal, Song Sparrow, American Tree Sparrow, Dark-eyed Junco, White-throated Sparrow, Lapland Longspur), as well as at least 6 species of mammals (Muskoxen, White-tailed Deer, Eastern Chipmunks, Eastern Gray Squirrels, Brown Rats, Northern Raccoons)

5.3 Compute resources

We use RunPod to instantiate compute instances with varying GPU, CPU and memory specifications depending on availability. Typical training times for the different tasks: 1 hour for tracking and detection runs in Figure 2; 1-3 hours for each Gaussian splat model used in Figures 3 and 4; 2 hours for GNN models used in Figure 6 (for the latter, see details in "Compute resources and batching" section in Section 6.4). The boid simulator generates about 6 frames per second on a Macbook Pro with an Apple M4 Max chip with 16 cores and 64 GB of Memory. The 3000 frames to create each of the two panels in Figure 5 took 8 minutes 12 seconds to generate.

5.4 External libraries

Our code depends on various open source tools which are automatically installed by the setup scripts. Their respective versions and licenses appear in the accompanying repositories. Throughout the paper we cite the major tools used:

- Detection and tracking: YOLO [15] (GPL-3.0), RF-DETR [40] (Apache 2.0), ByteTrack [66] (MIT), OpenCV [5] (Apache 2.0)
- Splatting and meshing: nerfstudio [52] (Apache 2.0), COLMAP [47, 46] (new BSD), Hierarchical Localization [44, 45] (Apache 2.0), MaskCLIP [10], dinov2 [25] (Apache 2.0), open3d [69] (MIT)
- Simulations: Gymnasium [54] (MIT)
- GNNs: pytorch-geometric [11, 12] (MIT)

6 Methods

6.1 Multi-animal Tracking

Raw data consisted of paired thermal and RGB videos recorded simultaneously in the field. For each session, thermal camera files (in `.csq` format) were converted to `.mp4` using a custom pipeline, available in the collab-environment repository, applying a perceptually uniform colormap to enhance contrast and facilitate downstream detection. RGB videos were cropped and resized to match the thermal frame dimensions. Manual alignment was performed to ensure spatial and temporal correspondence between the two modalities.

Spatial alignment involved a multi-step process. First, each individual RGB video was rotated to match the horizon line of the thermal video, compensating for differences in camera orientation. Next, a cropping rectangle was selected to approximate the thermal camera's field of view, using static environmental keypoints (e.g., trees, feeders, horizon features) as visual anchors. Homography or affine transformations were then estimated by manually selecting at least four corresponding points across both video frames, distributed across all quadrants of the image. These transformations were applied to warp the thermal frames into the RGB coordinate space, minimizing spatial misalignment. The homography matrix $H \in \mathbb{R}^{3\times3}$ was computed through OpenCV [5] such that for each pair of corresponding points $\mathbf{x}$ and $\mathbf{x}'$, the relationship $\mathbf{x}' \sim H\mathbf{x}$ holds, where $\sim$ denotes equality up to scale in homogeneous coordinates. This transformation preserves projective geometry and enables accurate warping between image planes. Note that homography assumes planar scenes, which is not guaranteed here.

Temporal alignment was conducted by overlaying the spatially aligned videos and manually inspecting frame-by-frame correspondence. Frame offsets were adjusted to synchronize motion and events across modalities, ensuring that each RGB frame corresponded to the correct thermal frame. While automated cross-correlation methods were considered, manual inspection was necessary to achieve precise alignment due to variability in frame rates and environmental motion.

All preprocessing steps use Python OpenCV [5], with user input facilitated through a graphical interface. Outputs were organized into session-specific directories to ensure reproducibility and efficient downstream processing.

Two object detection models were trained and evaluated in this study: YOLOv11 (Fast) [15] and RF-DETR (Nano) [40]. The YOLOv11 model was trained using a composite dataset comprising field-collected video frames and publicly available annotated thermal images. Field data were extracted from RGB and thermal videos collected during fieldwork sessions, preprocessed and aligned as described above. Individual frames were sampled from these videos and annotated manually to capture instances of animal presence across varied environmental conditions.

To supplement the field data and enhance model generalization, we incorporated an open-source instance segmentation dataset [4], which contains annotated thermal images of birds. This dataset was accessed via Roboflow Universe and integrated into the training pipeline after harmonizing class labels and preprocessing formats.

The combined dataset was uploaded to Roboflow and exported in YOLOv7 PyTorch format, which is compatible with all post-v7 Ultralytics YOLO models. It included 2,237 images, split into training (78%), validation (12%), and test (9%) sets. Preprocessing steps applied during export included auto-orientation, resizing to 480×480 pixels, contrast stretching, and class remapping. Data augmentation was configured to generate two outputs per image, with randomized hue shifts between $-23°$ and $+23°$, and saturation adjustments between -22% and $+22\%$.

Training was conducted on the YOLOv11 (Fast) architecture, selected for its balance of speed and accuracy. The model was trained on Roboflow's cloud infrastructure, and weights were exported as a `.pt` file for local inference, available with the codebase for this paper. Performance was evaluated using industry-standard metrics on the held-out test set, including mean Average Precision at IoU threshold 0.50 (mAP@50), precision, and recall.

We compared this model against a transformer-based alternative. We trained an RF-DETR (Nano) model on a separate dataset version, also available with this paper. Preprocessing steps included auto-orientation, resizing to 640×640 pixels, contrast stretching, grayscale conversion, and class remapping. Data augmentation was applied with two outputs per image, incorporating brightness adjustments between -26% and $+26\%$.

To validate the use of a fine-tuned model, we compared the performance to a default pre-trained YOLOv11 model, which showed a worse performance on both validation and test sets, highlighting the need for manual annotation of specific behavioral data.

We tracked the bounding boxes via the ByteTrack algorithm [66] through the Ultralytics library interface, using default parameters, and removed tracks that were shorter than 50 frames in length. We also validated the tracks using `idtracker.ai` tool [53], finding overlap between the identified masks and tracks of the animals during manual observation and track correction.

6.2 Environment modeling

6.2.1 Training 3D Gaussian splat models

At each fieldwork site, we recorded an additional RGB video that explored the spatial extent of the environment. We used this video to create a 3-D model of the each environment, parameterized with 3-D Gaussian Splats (3DGS) trained using nerfstudio [52]. Below, we outline the specifics of our 3DGS model training process.

We used COLMAP [47, 46] with Hierarchical Localization [44, 45] to estimate the spatial structure of the scene. Provided a set of overlapping images of an object/scene, COLMAP uses Structure-from-Motion (SfM) to extract camera extrinsics (world positions) and intrinsics (field-of-view, focal point) and build a 3-D reconstruction. Hierarchical Localization extends this functionality, improving the alignment of cameras by matching features extracted by a convolutional neural network (CNN) across the original images. For each aligned camera (video frame), this process yields a set of extrinsics and intrinsics that can be used for both training a 3DGS as well as aligning out-of-sample cameras to the scene.

We then trained a 3DGS model to reconstruct the scene from these camera poses and their associated images. 3DGS models explicitly represent the scene as sets of 3D gaussians with learned positions (mean), covariances (rotations/scales), colors (spherical harmonics), and opacities. During training, the model learns to reconstruct the original image via rasterization of the gaussians viewed from a given camera pose. Within our implementation, the core 3DGS model loss had three terms: (1) photometric, (2) structure similarity, and (3) total variation loss. Photometric loss is a pixel-wise $\mathcal{L}_2$ that encourages accurate color and pixel-wise reconstruction. Structural similarity (SSIM) encourages perceptual quality by comparing local image features (i.e., contrast, luminance, texture) between the two images. Total variation loss encourages spatial smoothness of the image and removes high-frequency noise. These losses are then weighted ($\lambda = 0.2$) and combined into a single function:

$$\mathcal{L}_{GS} = (1-\lambda)\,\|\hat{I} - I\|_1 + \lambda\left(1 - \text{SSIM}(\hat{I}, I)\right) + \text{TV}(\hat{I}) \tag{1}$$

In addition to the traditional 3DGS reconstruction loss ($\mathcal{L}_{GS}$), we included the depth-normal consistency loss ($\mathcal{L}_{DN}$) to regularize the gaussians to align with the surface of the scene ([63]). Specifically, we directly rasterize the depth and normals (the unit vector perpendicular to the gaussian) during the rendering process. We then measure the consistency (cosine similarity) between the rendered normal (n) and the normal computed from the depth map ($\tilde{n}$) with a per-pixel weighting (ω):

$$\mathcal{L}_{DN} = \sum_i \omega_i \left(1 - n^\top \tilde{n}_i\right), \tag{2}$$

Lastly, we integrated semantic information with the 3DGS representation of the scene [33]. For each image, we extracted semantic embeddings using MaskCLIP [10] and DINOv2 [25]. During training, the 3DGS model aims to reconstruct these semantic representations from low-dimension representations assigned to the gaussians. These gaussian semantic features are then projected to the original dimensionality via a two-layer MLP. These features are optimized via a feature reconstruction loss ($\mathcal{L}_{FS}$) with MaskCLIP features ($\mathcal{F}_C$) as the main objective and DINOv2 ($\mathcal{F}_D$) features for regularization:

$$\mathcal{L}_{FS} = \|\mathcal{F}_C - \tilde{\mathcal{F}}_C\|_2 + \lambda\|\mathcal{F}_D - \tilde{\mathcal{F}}_D\|_2 \tag{3}$$

The composed loss function for our 3DGS models used weights $\lambda_{DN} = 0.05$ and $\lambda_{FS} = 10^{-3}$ and is written as follows:

$$\mathcal{L} = \mathcal{L}_{GS} + \lambda_{DN}\mathcal{L}_{DN} + \lambda_{FS}\mathcal{L}_{FS} \tag{4}$$

All 3DGS models were trained for a total of $30,000$ steps using the ADAM optimizer [18]. Importantly, we trained these models for $15,000$ steps using solely the $\mathcal{L}_{GS}$ and $\mathcal{L}_{DN}$ losses. For the remaining $15,000$ steps, we added in the $\mathcal{L}_{FS}$ loss term until all steps were completed.

6.2.2 Creating mesh surfaces from Gaussian splats

Traditionally, meshes generated from Gaussian splats are relatively low fidelity. Our added depth-normal consistency loss aids the mesh generation process by encouraging the gaussians to align with the surface of the scene. We then performed surface reconstruction using Open3D's ScalableTSDFVolume (voxel size $= 0.005$, SDF truncation $= 0.03$, depth truncation $= 1.0$). In brief, this approach integrates depth and aligns camera frames, fuses associated information into a volume, and uses Marching Cubes to reconstruct a mesh surface. Lastly, we used MeshLib [22] to clean and repair the mesh, removing disconnected mesh clusters and interpolating over missing regions (max size $= 3.0$).

We mapped features associated with the gaussians to the mesh surface via a KDTree [57]. For each vertex on the mesh, we identified associated points within the original Gaussian splat point cloud using K-Nearest Neighbors ($k = 5$). We then averaged the feature values from these nearest points and assigned the resulting value to the vertex. We then repeated this process separately for each feature: colors, normals, and semantic representations.

6.2.3 Querying and segmenting the mesh surface

Provided with the semantic features associated with the mesh environment, we next leveraged the original semantic space (here, MaskCLIP; [10]) to understand the environment through natural language. To decompose the environment into semantic features of interest, we used the language-query method introduced within Feature Splatting [32].

We first selected a set of "positive" language queries that we aimed to identify (e.g., "tree", "feeder"). We then selected a set of "negative" queries that may contribute noise to identifying our queries of interest (e.g., "ground", "leaves"). For each query (positive or negative), we extracted a text embedding using the original semantic encoder. Next, we calculated the cosine similarity between each text embedding and the mesh-associated semantic features. Lastly, we applied a temperatured softmax ($\tau = 0.05$) to obtain a probability distribution of similarity values. For each "positive" query, this process yielded a continuous probability map of the mesh that indicates the likelihood of that query at a given place within the environment.

To discretize the environment into semantic categories, we took a thresholding and clustering approach to semantic segmentation. For a given query probability map, we thresholded the probability values (threshold $= 0.95$) to determine areas of the environment with high likelihood of that query. We then used DBSCAN to identify spatially contiguous areas of high probability, and assigned each cluster the label of the query. We then aggregated these labels across query maps to obtain a single semantic segmentation map.

6.2.4 Re-projecting tracks onto the mesh surface

We developed a multi-step approach to project 2-D tracks from the RGB videos onto the 3-D mesh surface. Specifically, we used the Hierarchical Localization toolbox [44] to localize the camera within the 3-D environment. For the new camera image (RGB video with tracks), we used a deep graph neural network [45] to extract and match image features to the cameras of the trained COLMAP model. We then calculated the pose of the RGB camera using RANSAC consensus (max error $= 50$) over the matched keypoints.

Given the aligned camera pose (i.e., extrinsics, intrinsics), we validated the quality of alignment by comparing the camera's perspective of the mesh (predicted image) against the ground-truth RGB image. We masked non-mesh portions of both images and calculated the structural (SSIM; [60])

and perceptual similarity (LPIPS; [65]) of the unmasked regions. These metrics provide an initial, quantitative estimate of camera alignment quality beyond pure visual comparison.

Within this work, we constrained the re-projection of tracks to moments where 2-D tracks intersected with the mesh surface. To this end, we calculated the distance of the mesh surface from the camera pose. This provided a bounded distance value for any point on the mesh that could be seen from the camera perspective. Next, for each track (uniquely identified animal), we calculated the image coordinates of the bottom-center of the bounding box (u, v). We used these (u, v) coordinates to extract a depth value for each tracking frame. We then used this depth heuristic to re-project the 2-D image coordinates to 3-D world coordinates, and repeated this process for each set of tracks.

6.3 Simulation models

Our simulations use a variant of the Boids model originally defined by Reynolds [36]. In the basic boids model, the movements of the agents are influenced by three forces: **separation** pushes agents within a specified distance away from each other to prevent collisions; **alignment** drives the agents to match the velocities of their immediate neighbors; and **cohesion** pulls agents toward the center of their neighbors. Our model adds environmental features to this framework, representing food and shelter, for example, that attract the agents. Each of these forces has an associated weight and the agent's acceleration is the weighted sum of these forces limited by a maximum total force allowed. Mathematically, the system can be described as follows. Let agent $n = 1, \ldots, N$ have position $x^{(n)}(t)$ and (instantaneous) velocity $v^{(n)}(t) = \frac{d}{dt}x^{(n)}(t)$ at time t. The overall state is then $(x(t), v(t)) = \left(x^{(1)}(t), \ldots, x^{(N)}(t), v^{(1)}(t), \ldots, v^{(N)}(t)\right)$ which evolves according to the dynamics

$$
\begin{aligned}
\frac{d}{dt}x^{(n)}(t) &= v^{(n)}(t) \\
\frac{d}{dt}v^{(n)}(t) &= \begin{cases} F^{(n)}_{\text{total}} & \text{if } \|F^{(n)}_{\text{total}}\| \leq m_F \\ \frac{F^{(n)}_{\text{total}}}{\|F^{(n)}_{\text{total}}\|} m_F & \text{otherwise} \end{cases}
\end{aligned}
$$

where

$$
\begin{aligned}
F^{(n)}_{total} &= F^{(n)}_{\text{align}}(x, v) + F^{(n)}_{\text{sep}}(x) + F^{(n)}_{\text{coh}}(x) + F^{(n)}_{\text{env}}(x, v, \mathcal{E}) \\
F^{(n)}_{\text{sep}}(x) &= w_s \left(\frac{1}{|S(n)|} \sum_{m \in S(n)} \frac{x^{(n)} - x^{(m)}}{\|x^{(m)} - x^{(n)}\|^2} \right) \\
F^{(n)}_{\text{align}}(x, v) &= w_a \left[\left(\frac{1}{|A(n)|} \sum_{m \in A(n)} v^{(m)} \right) - v^{(n)} \right] \\
F^{(n)}_{\text{coh}}(x) &= w_c \left[\left(\frac{1}{|C(n)|} \sum_{m \in C(n)} x^{(m)} \right) - x^{(n)} \right] \\
F^{(n)}_{\text{env}}(x, v, \mathcal{E}) &= \mathbf{w}_{\mathcal{E}} \cdot \mathbf{f}_{\mathcal{E}} \\
S(n) &= \{m : \|x^{(m)} - x^{(n)}\| < d_s\} \\
A(n) &= \{m : \|x^{(m)} - x^{(n)}\| < d_a\} \\
C(n) &= \{m : \|x^{(m)} - x^{(n)}\| < d_c\} \\
d_s &\in \mathbf{R}^+ \text{ is the minimum desired separation distance between boids} \\
d_a, d_c &\in \mathbf{R}^+ \text{ determine the size of the alignment and cohesion neighborhoods, respectively} \\
m_F &\in \mathbf{R}^+ \text{ is the maximum allowable acceleration for each boid in a single time step} \\
w_s, w_a, w_c &\in \mathbf{R} \text{ are weights determining the impact of each type of force}
\end{aligned}
$$

$\mathbf{f}_{\mathcal{E}} \in \mathbf{R}^k$ is a feature vector of length k describing the environment

$\mathbf{w}_{\mathcal{E}} \in \mathbf{R}^k$ is a weight vector determining the impact of the environmental features

Each agent in our simulation was 10 units in diameter. The separation and neighborhood distances were 20 and 80 units, respectively. The separation, alignment, and cohesion weights were 15.0, 1.0, and 0.5, while the target attraction weight was 0.002. At each time step, the independent agents chose their acceleration using a normal distribution with mean 0.1 and variance 0.01 along each axis. To prevent agents from crashing into the ground and other obstacles in the scene, when the agents are within 7.5 units of any part of the 3D scene, they ignore the boid rules and instead move with an acceleration in the opposite direction of the closest point on the scene in the horizontal plane but upward along the vertical axis. There is generally more freedom to move upward in the scene, so the upward movement prevents the agents from getting stuck in nooks and crannies along the ground. This acceleration is limited by the maximum force as it is when following the boid rules. The speed of each agent is also limited to be in the range $[0.1, 1.0]$. This prevents the agents from either stagnating or from moving furiously from one end of the world to the other and back. The world itself sits inside a cube that is 1500 units long on each axis. If an agent reaches the edge of the cube, its velocity is reversed. The initial velocity of each agent was chosen normally with mean 0.0 and variance 0.5 along each axis.

6.4 Graph neural network (GNN)

Input and output Given F frames of N animal positions or simulated agent positions $\{x_t^{(n)}\}_{t=1}^{F}$, $n = 1, \ldots, N$, we take finite differences to produce approximate instantaneous velocities $\{v_t^{(n)}\}_{t=1}^{F-1}$ as well as approximate instantaneous accelerations $\{a_t^{(n)}\}_{t=1}^{F-2}$. With frame-by-frame positions and velocities, we encode node features (see below GNN Encoder). In addition, for each frame, we construct a proximity graph based on each frame's node positions (see GNN Encoder below). For each frame, given the node features and the graph, we ask the GNN to predict all nodes' acceleration on the current frame.

We tested the GNN on data simulated in a simple 2D environment with a single fixed food source. Both Boid agents and independent agents were simulated. The environment is a $[0, 1] \times [0, 1]$ box, where agents bounce against the boundaries. All agents have some attraction to the food source, within a visual range of 0.4. Two types of Boid agents were simulated, with 3x different strengths of attraction to the food source.

GNN Encoder The Encoder constructs a graph $G = (V, E)$ from the input $\{x_t^{(n)}\}_{n=1}^{N}$ for each frame t and its two preceding frames. Each node $V^n \in V$ has features $\left(\{x_s^{(n)}, v_s^{(n)}, 1 - x_s^{(n)}\}_{s=t-2}^{t}, h^{(n)}\right)$ where $x^{(n)}$ and $1 - x^{(n)}$ encodes displacement to boundary at 0 and 1 and $h^{(n)}$ is a one-hot vector denoting species. Here $x^{(n)}, v^{(n)}$ are 2d or 3d vectors. Each node V^i receives connections from nearby nodes V^j if $\left\|x_t^{(i)} - x_t^{(j)}\right\|_2 \leq 0.1$. The incoming edge weights are normalized such that the total incoming weights for each node sum to 1. We add a food source as a node with a position, zero input velocity, and a unique species input.

GNN Architecture There are two layers in the network – a graph attention layer followed by a graph convolutional layer. This two-layer shallow architecture is common in previous applications to node location prediction problems [1, 30], with the attention mechanism added to allow for interpretability. We use the `pytorch-geometric` package [11, 12] with `GATv2Conv` [6] for the first layer and `GCNConv` for the second layer.

First, all node and edges features are passed through the graph attention layer, producing latent vectors of size 128 for each node in the graph and attention weights for each edge in the graph. The size of input for each node is 19 for the 2D dataset without food and 20 for the 2D dataset with food. Regardless of the input size, the size of output for each node is 128. We obtain the attention adjacency weights from this layer and use them as the edge features for the next convolutional layer. The activation is passed through a ReLU before feeding, along with the newly obtained edge feature, into the second main layer – the graph convolutional layer. The output size is again 128. We then pass the activation through a ReLU. Finally, we feed the activation to a linear layer with output predictions of size 2 in the case of 2D data or 3 in the case of 3D data.

Training We train the GNN by minimizing the frame-by-frame mean squared error between the GNN output and the correct acceleration across selected frames. Due to the nature of the Boid-rule interaction, once a flock is formed, the acceleration would be zero for a large number of frames until some of the agents interact with the boundaries. To prevent the network from over-training on the zero acceleration data, we train the network on selected frames. Frames are selected by first identifying frames where agents have incongruent velocities. We calculate the mean velocity vector for all frames and calculate the mean and standard deviation (SD) of the frame-to-frame changes of the norm. We select frames where the changes of the velocity vector norm is greater than mean + 1 SD. We pad each selected frame with 2 frames back in time and 3 frames further in time. All snippets are stitched and combined when continuous longer snippets can be formed and duplicated frames are removed. For batching operation (see section below "Compute resources and batching"), each file/sample still has its own threshold. For each frame, as long as it crosses the threshold for one file, the frame along with the padded snippet will be added to the training set for all files/samples in the batch. Finally, during both training and testing, acceleration is clipped to be max of 1.

Attention Using the attention mechanism in the Graph Attention Network, we obtain matrices whose (i, j)-th entry speaks about the importance of node j to node i.

Generating rollout trajectories. To generate rollout trajectories, the model is applied iteratively, initialized with the first 5 frames of node position and velocity. For each frame, per-node acceleration prediction is integrated with Euler's method to compute the next-frame position.

Rollout baseline Due to the chaotic nature of multi-agent systems with the boid-rule interaction, we would like to know to what extent a little perturbation would cause agents trajectories to diverge from ground truth. We simulate trajectories with initial velocity perturbed by adding Gaussian noise of size 0.005 and run forward the simulation using the boid update rule for the dataset without food or the boid with food update rule for the dataset with food. There is no additional noise when we run forward the model. The system diverges from the ground truth very early on, see figure 7.

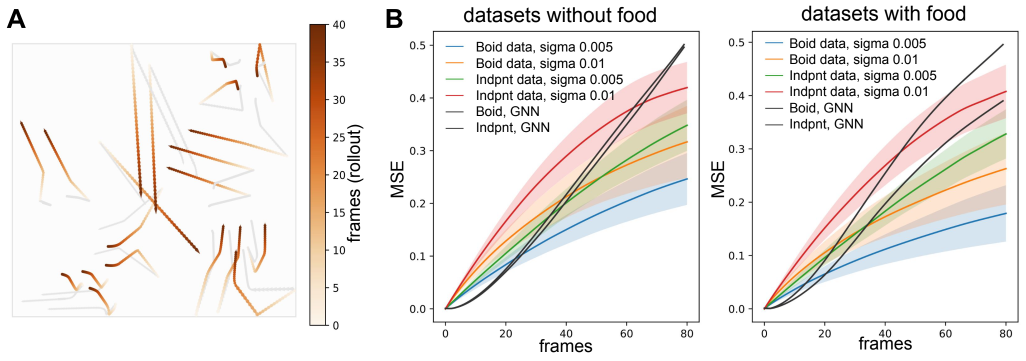


Figure 7: **Simulation of Boid dataset with initial velocities perturbed with small amount of noise.** **A**, original boid simulation shown in gray; trajectories with perturbed initial velocity shown in red. We normalized the view into a 1 by 1 box. **B**, colored lines show mean squared error between the original trajectories and the perturbed trajectories averaged across 300 test samples as a function of number of frames. Shadings show standard deviation. Black lines show the mean squared error between the original trajectories and the rollout trajectories from GNN models averaged across 300 test samples. The left panel shows MSE curves from the dataset without food source, and the right panel shows MSE curves from the dataset with food source. The MSE metric is calculated based on view data normalized into a 1 by 1 box.

Compute resources and batching We generated 1000 samples/files for each dataset and train/test with a 70/30 split. We use batch size 50 on a GPU. We fit each data set with five random seeds and select the best seed according to the average mean squared error of position across the test files/samples.

6.5 Fitting GNN to real data

As a proof of concept, we applied our GNN training and prediction pipeline to a excerpt of tracked movements, extracted from a tracked video of birds (Creamer's Field Migratory Waterfowl Refuge, Fairbanks, Alaska), Figure 8(A,B). We wondered whether the GNN trained on the real tracks would produce different rollout behaviour compared to a model pre-trained on simulated boid movements. Following data cleaning and filtering, we obtained 203 "episodes" such that the bird identities remain constant throughout each episode with varying number of birds in each, Figure 8(C). We obtained two GNNs using this data as follows: a) a model pre-trained on boids, and b) a model trained on the real tracks. We have then compared the rollouts generated by the two models, with the initial conditions and velocities taken from the tracked data. As can be seen from Figure 8(D), the overall movement pattern predicted by the two models are vastly different, suggesting that indeed the behavioral strategies employed by the real birds are substantially different from the boid rules.

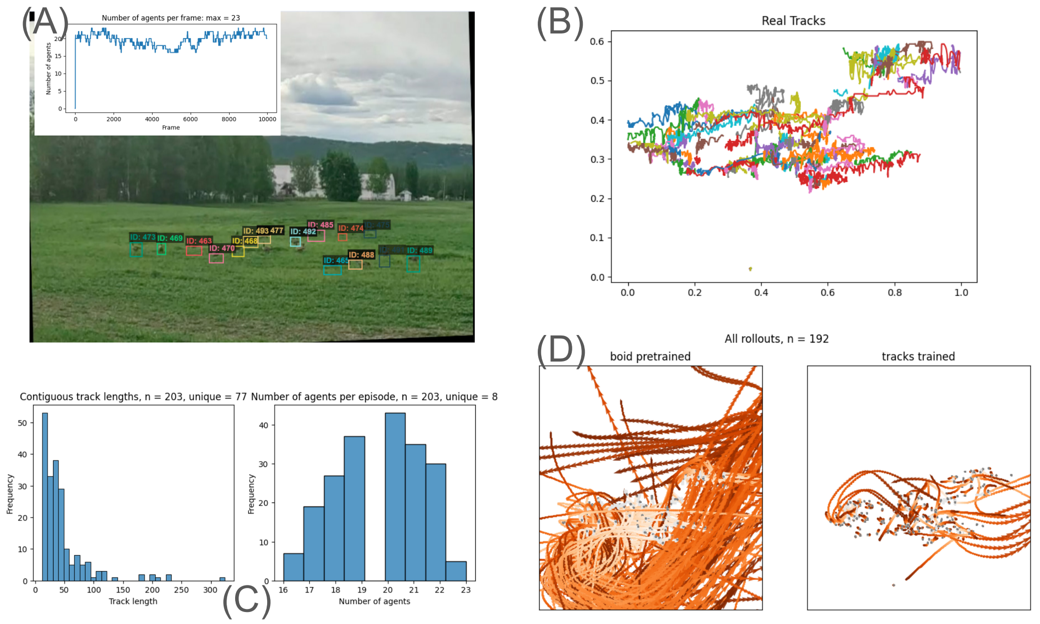


Figure 8: **GNN training and rollouts on real-world tracks.** (A) Snapshot of the tracked video. Number of birds in each frame varies between 15 and 20 (inset). (B) The time traces of the tracks in a normalized coordinate system, suitable as input for GNN training. (C) Dividing the data into episodes with constant bird identities, resulting in 203 episodes. (D) Comparison between the rollouts of a GNN pretrained on boids (left) vs. a GNN trained on the tracks themselves.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract are: 'Here, we introduce new behavioral data from groups of birds foraging in outdoor 3D environments, as well as an open-source framework for interpreting this data. The framework combines detection and tracking of multiple animals, semantic 3D environment reconstruction, multi-agent simulation, and graph neural network–based inference'. These claims are supported by the following results sections.

   (a) New behavioral data: Section 2.1
   (b) Open-source framework: Section 2.6. Note that the code is currently posted anonymously for review.
   (c) Detection and tracking of multiple animals: Section 2.1
   (d) Semantic 3D environment reconstruction: Section 2.2
   (e) Integration of tracking with 3D environment: Section 2.3
   (f) Multi-agent simulation: Section 2.4
   (g) GNN inference: 2.5

   The abstract and introduction also mention that we identify key challenges to inferring behavioral strategies in multi-agent systems. The challenges/limitations are described in the results (Section 2.5) and supplemental information (Section 6.4), as well as the discussion (Section 3).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 3.2 for discussion of limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide access to source code (anonymized for review purposes, to be open-sourced for camera-ready version). The accompanying documentation describes the concrete steps to follow in order to reproduce each figure. See Section 2.6.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case

of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Upon camera-ready submission, we will provide open source access to our code and data, with detailed instructions for reproducing the reported experimental results. During submission we provide anonymized access to the code and the data. See Section 2.6.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All model training details are located in Section 6 (Supplement).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: This is N/A for most of the paper which provides qualitative proof-of-concept results of technical advances. The exception is Figure 6, where statistical significance is explained in the figure legend and in Section 6.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe all the computational resources in Section 5.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, this is discussed in Section 3.2.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not pose risks of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Our code depends on various open source tools which are automatically installed by the setup scripts. Their respective versions and licenses appear in the accompanying repositories. Throughout the paper we cite the major tools used. A summary is provided in Section 5.4.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The work presented herein introduces an open-source framework under the Apache license. See code repository for more details.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper involves neither crowdsourcing experiments nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper involves neither crowdsourcing experiments nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not use LLMs in any important, original, or non-standard component of our research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.