Classification of Flying Objects using Multi-Camera 4D Gaussian Splatting

A research thesis by
Alessandro Verdiesen

TU Delft

Classification of Flying Objects using Multi-Camera 4D Gaussian Splatting

By

Alessandro Verdiesen

to obtain the degree of Master of Science at Delft University of Technology, to be defended publicly on 12th of June 2026.

Student number:	5885396	
Project duration:	A 9-month study	
Thesis committee:	Prof. Dr. Harm Peter Hofstee	TU Delft, Supervisor
	Dr. Michael Weinmann	TU Delft, Committee Member
	Dr. Ir. Zaid Al-Ars	External Advisor
	Wim Bos	Lumiad BV, External company supervisor

An electronic version of this thesis is available at repository.tudelft.nl.

Preface

I wish to thank my supervisors, Peter and Zaid (in no particular order), for their guidance, their steady support, and their willingness to engage with every aspect of this project.

I owe particular thanks to Wim, my supervisor at Lumiad. We first spoke at a gathering about our shared interests, where I mentioned that I would soon begin my thesis. He remembered, and on an otherwise ordinary day he called to ask whether I was ready and proposed a project I could not resist: working with flying aerial objects, quadcopters in particular, which have been part of my life since high school, combined with computer vision and AI, two fields I had wanted to explore more deeply alongside my master's in computer engineering. That he thought to call at exactly the right moment is something I genuinely admire. I am grateful, too, to Louis for keeping the project on course as its manager, and for the full exploratory freedom I enjoyed at Lumiad. Thank you for your trust and for making it possible.

I would also like to thank Reinier and Benno, whose clear and technically precise questions throughout the process repeatedly sharpened my thinking.

To my university friends Nicolas, Utkarsh, and Tristan, with whom I learned the academic methodology from the very beginning: thank you for your motivating friendship.

As with any project, there were ups and downs, and I am fortunate to have family who helped me keep a clear and fresh eye on complex matters through many late-night calls and shared problem-solving. I am especially thankful to my brother Remi and my father Bert. Remi took the time to truly understand the project in its full depth and generously guided me in academic thinking and the structuring of ideas into a clear, coherent narrative. My father, in turn, thought through every challenge with me, patiently listening as I explained my work until the ideas became clear enough for me to move forward. Both gave generously of their time and support throughout this thesis. To my mother Rosiane and girlfriend Nazareth, thank you for the heartfelt support that extended far beyond the academic side, keeping me grounded, calm, and motivated throughout.

To all of you, thank you once again. This work would not have been possible without your support.

Alessandro Verdiesen
Delft, the Netherlands
June 4, 2026

Cover image rendered from third-party 3D models. "The Prison" by Veterock, "Crow" by Alexei Ostapenko, "Tree Animate" and "Tree Wind System" by Node_Art, and "Animated Oak Trees" by Jagobo were obtained from Sketchfab under the CC BY 4.0 license and modified for use on the cover. "Tarot Ironman 650" by Galih Sukma Adjie was obtained from the GrabCAD Library and is used under GrabCAD's Terms of Service.

Abstract

Monitoring the lower airspace for small drones and distinguishing them from birds, helicopters and airplanes, is a growing security need that radar, radio-frequency, and acoustic sensors meet only at considerable cost. This thesis asks whether a ground-based network of synchronized, overlapping RGB cameras can instead reconstruct and classify flying objects directly in 3D, recovering range through multi-view geometry rather than a long-range sensor. The central hypothesis is that the temporal evolution of a 3D Gaussian Splatting representation carries motion cues more discriminative than per-frame 2D or static 3D appearance.

Four contributions support this investigation, which, to our knowledge, is the first to classify flying objects using temporal 4D Gaussian features. `AeroSplat-4D` is a synthetic multi-camera dataset and NVIDIA Isaac Sim pipeline emitting synchronized RGB, instance masks, depth, 3D trajectories, and exact calibration across the four classes, with class-balanced, identity-disjoint splits. `DepthSplat-OC` adapts feed-forward Gaussian splatting to thin, distant targets against a texture-less sky via a mask-gated photometric loss. `MambaSplat-4D`, the main contribution, classifies the temporal Gaussian sequences by pairing a rotation-equivariant Vector-Neuron Transformer with a linear-time Mamba temporal encoder, enforcing SO(3) invariance architecturally rather than through augmentation.

In an augmentation-free ablation, aggregating a 24-frame clip rather than classifying a single frame raises accuracy from 59.1 % to 78.8 %, confirming that motion, not single-frame appearance, drives discrimination. Because SO(3) invariance is enforced architecturally, the full-attribute model attains the same 70.2 % four-class accuracy on clean and arbitrarily rotated data, about eight percentage points above a position-only baseline; it trails the strongest temporal baseline by roughly five points on clean data but is uniquely robust under rotation, with zero classification changes across 9600 rotated forward passes. `DepthSplat-OC` surpasses the closest-protocol baseline (24.65 versus 21.44 PSNR) despite roughly two orders of magnitude less training compute, and the compact 1.9 M-parameter classifier runs in under a millisecond per frame. On the out-of-distribution probe the pipeline does not yet surpass the 2D baselines, a gap that likely reflects their ImageNet-pretrained (∼1.2M-image) backbones rather than a limit of the 3D representation; real-camera transfer remains open, and and the core of the pipeline is released as open-source software at github.com/lumiad-bv/MambaSplat-4D.

This work thereby points toward multi-view 3D reconstruction and temporal reasoning as an effective alternative to the per-frame 2D detection that currently dominates aerial object classification.

Contents

List of Figures

List of Symbols and Abbreviations

This section provides definitions for the main symbols and abbreviations used throughout this thesis.

Symbols

c	Camera index
t	Time
$\mathbf{x}, (x, y, z)$	3D point coordinates
μ_i	Mean position of Gaussian splat i
Σ_i	Covariance matrix of Gaussian splat i
α_i	Opacity of Gaussian splat i
$\mathbf{s} = (s_x, s_y, s_z)$	Scales of a Gaussian splat
$\mathbf{q}$	Quaternion representing rotation
$\mathbf{c}^{\text{SH}}$	Spherical-harmonic color coefficients of a Gaussian
$\mathcal{G}$	Set of Gaussian primitives
$\mathbf{P}_c$	Camera projection matrix for camera c
$\mathbf{K}_c$	Camera intrinsic matrix for camera c
$\mathbf{R}_c$	Camera rotation matrix for camera c
$\mathbf{t}_c$	Camera translation vector for camera c
ω	Dominant frequency (Hz) of periodic motion
K	Number of tokens selected
$\Delta\mu/\Delta t$	Velocity (temporal derivative of position)
$\Delta s/\Delta t$	Temporal derivative of scales
Δrot	Temporal change in rotation
N	Number of cameras ($N \geq 3$); also number of Gaussians or input tokens
T	Number of temporal frames in a clip (sequence length)
Δt	Frame interval
C	Number of vector-neuron channels (distinct from camera index c)
D	Per-channel coordinate dimension ($D = 3 + d_{\text{inv}}$)
d_{inv}	Number of injected invariant scalars
$\mathcal{V}$	Monitored airspace volume
R_{cam}	Camera-rig radius
R_{vol}	Monitored-volume radius
$\sigma_{\min}, \sigma_{\max}$	Lower and upper bounds on Gaussian scale magnitude
λ	Photometric-loss weight
ε	VN-Transformer bias magnitude (10^{-6}); separately, label-smoothing weight (0.1)
$\mathbf{T}$	Learned bridge frame (distinct from camera translation $\mathbf{t}_c$)
d_{mamba}	Mamba hidden dimension
n_{layers}	Mamba depth (number of layers)
$\Delta_{\max}$	Worst-case classifier-logit drift under rotation

$m_{\min}$	Minimum predicted-class margin
$E(\cdot)$	Feature-mode family: $E(C)$, $E(C,O)$, $E(C,\mathrm{SH})$, $E(C,S,R)$, $E(O,C,S,R)$, $E(X)$ (all attributes)

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
4D	Four-dimensional (spatial + temporal)
3DGS	3D Gaussian Splatting
4DGS	4D Gaussian Splatting
3D-4DGS	Hybrid 3D–4D Gaussian Splatting
AdamW	Adam optimizer with decoupled weight decay
AP	Average Precision (%)
BA	Bundle Adjustment
BEV	Bird's-Eye-View
bf16	16-bit brain floating-point format
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
COLMAP	Structure-from-Motion software package
DBT	Detection Before Track
DINOv2	Self-supervised vision transformer (third version)
DoF	Degrees of Freedom
DPT	Dense Prediction Transformer
DR	Domain Randomization
ECE	Expected Calibration Error
EWA	Elliptical Weighted Average
F1	F1-score (harmonic mean of precision and recall; unitless)
FFT	Fast Fourier Transform
FoV	Field of View
fp32	32-bit floating-point format
FPS	Farthest Point Sampling (or Frames Per Second, context-dependent)
GELU	Gaussian Error Linear Unit
GFLOPs	Giga Floating-Point Operations (model compute cost)
GPU-hours	Training compute (number of GPUs times wall-clock hours)
GT	Ground Truth
HOTA	Higher Order Tracking Accuracy
IDF1	ID F1-score (tracking metric)
InfoNCE	Contrastive loss function
IoU	Intersection over Union (unitless)
LN	Layer Normalization
LPIPS	Learned Perceptual Image Patch Similarity (unitless, lower is better)
LRM	Large Reconstruction Model
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
macro-F1	Class-averaged F1-score (unitless)
mAP	mean Average Precision (%)
MLP	Multi-Layer Perceptron
MOT	Multi-Object Tracking
MSE	Mean Squared Error (unitless, training loss)
MVG	Multi-View Geometry
MVS	Multi-View Stereo
OOD	Out-of-Distribution
PnP	Perspective-n-Point (camera pose estimation)
PPM	Pixels Per Meter (pixels/m)
PSNR	Peak Signal-to-Noise Ratio (dB)
PTv3	Point Transformer v3 (backbone architecture)
RANSAC	Random Sample Consensus

re-ID	Re-identification
RGB	Red-Green-Blue (color channels)
RMSE	Root Mean Square Error
RPM	Revolutions Per Minute (rotor speed)
SCR	Signal-to-Clutter Ratio (dB)
SE(3)	Special Euclidean group (3D rotations and translations)
SfM	Structure-from-Motion
SGD	Stochastic Gradient Descent
SH	Spherical Harmonics
SNR	Signal-to-Noise Ratio (dB)
SO(3)	Special Orthogonal group (3D rotations)
SR	Super-Resolution
SSIM	Structural Similarity Index Measure (unitless)
SSM	State Space Model
TBD	Track Before Detect
TF32	TensorFloat-32 numeric format
UAV	Unmanned Aerial Vehicle
UMAP	Uniform Manifold Approximation and Projection
USD	Universal Scene Description
ViT	Vision Transformer
VN	Vector Neuron

1 Introduction

1.1 Motivation and Context

The rapid growth of small unmanned aerial vehicles (UAVs) and the increasing need to monitor airspace pose significant challenges for security systems. Reliably distinguishing the flying objects that share the lower atmosphere (drones, birds, helicopters, and airplanes) underpins airport security, vital infrastructure protection, correctional facility perimeter security, and wildlife monitoring. The stakes are concrete: drone incursions have repeatedly closed major airports [1], and contraband flown over prison walls has made perimeter airspace a routine operational blind spot [2].

Conventional detection uses dedicated sensing hardware like RADAR, radio-frequency receivers, or directional microphones. These are costly and have limits in range, angular accuracy, and robustness to interference. However, sites requiring this capability already operate dense networks of fixed, overlapping cameras. Their feeds are processed frame by frame in 2D, discarding both the multi-view geometry the cameras provide and the motion of tracked objects [3]. Multi-view reconstruction evolving over time exposes motion cues that are more discriminative than per-frame 2D or static 3D appearance. For example, rigid drone flight and a bird's periodic wing-beat separate more clearly in time than in a single frame [4].

Across the board, current paradigms do not jointly exploit *both* the spatial signal (multi-view geometric constraints) *and* the temporal signal (object motion), a gap this thesis addresses. We therefore deploy a ground-based network of $N \geq 3$ overlapping RGB cameras and, from their synchronized feeds alone, recover 3D occupancy and per-object class (one of drone, bird, helicopter, or airplane). Intrinsics and extrinsics are assumed known and fixed, and the system must run in real time on standard camera-computing hardware under an end-to-end per-frame latency budget below 100 ms, a constraint that shapes the compact models and design choices adopted here.

This regime is demanding: targets appear at long range and at low pixel resolution against a texture-less sky, which weakens the photometric cues that multi-view geometry relies on; their apparent scale varies sharply across views; and real deployments must contend with calibration drift.

We build on 3D Gaussian Splatting (3DGS), which represents a scene as learnable 3D Gaussian primitives (position, shape, opacity, and color) rendered via fast tile-based rasterization, enabling real-time reconstruction. Feed-forward 3DGS and its 4D extensions now reconstruct dynamic scenes in real time, yet classification grounded directly in temporal Gaussian representations remains underexplored; to the best of our knowledge, no prior work applies such temporal features to the classification of flying objects. The Drone vs Bird Detection Challenge is the field's primary community benchmark [3], and we aim to advance it with multi-camera 3D reconstruction and to steer it towards a multiple-view paradigm.

1.2 Problem Statement

Given a network of $N \geq 3$ fixed RGB cameras with overlapping fields of view monitoring a central airspace volume, we address the following problem:

> *How can we reconstruct and classify flying objects (drones, birds, helicopters, and airplanes) in 3D space using only synchronized RGB camera feeds, while leveraging the temporal dynamics of 3D Gaussian representations to improve classification beyond what is achievable with per-frame 2D or static 3D approaches?*

1.3 Research Questions

This thesis investigates the following research questions:

1. **RQ1: Synthetic Dataset Generation and Annotation.**
   How can we generate a synchronized multi-camera dataset of flying objects with full ground-truth annotations (RGB, instance masks, depth, 3D trajectories, and camera intrinsics/extrinsics) suitable for training and evaluating feed-forward 3DGS reconstruction and 4D temporal classification?
2. **RQ2: 3D Reconstruction from Sparse Multi-View RGB.**
   How can feed-forward Gaussian Splatting methods be adapted to reconstruct small, distant flying objects from sparse multi-camera views against texture-less sky backgrounds?
3. **RQ3: Temporal Dynamics for Classification.**
   Can temporal changes in 4D Gaussian parameters (position, scale, rotation, opacity) provide discriminative features for classification that improve upon static 3D or 2D appearance-based methods?
4. **RQ4: End-to-End System Performance.**
   What classification accuracy can the integrated pipeline achieve on held-out synthetic test data, both in-distribution and out-of-distribution, and how does it compare to 2D detection baselines?

1.4 Approach

Our approach integrates multiple components into an end-to-end pipeline, illustrated in Fig. 1.1.

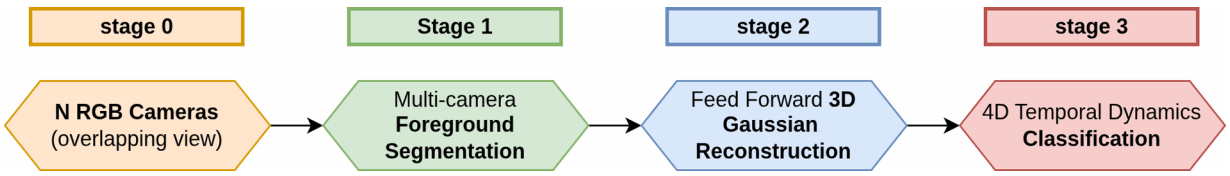


Figure 1.1: Overview of the proposed multi-camera 4D classification pipeline for flying object classification and 3D reconstruction. Stage 0 acquires synchronized multi-camera RGB, Stage 1 extracts foreground masks, Stage 2 performs feed-forward 3D Gaussian reconstruction, and Stage 3 classifies the temporal Gaussian sequence. Stages 0–2 are engineering contributions; Stage 3 is the core academic contribution.

We generate synthetic training data in NVIDIA Isaac Sim [5], simulating animated drones, birds, helicopters, and airplanes across a diverse range of scenes. The simulator emits synchronized multi-camera RGB (Stage 1) images, instance-segmentation masks, depth, per-object six-degrees-of-freedom (6-DoF) poses, and the exact camera intrinsics and extrinsics.

Three named subsets share the same class set and re-constructor (`DepthSplat-OC`): Sim-1 is a distance-sweep static rig, Sim-2 is an outdoor circle rig with a rig radius of 100 m and a monitored-volume radius of 25–50 m, and Sim-3 is a line-formation out-of-distribution probe spanning roughly 80–380 m. All cameras render at 2560×1440 pixels. Sim-1 and Sim-2 are split 50/25/25 identity-disjoint for training and in-distribution evaluation, whereas Sim-3 renders only the held-out test identities (described in §6.2).

Stage 1 of the pipeline is multi-camera foreground segmentation. A learned segmentation module is part of the system design but is not implemented in this thesis; instead, the simulator's instance-segmentation masks are directly used as ground-truth foreground, isolating Stages 2 and 3 from segmentation error and allowing them to be trained and evaluated on their own merits. A prototype direction for a track-before-detect approach based on ray-march voxel voting is documented in Appendix C.

In Stage 2 we apply feed-forward Gaussian Splatting to reconstruct 3D Gaussian representations from the multi-view RGB frames, supervised by a mask-gated photometric loss that uses the Stage 1 masks to confine supervision to the foreground regions of each view. The resulting temporal Gaussian sequences are then passed to our Stage 3 classifier, `MambaSplat-4D`, which pairs a rotation-equivariant VN-Transformer spatial encoder with a Mamba temporal encoder to provide provable rotation invariance over the 4D Gaussian sequence. The classifier extracts discriminative features from the temporal evolution of the Gaussian position, opacity, color, scale, and quaternion attributes and assigns each object to one of the four classes (described in Chapter 5).

1.5 Key Contributions

This thesis makes the following contributions:

1. **AeroSplat-4D Synthetic Dataset and Generation Pipeline.** A multi-camera Isaac Sim renderer that emits synchronized RGB, instance-segmentation masks, depth, 3D trajectories, and exact camera calibration across four classes (bird, drone, helicopter, airplane), together with in-distribution training and evaluation splits and a dedicated out-of-distribution test set.
2. **Feed-Forward 3DGS Adaptation for Thin, Distant Aerial Targets.** `DepthSplat-OC`, a feed-forward 3DGS variant for sparse wide-baseline rigs. A mask-gated photometric loss decouples reconstruction from the background, and the cost volume is adapted to support matching across low-texture sky regions.
3. **MambaSplat-4D: Rotation-Invariant 4D Gaussian Classifier.** The core academic contribution is a classifier that operates directly on temporal sequences of 3D Gaussian representations. Rotation invariance is enforced at the architecture level, not imposed through data augmentation. Six feature-mode ablations isolate the contributions of position, opacity, color, scale, and quaternion attributes.
4. **End-to-End Pipeline and Evaluation.** An integrated system that composes Stages 1–4, benchmarked against 2D detection baselines under in-distribution and out-of-distribution regimes on synthetic data. The reference implementation is released as open-source.

1.6 Thesis Scope and Limitations

We separate three categories. *Scope* denotes the boundaries chosen deliberately before the work began. *Limitations* are constraints encountered during the work and are discussed in Chapter 7. *Future work* comprises extensions that we endorse but do not pursue here, collected in §8.2. Each item below is filed under exactly one of these categories.

1.6.1 In Scope

- Classification and 3D reconstruction of flying objects.
- A four-class taxonomy comprising drone, bird, helicopter, and airplane, which extends naturally as further rendered assets become available.
- Scenarios with $N \geq 3$ overlapping-view cameras observing objects that traverse a monitored central volume.
- Synthetic Isaac Sim evaluation only, covering the Sim-1 and Sim-2 in-distribution test splits and the Sim-3 out-of-distribution renders.
- Training and evaluation under simulator ground-truth foreground masks.

1.6.2 Out of Scope

The following are excluded entirely; each constitutes a separate research problem orthogonal to our research questions:

- Fine-grained drone-model recognition and bird-species classification, which require dedicated datasets and appearance models beyond the scope of this work.
- Long-range tracking beyond the monitored volume, which falls outside the geometric regime for which the camera rig is designed.
- Adversarial evasion scenarios, which constitute a security-robustness question distinct from classification accuracy.
- Privacy and legal aspects of surveillance systems, which belong to a regulatory and ethical domain beyond this technical contribution.

- Real-time guarantees with formal verification; we report measured latency, but make no formally verified timing claim.

The following are likewise out of scope for this thesis, but are endorsed as future work:

- Real-world (sim-to-real) evaluation, deferred to §8.2.
- RGB-only foreground segmentation at inference time; the pipeline currently consumes simulator ground-truth masks, and an inference-time prototype is documented in Appendix C.
- Object detection and localization built on multi-camera foreground segmentation, a natural extension of the aforementioned segmentation work.

1.7 Document Structure

The remainder of this thesis is organized as follows:

- **Chapter 2** defines the problem setting and assumptions in detail, formalizing the multi-camera monitoring scenario, sensing constraints, and evaluation criteria.
- **Chapter 3** provides essential background on the three main pillars of the approach in order to contextualize it.
- **Chapter 4** presents a detailed review of related work, organized by technical component.
- **Chapter 5** describes the complete pipeline in detail and justifies each major design choice on the basis of the comparative analysis presented in Chapter 3.
- **Chapter 6** presents the experimental methodology, datasets, evaluation metrics, baseline comparisons, and ablation studies addressing each research question.
- **Chapter 7** discusses findings, limitations, failure cases, computational performance, and ethical considerations.
- **Chapter 8** concludes with a summary of contributions and directions for future work.

2

Problem Setting and Assumptions

This chapter formalizes the problem setting, hardware configuration, sensing assumptions, and evaluation criteria for the multi-camera flying-object detection-and-classification system developed in this thesis. The goal is to make the scenario constraints explicit enough that the architectural choices in later chapters can be traced back to the requirements imposed by the monitoring task.

2.1 Scenario: Multi-Camera Airspace Monitoring

2.1.1 Monitored Volume

We consider a ground-based surveillance system that observes a central airspace volume $\mathcal{V} \subset \mathbb{R}^3$ in a fixed world coordinate frame. The monitored region may be polygonal or circular in horizontal extent, with a radius of approximately 25–50 m, and extends vertically from ground level to an altitude of approximately 25–50 m. In practice, this volume is defined by the intersection of the camera frustums, because classification and reconstruction are performed only where the target is visible from enough viewpoints to support multi-view reasoning.

The scenario geometry relies on three related quantities that must be kept separate. The camera-rig radius $R_{\text{cam}} = 100\,\text{m}$ denotes the radius of the ring on which the cameras are mounted. The monitored-volume radius $R_{\text{vol}} = 25\text{–}50\,\text{m}$ denotes the central region in which the system performs classification and reconstruction. The target-distance envelope of 2–380 m denotes the union of camera-to-target distances exercised across the three simulation scenarios; it is a range of observed distances rather than an in-volume position. In the Sim-2 circular rig, where $R_{\text{cam}} = 100\,\text{m}$ and $R_{\text{vol}} = 25\text{–}50\,\text{m}$, an in-volume target lies roughly 50 m from any camera. Sim-1 spans the lower portion of the envelope through a controlled 2–280 m sweep, Sim-2 lies in its mid-range, and Sim-3 reaches the upper, out-of-distribution extreme near 380 m.

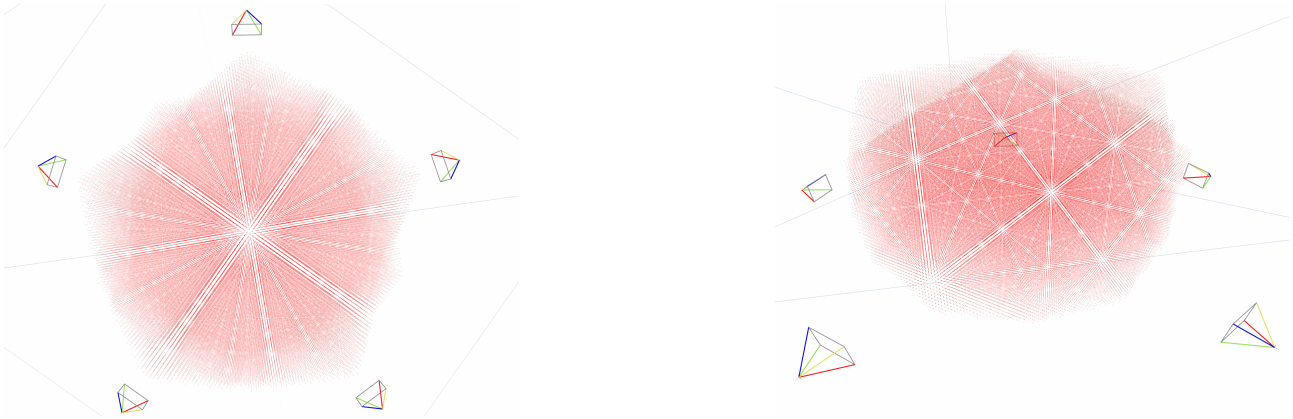
Targets are assumed to enter, traverse, and exit $\mathcal{V}$. The system therefore does not attempt to classify or reconstruct an object over an unbounded trajectory. Its operating claim is restricted to the portion of the trajectory for which the object lies inside the monitored volume and satisfies the multi-view visibility assumptions stated below.

2.1.2 Multi-Camera Configuration

The system uses $N \geq 3$ RGB cameras positioned on the ground and oriented upward toward a shared central region. Each camera is modeled as a pinhole camera with a resolution of 2560×1440 pixels, 49.1° or 30° of Field of View (FoV), and a capture rate of 30 FPS. The cameras are statically mounted on rigid fixtures such as poles, buildings, or towers. Because the datasets used in this thesis are simulated, the cameras are assumed to be perfectly synchronized in the primary workflow.

The camera arrangement is inward-looking: each camera observes the same central airspace from a different viewpoint, and every point in the monitored volume is assumed to be visible to at least three cameras. This requirement is the minimum required to make the system a multi-view reconstruction problem rather than a collection of independent monocular classifiers. The configuration is representative of fixed surveillance in-

stallations for perimeter security, airport monitoring, or protected-area surveillance. Figure 2.1 illustrates the monitored-volume layout.



(a) Top view (b) Side view

Figure 2.1: Occupancy-grid representation of monitored volume $\mathcal{V}$. Cameras are positioned around the perimeter and look upward into the volume where flying objects are reconstructed and classified.

2.1.3 Flying-Object Characteristics

The thesis considers four object categories: drones, birds, helicopters, and airplanes. Drones and birds form the primary classification target. However, in this thesis, helicopters and airplanes remain in scope as additional in-set classes, allowing the classifier and evaluation protocol to operate on a four-class taxonomy rather than on a binary drone-versus-bird distinction.

Drones are primarily represented by quadcopters and multicopters. Their motion is constrained by rigid-body flight dynamics, often producing smooth trajectories and hover-capable behavior. Their appearance is also relatively rigid and symmetric. Although propeller blades may be resolvable at short range or with future high-resolution or zoom-capable cameras, they are not assumed to be visible across the operating envelope used in this thesis. At typical long ranges, a drone may only subtend about 10–15 pixels [6], so the classification problem cannot rely on propeller visibility and must instead exploit 3D motion and multi-view spatial features.

Birds differ from drones in both kinematics and appearance. Their flight includes flapping, gliding, and other varied motions, while their shape is deformable due to wing articulation and biological texture. Helicopters are represented as mostly static bodies with a single main rotor and a comparatively simple structure. Airplanes are represented as mostly static bodies with wings and a tail, with some assets also carrying propellers.

Each asset is normalized to 1m so that its apparent pixel extent within a given range is well defined. For birds and airplanes, this dimension is wingspan, for quadcopters it is the motor-to-motor diagonal, and for helicopters it is nose-to-tail body length. Absolute apparent size is not treated as a discriminative class feature.

2.1.4 Scene Constraints

The monitoring scenario is intentionally difficult for conventional appearance-based methods. The open sky provides a largely texture-less background, leaving few reliable features for photometric depth estimation or feature matching. Targets may occupy only a small number of pixels per camera when they are far from the rig, and the apparent scale of the same object can differ substantially across cameras because each view observes the object from a different distance and angle.

The cross-scenario target-distance envelope is 2–380 m. This range should not be interpreted as the radius of the monitored volume; rather, it describes the camera-to-target distances that appear across the simulated regimes. Supporting this envelope requires wide camera baselines because the system must remain useful as long as a target remains visible in the camera views and until the object becomes unrecognizable to the system. These constraints motivate the use of multi-view geometrical consistency and temporal dynamics instead of relying solely on single-frame appearance features.

2.1.5 Pixel Density and Resolution Limits

A central constraint of the long-range monitoring setting is that distant objects have very low spatial resolution in each camera view. IEC 62676-4 defines pixel-density thresholds for surveillance tasks, including 25 pixels per meter (PPM) for detection, 125 PPM for recognition, and 250 PPM for identification [7]. With a 2K camera at 2560×1440 pixels and a $50°$ horizontal field of view, the approximate pixel density is $\mathrm{PPM}(d) \approx 2745/d$, where d is the object distance in meters. At $d = 200$ m, this gives approximately 13.7 PPM, which falls below the detection threshold. A 1 m drone at that distance therefore occupies only about 13.7 pixels across its characteristic dimension.

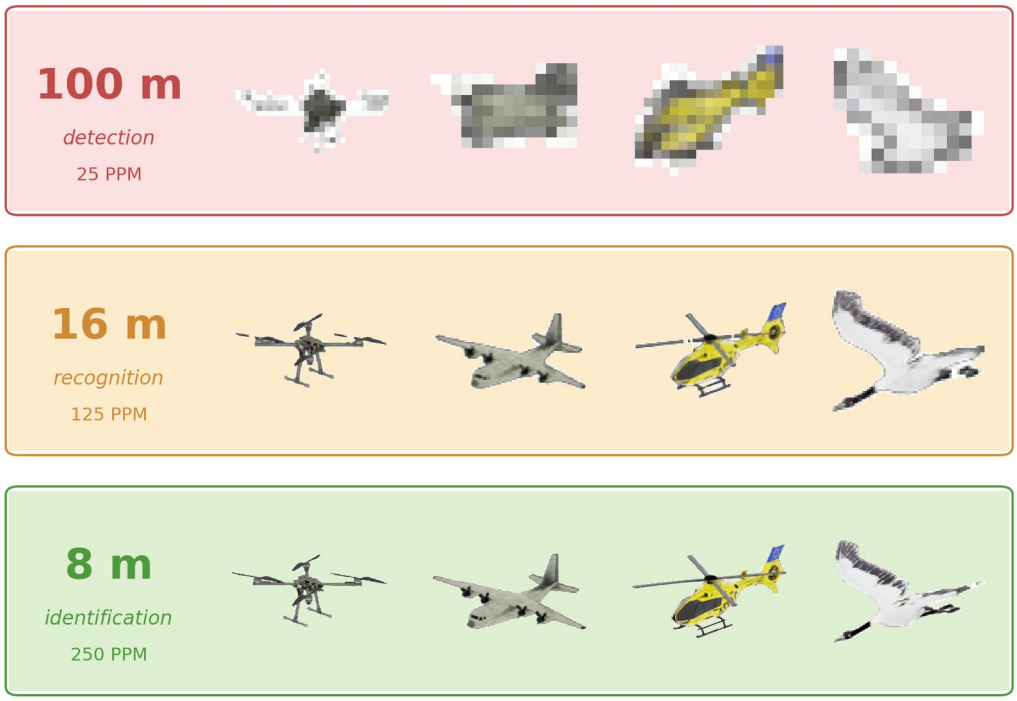


Figure 2.2: Pixel density decreases rapidly with range, placing long-range flying-object classification below conventional recognition thresholds [7].

This spatial limit is also an information-theoretic limit. The imaging channel is bounded by spatial bandwidth and signal-to-noise ratio, and targets spanning less than roughly 10–20 pixels lose many of the class-specific high-frequency features that would otherwise distinguish rigid mechanical objects from deformable biological ones [8]. Noise further reduces the effective bit depth of each pixel. Temporal integration offers a partial remedy: integrating over T frames can recover discriminative information when motion or rotation reveals new views of the object [9]. The strategy in this thesis is therefore to compensate for limited per-frame spatial bandwidth by leveraging multi-view 3D reconstruction and exploiting the geometry and temporal evolution of that 3D representation to classify targets that remain spatially unresolved in any individual view.

2.2 Assumptions

2.2.1 Camera Calibration

The primary workflow assumes that the intrinsic parameters $\mathbf{K}_c$ of each camera $c \in \{1, \ldots, N\}$ are known, including focal lengths, principal points, and distortion parameters. In a practical deployment, these quantities can drift with temperature, lens changes, or mechanical stress, but such environmental drift is not modeled in the main experiments.

The extrinsic parameters $[\mathbf{R}_i \mid \mathbf{t}_i]$ are likewise assumed to be known and fixed during the primary workflow. They may be initialized through offline Structure-from-Motion, manual control-point alignment, installation blueprints, or GPS-based camera-position estimates. Camera positions $\mathbf{t}_i$ are often easier to recover from deployment metadata than orientations $\mathbf{R}_i$, which are more prone to installation error and subsequent drift. This thesis acknowledges such extrinsic drift as a real-world concern, but does not refine camera poses online.

Online extrinsic refinement is left for future work, where the accumulated 3D positions of flying assets could serve as calibration information.

2.2.2 Temporal Synchronization

Real multi-camera deployments commonly synchronize cameras through software-based NTP, with residual offsets on the order of ±10 ms on typical networks [10], or through hardware-supported PTP (IEEE 1588), which achieves sub-microsecond residuals [11]. At typical object speeds of 0–15 m/s, the NTP regime corresponds to a per-camera positional uncertainty of up to ∼15 cm, whereas the PTP regime produces sub-millimeter uncertainty that is negligible for the scenarios considered here. In a real system, residual synchronization error would therefore need to be treated as a per-camera time offset, with its magnitude depending on the synchronization method. The simulated datasets used in this thesis assume perfect synchronization, so temporal misalignment is not part of the primary experimental workflow.

2.2.3 Data Availability During Training and Inference

The training phase uses synthetic data generated in NVIDIA Isaac Sim [5]. For each camera and frame, the simulator provides synchronized RGB images, ground-truth depth maps, pixel-wise foreground segmentation masks, 3D object trajectories in the world frame, hierarchical object class labels, and exact camera intrinsics and extrinsics. The hierarchical labels encode both superclass and model identity, although the evaluation in this thesis focuses on superclass-level four-class classification.

The inference-time setting is intentionally more restrictive. At deployment time, the system assumes access only to synchronized RGB frames from all cameras, known intrinsics $\mathbf{K}_i$, and known extrinsics $[\mathbf{R}_i \mid \mathbf{t}_i]$. RGB-only foreground segmentation at inference time is not solved in this thesis. Stages 2 and 3 are supervised under simulator ground-truth masks, and the gap to inference-time segmentation is documented as future work in Appendix C. Sim-to-real transfer therefore remains a central challenge and is evaluated only within the limits described in Chapter 6.

2.3 Evaluation Requirements

The system is evaluated along three axes, classification, 3D reconstruction, and computational performance, that together determine whether the pipeline meets the demands of the monitoring task. Classification quality assesses whether the system can separate the four object categories; reconstruction fidelity assesses whether the intermediate 4D representation faithfully captures the target; and computational performance assesses whether the pipeline can run within the edge budget set out below. We require classification to be assessed across the full four-class taxonomy rather than a binary drone-versus-bird decision, so that the evaluation exposes class-specific biases rather than hiding them in an aggregate score. The reconstruction axis is judged by the fidelity of rendered novel views, and the computational axis by end-to-end latency, throughput, and memory footprint, against the edge constraints stated below.

These requirements fix what the system must achieve and why; the precise metric definitions and the consolidated metric set are deferred to the experimental protocol of the Experiments chapter (§6.1), where every reported number is measured. Stating each metric once there, rather than separately in this chapter and again in the Method chapter, keeps a single canonical definition for it.

All evaluation in this thesis is conducted on three synthetic regimes, two in-distribution and one out-of-distribution, defined with their rigs, scenes, and split protocol in the Method chapter (§5.3.4); real-camera evaluation is out of scope and deferred to future work (§8.2), so the results reported here are synthetic-evaluation claims rather than completed real-world deployment results.

2.3.1 Deployment Targets

The motivating deployment target is an edge system that can process multi-camera data with an end-to-end per-frame latency below 100 ms. The relevant edge hardware family is NVIDIA Jetson [12], with candidate devices including the Jetson Orin Nano Super with 8 GB of memory and 67 TOPS (sparse INT8), and the Jetson AGX Orin with 64 GB of memory and 275 TOPS (sparse INT8). This target shapes the architectural choices throughout the thesis: model capacity, memory footprint, and per-stage latency are all considered with edge deployment in mind. Development and training were carried out on an NVIDIA RTX 5090 workstation, while

the deployment-oriented claims are framed around the Jetson AGX Orin as the intended inference platform. Specific power-envelope claims are deferred to future on-device validation.

At the specified resolution and frame rate, each camera produces approximately 110 megapixels per second. A rig with N cameras therefore produces $N \times 110$ megapixels per second before compression. A shared 1 Gbps link presumes on-camera compression or other bandwidth-reduction measures. Edge deployment is treated as a design constraint guiding the architecture rather than a fully validated result, with on-device latency and power validation left to future work.

2.4 Chapter Summary

This chapter defines operating assumptions for the thesis. The system monitors a central airspace volume $\mathcal{V}$ using $N \geq 3$ upward-looking, perfectly synchronized RGB cameras. It classifies and reconstructs small distant flying objects against sky background, with drones, birds, helicopters, and airplanes forming the four in-scope classes. The sensing model assumes known intrinsics and extrinsics while recognizing calibration drift and synchronization error as real-world issues outside the primary simulated workflow.

The data model separates learning from inference. During synthetic training, the simulator provides RGB, depth, segmentation masks, trajectories, labels, and precise calibration. During real-world inference, the intended inputs are only synchronized RGB images and known calibration; inference-time segmentation and real-camera validation are deferred to future work. Evaluation is correspondingly organized around classification accuracy, reconstruction fidelity, and computational efficiency, with an edge-oriented target of NVIDIA Jetson AGX Orin and an end-to-end per-frame latency budget below 100 ms.

3 Background

This chapter reviews the three families of primitives on which the rest of the thesis builds: multi-view geometry, 3D Gaussian Splatting, and the deep-learning components that drive the temporal classifier. The Multi-View Geometry Fundamentals section 3.1 develops the projective principles that govern reconstruction from calibrated cameras [13], [14]. The 3D Gaussian Splatting Formulation section 3.2 introduces the mathematics of 3DGS as an explicit scene representation [15], [16]. Finally, the Deep Learning Primitives section 3.3 covers self-attention [17], equivariance [18], set processing [19], and state space models [20]. Together, these form the building blocks of the temporal classification pipeline.

3.1 Multi-View Geometry Fundamentals

Multi-view geometry (MVG) provides the mathematical framework for recovering 3D structure from two or more 2D images [13], [14]. This section first reviews the single-camera projection model and then introduces the multi-camera constraints that enable triangulation and silhouette-based reconstruction. Classical Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipelines [21], [22] rely on matching local image features such as SIFT [23] or ORB [24], and on propagating those correspondences to estimate camera poses and dense geometry. A well-known limitation of this approach is that local descriptors degenerate at low pixel resolutions, which is precisely the regime we address in this thesis.

3.1.1 Camera Models and Projection

The pinhole camera model treats image formation as a central projection of light through a small aperture onto an image plane [13]. A world point $\mathbf{X} = (X, Y, Z)^\top$ maps to an image point $\mathbf{x} = (u, v)^\top$ via the projection equation

$$\tilde{\mathbf{x}} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\tilde{\mathbf{X}} \tag{3.1}$$

where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{X}}$ are homogeneous coordinates. The projection decomposes cleanly into an intrinsic matrix $\mathbf{K}$ and an extrinsic matrix $[\mathbf{R} \mid \mathbf{t}]$ [13]. The intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ encodes the camera's internal optical properties [13]:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

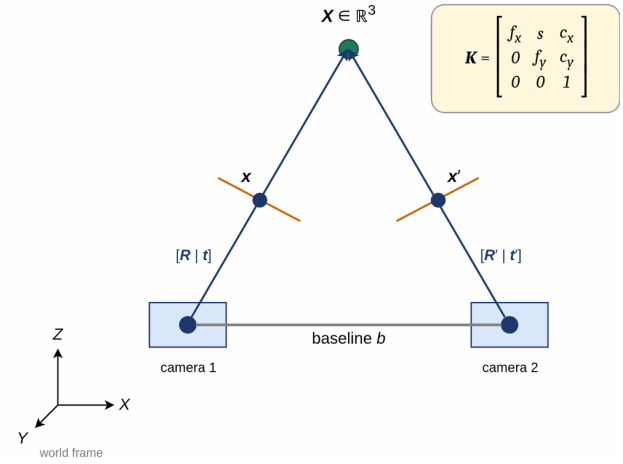


Figure 3.1: Two-view pinhole projection. Each camera maps the world point $\mathbf{X}$ along the ray through its optical center to an image point ($\mathbf{x}$ for camera 1, $\mathbf{x}'$ for camera 2). The shared intrinsics $\mathbf{K}$ collect the focal lengths f_x, f_y and principal point (c_x, c_y); the per-camera extrinsics $[\mathbf{R} \mid \mathbf{t}]$ place each camera in the world frame. The two optical centers are separated by the baseline b. Image planes are drawn in front of the pinholes (virtual-image-plane convention).

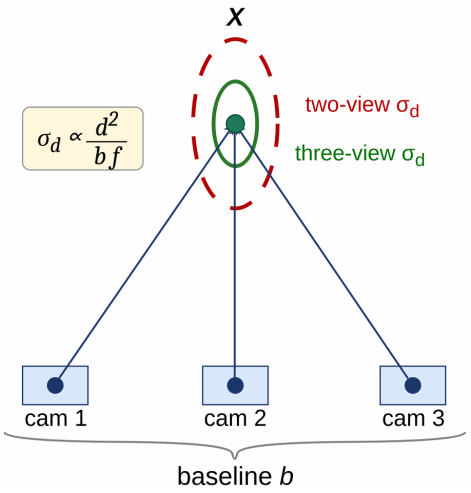


Figure 3.2: Triangulation uncertainty around $\mathbf{X}$ as a function of view count and baseline. The dashed red ellipse shows the uncertainty volume from any pair of cameras alone; adding a third camera reduces it to the green ellipse. The bound $\sigma_d \propto d^2/(b\,f)$ links depth uncertainty σ_d to scene depth d, baseline b, and focal length f, so wider baselines and additional views both shrink the ellipse.

Here f_x and f_y are the focal lengths in pixels, (c_x, c_y) is the principal point, and s is the sensor skew, which is effectively zero for modern cameras. The intrinsics are fixed once a calibration procedure has been carried out [25], [26]. The extrinsic matrix $[\mathbf{R} \mid \mathbf{t}]$ (with $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$) then transforms world points into the camera frame [13]. Figure 3.1 sketches the two-view configuration used throughout this chapter.

3.1.2 Triangulation and 3D Point Recovery

Triangulation recovers a 3D point $\mathbf{X}$ from a pair of image correspondences $\mathbf{x}$ and $\mathbf{x}'$ once the camera matrices are known [13]. Geometrically, the reconstructed point is the intersection of the two rays back-projected from each image point. In practice, image noise prevents those rays from meeting exactly, so triangulation methods minimize re-projection error rather than searching for an exact intersection. Linear triangulation starts from the constraint $\tilde{\mathbf{x}} \times \mathbf{P}\tilde{\mathbf{X}} = \mathbf{0}$, which yields two independent equations per view [13]. Stacking these equations over multiple views produces an overdetermined linear system $\mathbf{A}\tilde{\mathbf{X}} = \mathbf{0}$, which is solved via singular value decomposition by selecting the singular vector associated with the smallest singular value [13].

The accuracy of triangulation scales with the baseline-to-depth ratio: wider baselines yield more precise depth estimates [13]. The relationship between depth uncertainty σ_d, scene depth d, baseline b, focal length f, and pixel noise σ is captured by the bound $\sigma_d \propto d^2/(b \cdot f)$; adding a third view further tightens this bound by overdetermining the linear triangulation system (Fig. 3.2). The operating envelope that this bound constrains is specified in the Problem Setting chapter 2.

3.1.3 Visual Hulls and Silhouette Consistency

A visual hull is the intersection of the viewing cones swept out by an object's silhouettes in each calibrated view [27]. Formally, $\mathcal{V} = \bigcap_{i=1}^{N} \mathcal{C}_i$, where each $\mathcal{C}_i$ is the cone defined by silhouette S_i. The visual hull is therefore a feature-free outer bound on the object's 3D extent, derived purely from silhouettes rather than from dense correspondences. Space carving extends this idea by removing voxels whose photometric projections are inconsistent across views, again without requiring explicit point correspondences [28]. Although this thesis does not perform explicit silhouette reconstruction, the Stage-2 mask-gated photometric loss applies the same multi-camera consistency principle that underlies visual hulls: a 3D structure is accepted only when it agrees across multiple views.

3.1.4 Multi-Camera Geometry under Texture-less Backgrounds

In purely two-view triangulation, depth becomes ambiguous in texture-less regions because photometric matching has no signal to lock onto. Introducing a third camera resolves this ambiguity (Fig. 3.2): the additional view discriminates between depth hypotheses that the first two cameras alone cannot separate.

3.2 3D Gaussian Splatting Formulation

This section lays out the mathematics behind the 3D Gaussian representation used in our reconstruction stage. We describe how each Gaussian is parameterized and how it transforms under rotation, how it is projected and composited into a rendered image, and how its parameters are optimized, before connecting the representation to flying-object reconstruction.

Gaussian Primitive Parameterization. A scene is parameterized as a set $\mathcal{G} = \{G_i\}_{i=1}^{N}$ of N Gaussian primitives [15]. Each primitive carries a mean $\boldsymbol{\mu} \in \mathbb{R}^3$ that places the Gaussian center in world coordinates, and a covariance $\boldsymbol{\Sigma}$ that is factored into a scale vector $\mathbf{s} \in \mathbb{R}^3$ and a unit quaternion $\mathbf{q} \in \mathbb{R}^4$. The covariance is reconstructed from these as

$$\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T \tag{3.3}$$

with $\mathbf{S} = \mathrm{diag}(\mathbf{s})$ and $\mathbf{R}$ obtained from the unit quaternion $\mathbf{q}$. This factorization has two practical advantages: it guarantees that $\boldsymbol{\Sigma}$ is positive semi-definite, and it uses only seven parameters (three for scale, four for the quaternion) rather than the six independent entries of a general symmetric matrix, while keeping rotation explicit through the unit-quaternion constraint. We adopt the scalar-first quaternion convention (w, x, y, z) throughout.

Each primitive also carries an opacity $\alpha \in [0, 1]$ that controls its contribution to the rendered image (low opacity: semi-transparent surfaces, high opacity: solid surfaces) and a color encoded as spherical-harmonic (SH) coefficients. The SH degree ℓ governs the angular resolution of view-dependent color, with $(\ell + 1)^2$ coefficients per channel: degree 0 captures purely diffuse color, while higher degrees encode specular reflections. At render time, the SH coefficients are evaluated for the current viewing direction to produce a color $\mathbf{c} \in \mathbb{R}^3$. The full per-Gaussian parameter vector is therefore

$$\theta_i = \{\boldsymbol{\mu}_i, \mathbf{s}_i, \mathbf{q}_i, \alpha_i, \mathbf{c}_i^{\mathrm{SH}}\} \tag{3.4}$$

where $\mathbf{c}_i^{\mathrm{SH}}$ denotes the SH coefficient tensor. The Gaussian is the atomic primitive of 3DGS, and a typical scene may contain on the order of $\approx 10^5$–10^7 such primitives. Under a rigid transformation, translation rotates $\boldsymbol{\mu}$, rotation $\mathbf{R}$ rotates $\boldsymbol{\mu}$ and pre-multiplies $\mathbf{q}$, while scale and opacity are rotation-invariant. The $(\mathbf{R}, \mathbf{S})$ factorization is what later lets Vector Neurons [29] in the Stage-3 classifier treat each Gaussian as a set of co-rotating 3-vectors. Figure 3.3 illustrates these parameters on three overlapping Gaussians, together with the camera and screen tile through which they are eventually rasterized.

Action of SO(3) on a Gaussian. An external rotation $\mathbf{R} \in SO(3)$ transforms the pair $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ jointly, while scale, opacity, and DC color are left unchanged. The update rule is

$$\boldsymbol{\mu} \mapsto \mathbf{R}\boldsymbol{\mu}, \quad \boldsymbol{\Sigma} \mapsto \mathbf{R}\,\boldsymbol{\Sigma}\,\mathbf{R}^\top, \quad \{\mathbf{s}, \alpha, \mathbf{c}^{\mathrm{SH}}\} \text{ fixed} \tag{3.5}$$

where $\boldsymbol{\Sigma} = \mathbf{R}(\mathbf{q})\,\mathrm{diag}(\mathbf{s})^2\,\mathbf{R}(\mathbf{q})^\top$ is the Gaussian covariance built from the per-Gaussian rotation quaternion $\mathbf{q}$ and scales $\mathbf{s}$.

Rendering proceeds by splatting each Gaussian onto the image plane as a 2D ellipse rather than by volumetric ray marching [15]. The 3D-to-2D projection follows the Elliptical Weighted Average (EWA) splatting framework [16]: given camera extrinsics $[\mathbf{R}_c \mid \mathbf{t}_c]$ and the local affine-approximation Jacobian $\mathbf{J}$, the projected 2D covariance is

$$\boldsymbol{\Sigma}' = \mathbf{J}\mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^T\mathbf{J}^T \tag{3.6}$$

where $\mathbf{W}$ is the viewing transformation matrix and $\boldsymbol{\Sigma}' \in \mathbb{R}^{2\times 2}$ defines the elliptical image-plane footprint (Fig. 3.4). The contribution of the Gaussian at pixel $\mathbf{p}$ is then

$$G(\mathbf{p}) = \exp\left(-\tfrac{1}{2}(\mathbf{p} - \boldsymbol{\mu}')^T\boldsymbol{\Sigma}'^{-1}(\mathbf{p} - \boldsymbol{\mu}')\right) \tag{3.7}$$

with $\boldsymbol{\mu}'$ the projected 2D mean. To make rasterization efficient, the image is tiled into 16×16 pixel blocks—one CUDA thread block per tile, mapping one thread to one pixel so that 256 threads fill the block and align with

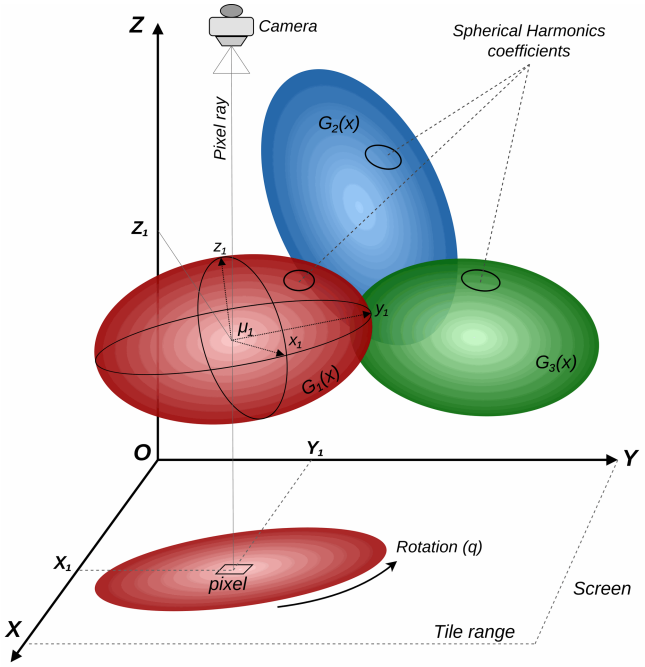


Figure 3.3: 3D Gaussian primitives in the world frame (X, Y, Z). On primitive G_1 the local-frame axes (x_1, y_1, z_1) visualize the scale vector $\mathbf{s}$ and the orientation set by the quaternion $\mathbf{q}$, with the mean $\boldsymbol{\mu}_1$ at the center. The small inner lobes label the spherical-harmonic coefficients $\mathbf{c}^{\mathrm{SH}}$ that encode view-dependent color. A pixel ray from the camera intersects the screen tile beneath the Gaussians, indicating the projected 2D ellipse onto which G_1 is splatted. Modified from [30].

GPU warp scheduling [15]. Gaussians are depth-sorted and assigned to the tiles whose extents they overlap, and each tile processes only the Gaussians relevant to it. This tile-based pipeline enables highly parallel GPU rasterization [15] and enables fast feed-forward training regimes as outlined in Section 4.3.2.

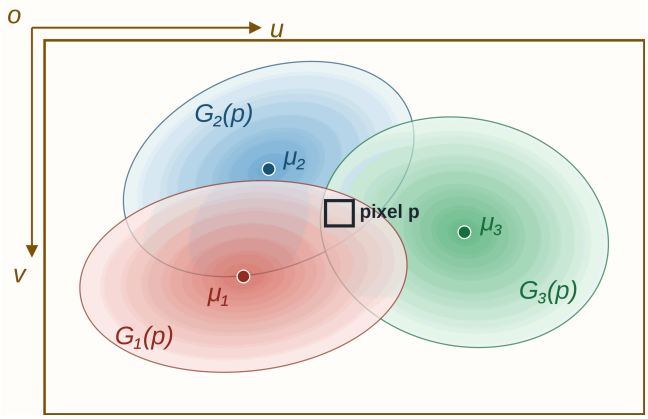


Figure 3.4: Image-plane footprint after EWA projection. Three Gaussians with 2D projected means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3$ and covariances $\boldsymbol{\Sigma}'_i$ overlap a single pixel $\mathbf{p}$ in the image plane (u, v). The per-Gaussian value $G_i(\mathbf{p})$ at that pixel is then accumulated across the overlapping primitives within the tile during rasterization.

Alpha Compositing. Within each tile, the depth-sorted Gaussians $\{G_1, \ldots, G_M\}$ are composited front-to-back. The rendered color at a pixel is

$$C = \sum_{i=1}^{M} c_i \alpha_i G_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - \alpha_j G_j(\mathbf{p})) \tag{3.8}$$

where the transmittance $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j G_j(\mathbf{p}))$ expresses the fraction of light still reaching Gaussian i. Rendering terminates early once accumulated opacity approaches one, which keeps the per-pixel cost bounded

even in dense scenes (Fig. 3.5).

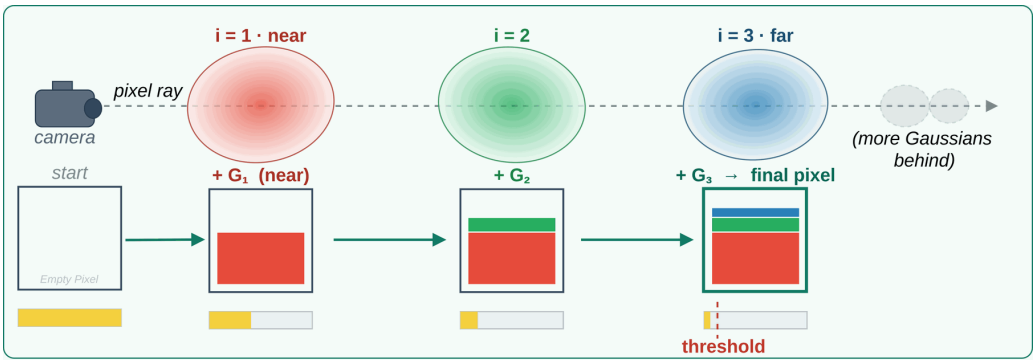


Figure 3.5: Front-to-back α-compositing along a single pixel ray. Gaussians ordered by depth (i=1 nearest, i=3 farther) contribute in turn to the pixel color shown in the bar below; the yellow opacity strip tracks the accumulated transmittance, and rendering terminates once it crosses the threshold mark even when more Gaussians lie behind.

Optimization Objective. Standard 3DGS training minimizes a photometric objective that combines an L1 term with a perceptual term [15]:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} \tag{3.9}$$

where $\mathcal{L}_1 = \|C_{\text{pred}} - C_{\text{gt}}\|_1$ is the pixel-wise L1 error and $\mathcal{L}_{\text{D-SSIM}} = 1 - \text{SSIM}(C_{\text{pred}}, C_{\text{gt}})$ uses the structural similarity index [31]. The weight $\lambda = 0.2$ balances the two terms, and the D-SSIM component encourages perceptually coherent reconstructions rather than only pixel-accurate ones. Because the entire rasterizer is differentiable, gradient descent over all Gaussian parameters can be performed end-to-end from multi-view image supervision.

3.2.1 Connection to Flying Object Reconstruction

3DGS is well matched to the demands of flying-object reconstruction for several reasons. Its explicit primitives expose 3D positions and extents directly, which is useful for the downstream temporal classification task. Feed-forward 3DGS predictors (Section 4.3.2) can produce a reconstruction in a single forward pass per multi-view frame, which matches the per-frame budget required to keep up with the camera stream. The decomposed parameterization (position, scale, rotation, opacity, and color) provides the temporal classifier with a set of variables whose temporal changes are inherently discriminative between drones, birds, helicopters, and airplanes. Gaussians also require fewer views than dense stereo to recover a serviceable reconstruction, which matches the sparse ground-rig camera configuration. The explicit primitive representation is also storage-efficient: a typical flying-object reconstruction requires on the order of 10^3 Gaussians (∼14 kB in compact form), orders of magnitude smaller than a dense voxel grid or a neural-radiance-field weight file covering the same volume. The reconstructed Gaussians ultimately serve as input tokens to the classifier described in the Deep Learning Primitives section 3.3.

3.3 Deep Learning Primitives

This section reviews the architectural components used to classify the 4D temporal sequence of Gaussian primitives. We cover three complementary mechanisms: attention as a content-based aggregation operator, the symmetry constraints required for invariant and equivariant set processing, and state-space models as linear-time alternatives to attention for long temporal sequences.

3.3.1 Self-Attention and Transformer Architecture

Self-attention re-weights each input dynamically based on the relevance of every other input to a query [17]. Given a sequence of N tokens of dimension d, three learned linear projections produce queries $\mathbf{Q} \in \mathbb{R}^{N \times d_k}$, keys $\mathbf{K} \in \mathbb{R}^{N \times d_k}$, and values $\mathbf{V} \in \mathbb{R}^{N \times d_v}$. Scaled dot-product attention is then defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{3.10}$$

where the $\sqrt{d_k}$ factor prevents the softmax from saturating and the gradient from vanishing as the dimensionality grows. Each query is compared against all keys to produce a distribution over the values from which the output is computed. Multi-head attention extends this by running H parallel attention operations in different learned subspaces:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}^O \tag{3.11}$$

with each head defined as $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$. The intuition is that different heads can attend to different aspects of the input such as geometry, color, or temporal change in parallel. The Transformer [17] stacks multi-head attention together with position-wise feed-forward networks, and Vision Transformers (ViTs) [32] adapt the same machinery to images by splitting them into patches, embedding each patch linearly, and adding learnable positional embeddings. A central drawback is that self-attention scales as $O(N^2)$ in both time and memory with increasing sequence length, which motivates the SSM alternative reviewed in Section 3.3.4. Where token order matters, the classical sinusoidal positional encoding

$$\text{PE}(t, 2i) = \sin\left(\tfrac{t}{10000^{2i/d}}\right), \quad \text{PE}(t, 2i{+}1) = \cos\left(\tfrac{t}{10000^{2i/d}}\right) \tag{3.12}$$

is added to the token embeddings to inject position information into the otherwise permutation-equivariant attention operator.

3.3.2 SO(3) Equivariance and Invariance

The special orthogonal group SO(3) is the group of 3D rotations: its elements are orthonormal matrices $\mathbf{R} \in \mathbb{R}^{3\times3}$ with $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$ and $\det\mathbf{R} = 1$, forming a three-dimensional Lie group [18]. A function f is *equivariant* under SO(3) when

$$f(\mathbf{R}\,\mathbf{x}) = \mathbf{R}\,f(\mathbf{x}) \quad \forall\,\mathbf{R} \in \text{SO}(3) \tag{3.13}$$

so that the output transforms predictably together with the input [18]. A function is *invariant* under SO(3) when

$$f(\mathbf{R}\,\mathbf{x}) = f(\mathbf{x}) \quad \forall\,\mathbf{R} \in \text{SO}(3) \tag{3.14}$$

so that its output is unaffected by any rotation of the input [18]. Classification typically requires invariance: rotating a drone in space does not change its class. Equivariant intermediate representations are useful because they preserve geometric information that would be lost by an early invariance bottleneck, and a final invariant layer can still combine equivariant features into an invariant prediction. The composition of equivariant maps remains equivariant [18], which is what permits stacking equivariant blocks while keeping the global invariance guarantee at the head of the network. Equation 3.5 shows how SO(3) acts on the Gaussian primitive specifically: the mean and covariance rotate while scale, opacity, and DC color are fixed. The Frobenius inner product $\langle\mathbf{V}, \mathbf{W}\rangle_F = \text{tr}(\mathbf{V}^\top\mathbf{W})$ is itself rotation-invariant, which makes it a natural similarity measure between equivariant feature tensors. Namely: $\langle\mathbf{V}\mathbf{R}, \mathbf{W}\mathbf{R}\rangle_F = \langle\mathbf{V}, \mathbf{W}\rangle_F$.

Relation to SE(3). The special Euclidean group $\text{SE}(3) = \text{SO}(3) \ltimes \mathbb{R}^3$ extends rotations with translations, giving six degrees of freedom in total [18], [33]. This thesis uses only SO(3) invariance: translation invariance is achieved by mean-centering the Gaussian cloud before it enters the classifier (§5.6.4), which removes the translational component without requiring equivariant layers for $\mathbb{R}^3$. Scale invariance is not addressed and is noted as future work (§8.2).

3.3.3 Permutation Invariance for Set Processing

A set of Gaussians has no inherent order: reordering the primitives leaves the underlying scene unchanged. Any network operating on such data must therefore respect permutation symmetry. A function f is permutation-invariant when

$$f(\{x_{\pi(1)}, \ldots, x_{\pi(n)}\}) = f(\{x_1, \ldots, x_n\}) \tag{3.15}$$

for any permutation π. DeepSets [19] provides a constructive characterization of such functions through the decomposition

$$f(\mathcal{X}) = \rho\left(\sum_{x\in\mathcal{X}} \phi(x)\right) \tag{3.16}$$

where ϕ encodes individual set elements and ρ processes the aggregated representation. Using sum or mean pooling for the aggregation step enforces invariance by construction. In our setting, each Gaussian primitive (parameterized by mean $\boldsymbol{\mu}$, scale $\mathbf{s}$, quaternion $\mathbf{q}$, opacity α, and DC color coefficients) serves as a single set element, and a permutation-invariant encoder yields a fixed-dimensional descriptor regardless of the number of Gaussians in the scene.

3.3.4 State Space Models

Transformers exhibit quadratic complexity with respect to sequence length, which becomes prohibitive for long temporal sequences such as multi-frame Gaussian streams. State space models (SSMs) provide a linear-complexity alternative. A classical linear time-invariant SSM in continuous time is defined by

$$\frac{d\mathbf{h}(t)}{dt} = \mathbf{A}\mathbf{h}(t) + \mathbf{B}x(t) \tag{3.17}$$

$$y(t) = \mathbf{C}\mathbf{h}(t) \tag{3.18}$$

with hidden state $\mathbf{h}(t) \in \mathbb{R}^D$, scalar input $x(t)$, scalar output $y(t)$, and learned matrices $\mathbf{A} \in \mathbb{R}^{D\times D}$, $\mathbf{B} \in \mathbb{R}^{D\times 1}$, $\mathbf{C} \in \mathbb{R}^{1\times D}$. Discretizing with step size Δ yields the recurrence

$$\mathbf{h}_k = \bar{\mathbf{A}}\mathbf{h}_{k-1} + \bar{\mathbf{B}}x_k \tag{3.19}$$

$$y_k = \mathbf{C}\mathbf{h}_k \tag{3.20}$$

where $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are the discretized counterparts of the continuous parameters. The deep-learning SSMs extend classical LTI SSMs with structured $\mathbf{A}$ matrices and selective parameterization for long-range sequence modeling. Of these, Mamba introduces *selective* state spaces by making $\mathbf{B}$, $\mathbf{C}$, and Δ input-dependent while keeping the continuous $\mathbf{A}$ a fixed learned parameter:

$$\mathbf{B}_k = \mathrm{Linear}_B(x_k), \quad \mathbf{C}_k = \mathrm{Linear}_C(x_k), \quad \Delta_k = \mathrm{softplus}(\mathrm{Linear}_\Delta(x_k)) \tag{3.21}$$

so that the model can selectively propagate or forget content based on the current input. Selectivity enters the state-update matrix only through the discretization step, with $\bar{\mathbf{A}}_t = \exp(\Delta_t \mathbf{A})$ inheriting input-dependence from Δ_t while $\mathbf{A}$ itself remains fixed. This combines the linear-time efficiency of SSMs with a form of context-awareness reminiscent of attention. The full selective SSM update, including a skip connection, takes the form

$$\mathbf{h}_t = \bar{\mathbf{A}}_t\,\mathbf{h}_{t-1} + \mathbf{B}(x_t)\,x_t, \qquad y_t = \mathbf{C}(x_t)\,\mathbf{h}_t + \mathbf{D}\,x_t \tag{3.22}$$

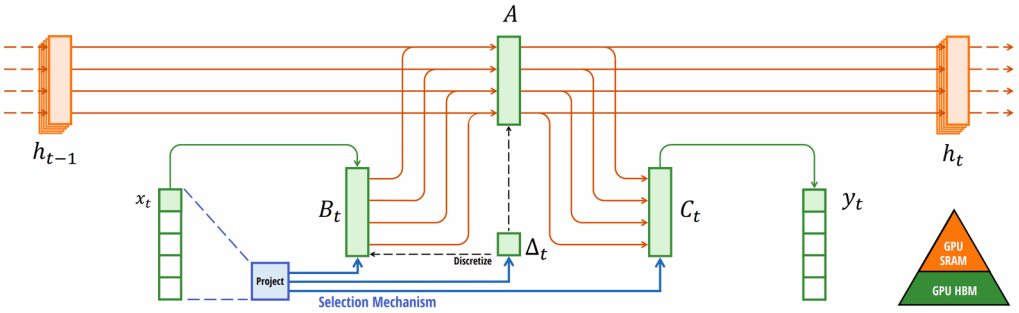


Figure 3.6: Mamba SSM block. Source [20].

The practical consequence is that Transformer attention scales as $O(N^2)$ while Mamba scales as $O(N)$ in its recurrent form [17], [20]. For the temporal-classification task in this thesis, sequences of dozens of frames combined with hundreds of Gaussian primitives per frame would push attention into uncomfortable memory regimes; the linear scaling of Mamba supports real-time temporal modeling on the same hardware. The contrast between the three relevant families is summarized in the table below; the LSTM row references the original gating formulation of Long Short-Term Memory (LSTM) networks [34].

Table 3.1: Comparison of sequence modeling approaches for temporal classification.

Model	Complexity	Context-Awareness	Edge Suitability
LSTM	$O(N)$	Limited	Yes
Transformer	$O(N^2)$	Global	Limited
Mamba	$O(N)$	Selective	Yes

4 Related Work

This chapter reviews prior art organized by technical component, surfacing the gaps that motivate the AeroSplat-4D pipeline. Three pillars structure the survey. Namely, detection and tracking of flying objects, multi-view 3D reconstruction (with its dynamic 4D extension), and classification from 3D representations. This is followed by a synthesis that maps each surveyed gap onto a thesis contribution. The pipeline-stage axis orders the sections so that each survey block is read against the corresponding stage of the proposed system rather than against an arbitrary 2D-vs-3D split.

The literature spans three field-level observations that motivate the rest of the thesis: (1) no prior work classifies flying objects via temporal dynamics of a 4D Gaussian representation; (2) the bandwidth available for multi-frame integration scales with frame count under motion [9]; and (3) existing pipelines neither exploit cross-view geometry nor a feed-forward reconstruction backbone in the sparse-wide-baseline aerial regime. The end-to-end integration argument, the DepthSplat-OC architectural choice, the VN-Transformer adoption for the MambaSplat-4D spatial encoder, and the repurposing of 4D dynamics for classification are stated in §1.3 (Introduction) and the Methods chapter 5. This chapter focuses on the literature gaps and the selection rationale.

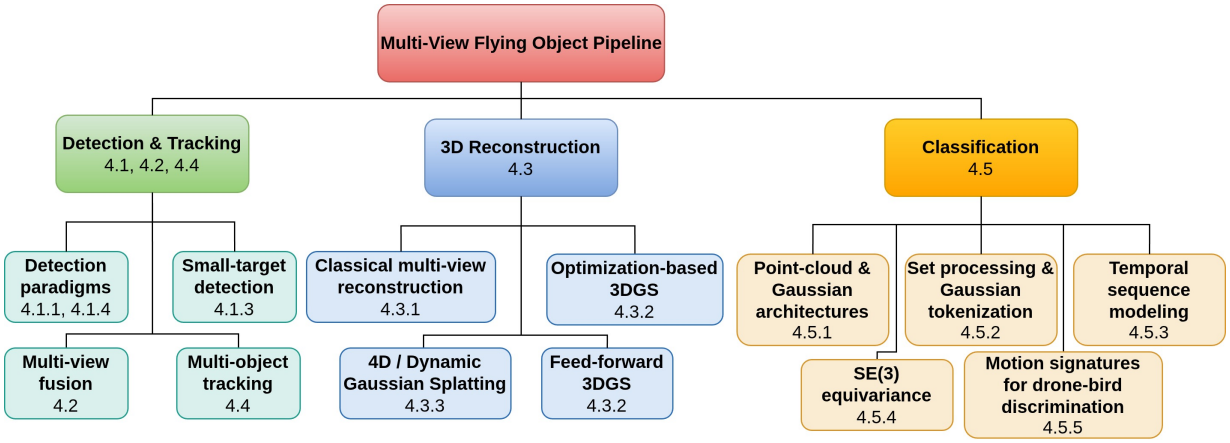


Figure 4.1: Overview of the literature components surveyed in this chapter and their mapping to the proposed pipeline.

4.1 Detection and Tracking of Flying Objects

4.1.1 Detection Paradigms: DBT vs. TBD

Two paradigms dominate aerial object detection, and the choice between them is driven by target resolution and the signal-to-clutter ratio (SCR), distinct from SNR, which compares signal to sensor noise (thermal, shot, and read noise combined). Detect-Before-Track (DBT) runs a per-frame detector and feeds confident detections into

a tracker; Track-Before-Detect (TBD) integrates raw unthresholded measurements across consecutive frames and decides on tracks rather than detections.

DBT dominates real-time applications. Convolutional and transformer detectors in the YOLO and RT-DETR (Real-Time DEtection TRansformer) families sustain 50–130 FPS on resolved targets [35], [36], [37], but the paradigm fails systematically as target size shrinks. Anchor-based designs commonly miss objects below $\sim 16 \times 16$ px due to stride and receptive-field constraints [38], [39]; CNN-based deep-feature down-sampling progressively reduces small-target spatial support to zero, eliminating any signal the detector could be trained on; and DBT degrades further once the background becomes cluttered.

TBD extends the operating envelope below conventional SNR thresholds at the cost of computational overhead. Multi-frame integration recovers targets where single-frame detection cannot: the foundational dynamic-programming TBD result of Barniv [40] established detection rates above 90 % at SCR $\approx$ 1.8 dB with false-alarm rates below 0.01 %, and modern analyses confirm the regime [41], [42]. Practical TBD pipelines integrate 5–20 frames to handle SCR < 3 dB [41], [43]; the price is complexity $O(N \times V \times K)$, which limits real-time deployment without dedicated parallelism. Both paradigms operate on single-view image sequences and do not incorporate multi-view 3D geometric constraints, which we discuss further in §4.1.5.

4.1.2 Background complexity dominates detection performance

The Drone-vs-Bird Challenge is the most comprehensive open benchmark for aerial flying-object detection [6], in its 8th edition at IJCNN 2025. The 2025 winner reached 73.7 % mAP using YOLOv11m with multi-scale processing and heavy augmentation [6]. Per-background-type breakdowns expose the paradigm's vulnerability: sky backgrounds reach a median AP of 51 % (consistent across runs); vegetation drops to a median of 26 % with a 0–89 % spread; buildings and mixed scenes fall below a median of 15 % [6]. Cluttered environments thus defeat appearance-based methods even when the detector is state of the art (see Fig. 4.2). **This motivates the foreground-segmentation stage that opens the pipeline in the Methods chapter 5:** by removing the background before reconstruction, the proposed system avoids the background-complexity penalty rather than addressing it within a 2D detector.

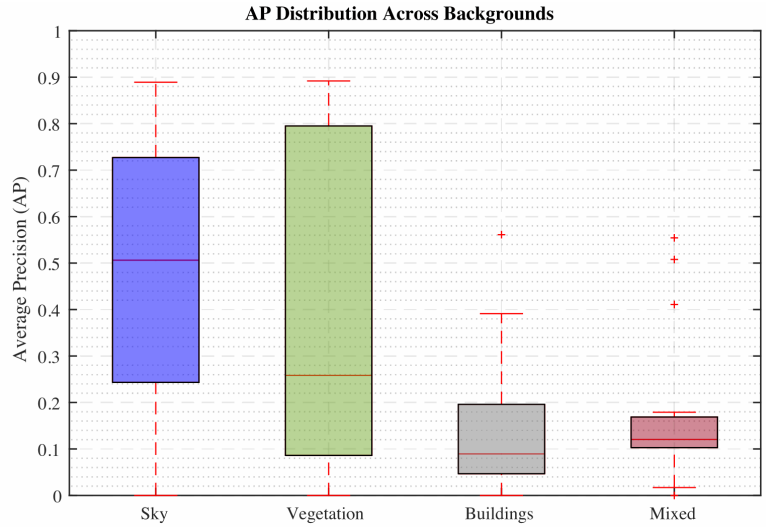


Figure 4.2: Box plot of AP distribution across background types [6].

4.1.3 Approaches to small target detection

Two families address the small-target failure mode of DBT. Super-resolution preprocessing enlarges images before detection: an end-to-end SR + detector pipeline improves recall by up to 32.4 % [44]. Spatiotemporal fusion extracts motion signatures from small targets at no resolution change [45]; in particular, optical-flow fusion attains 86.87 % AP (an absolute gain of 11.49 % over single-frame detection) while sustaining frame rates above 30 FPS [45]. Both SR and spatiotemporal fusion remain 2D-only: they extend range and add computational cost, yet they do not exploit cross-view geometric constraints, which is the gap §4.2 addresses.

4.1.4 Track-Before-Detect for weak signal conditions

TBD processes raw, unthresholded measurements integrated across consecutive frames and is therefore the method of choice in regimes where DBT fails [40], [42], [43]. Performance analyses across the dynamic-programming TBD family give the modern envelope: detection above 90 % at SCR $\approx$ 1.8 dB with false-alarm probability below 0.01 % [40], [42], and 5–20 frame integration for SCR $<$ 3 dB [41], [43]. The complexity $O(N \times V \times K)$ is the principal obstacle to real-time TBD on edge hardware without parallel implementations.

4.1.5 The multi-view 3D constraint gap

In the drone-vs-bird detection literature, the dominant DBT detectors such as YOLO, Faster R-CNN, and the DETR family are trained and evaluated on monocular imagery [6]. They carry no native multi-view consistency or epipolar constraints. Multi-view 3D detection certainly exists in adjacent domains: BEVFormer, BEVFusion, DETR3D, and related work address the ground-vehicle setting (§4.2), and MVDet established feature-perspective transformation as a multi-view-detection primitive [46], [47]. Yet no published aerial-target detector inherits those constraints, and the surveys that catalog motion-based multi-target tracking confirm that the DBT–MVS gap remains unaddressed for sparse-wide-baseline aerial capture [47].

Multi-view 3D reconstruction methods do exist in the aerial setting, but they treat detection as a downstream task. Bundle adjustment with flight-dynamics priors reconstructs 3D trajectories [48]; ad-hoc unsynchronized consumer-camera networks achieve centimeter-accurate trajectories via rolling-shutter correction [49]. DP-JAIT [50] leverages synchronized multi-camera capture with Vicon ground truth to refine 3D tracking, but does not address the reconstruction step itself. Multi-camera flying-object work with motion-capture ground truth has likewise been demonstrated [51], with centroid-distance metrics proposed for 3D tracking evaluation. At the high-modality end, MMAUD [52] and the CVPR 2024 UG2+ winner [53] combine stereo, LiDAR, radar, and audio sensing; the UG2+ winner specifically combined those modalities with dynamic-points analysis and trajectory completion. These are valuable contributions, but none target the setting of this thesis: sparse RGB-only wide-baseline cameras against a texture-less sky.

4.2 Multi-View Fusion Strategies

Fusion timing defines the geometric-consistency-versus-efficiency trade-off, and the literature partitions cleanly along this axis. Early fusion projects features into a shared 3D representation before any detector runs; late fusion runs per-view 2D detectors and aggregates downstream.

Early fusion: 3D-first representations. Early fusion ensures geometric consistency by construction: features land in a common volumetric space before any high-level reasoning, so downstream tasks operate on a single coherent representation. Lift-Splat-Shoot pioneered depth-based feature lifting [54]; BEVFormer adds spatiotemporal attention over multi-view Bird's-Eye-View features [55]; BEVFusion fuses camera and LiDAR features in shared BEV space [56]; RCBEVDet achieves real-time BEV inference [57]. Dense voxel grids become prohibitive beyond a few hundred meters. Moreover, BEV is designed for outward-looking automotive perception with a ground-plane prior, whereas the inward-looking volumetric capture in this thesis has neither.

Late fusion: 2D-first detection. Mature 2D detectors identify objects per view and triangulate the resulting boxes into 3D [50], [51]. By construction, per-view detection cost scales linearly with the number of cameras, which is attractive for engineering. The paradigm also benefits from the abundance of 2D training data and from rapid progress on monocular detectors. Two failure modes are decisive in the present regime, however. A missed detection in one view cannot be recovered, and because 2D detectors require a confidence threshold, false positives in any view can be triangulated into spurious 3D ghosts. Both modes are fatal for small distant targets against texture-less skies, where confidence margins are inherently small, and any threshold choice trades misses for ghosts.

Gap: BEV depends on ground-plane and texture priors; late fusion suffers irreversible 2D-detection errors. Two limitations remain unaddressed in the fusion literature surveyed above. BEV's design assumes outward-looking imagery against ground-plane texture; late fusion inherits the brittleness of its 2D front-end. The present thesis sidesteps both by reconstructing in 3D directly from sparse wide-baseline views, without an intermediate 2D-detection stage; the architectural choice is motivated in §1.3 and instantiated in the Methods chapter 5.

4.3 3D Reconstruction Methods

This section reviews how prior work reconstructs 3D structure from multiple images, from classical correspondence-based methods to feed-forward Gaussian Splatting and its 4D extension. The section surveys these three families to justify the DepthSplat-OC selection (§5.5.1) and surface the temporal gap §5.6.7 that DepthSplat-OC closes.

4.3.1 Classical Multi-View Reconstruction

Structure-from-Motion (SfM) and Multi-View Stereo (MVS) rely on local feature matching, as recalled in §3.1. Classical descriptors (SIFT, ORB) degenerate in texture-less regions [58], [59], and aerial sky is the worst case: initialization fails outright, and reconstructions remain empty. Space carving extends visual hulls without correspondence [28] but struggles with concavities and requires precise foreground segmentation; bundle adjustment exceeds the latency target of this thesis even on moderate scenes; and none of these methods amortizes cost across scenes.

4.3.2 3D Gaussian Splatting and Feed-Forward Reconstruction

3D Gaussian Splatting replaces implicit neural fields with explicit anisotropic primitives [15], and novel neural designs opened the door to feed-forward reconstruction at speeds compatible with downstream processing. Two systemic limitations of vanilla 3DGS frame the rest of this section: vanilla 3DGS optimizes per-scene against PSNR / LPIPS / SSIM until convergence. This is incompatible with real-time inference. Additionally, the SfM-initialized pipeline fails on texture-less regions [60]. Sky-dominated captures are therefore a degenerate case for SfM-initialized 3DGS; recent feed-forward methods such as VGGT mitigate this by removing sky regions, but this discards exactly the foreground-target geometry of interest in the aerial setting, which inverts the typical priority.

Feed-Forward Generalizable Architectures

Feed-forward architectures regress Gaussian parameters from posed images and are therefore the natural family for real-time aerial reconstruction. Four sub-families dominate the literature.

Pixel-aligned and single-view methods. Splatter Image is image-to-image with one Gaussian per pixel [61]; it achieves 38 FPS single-view reconstruction but generalizes poorly to out-of-distribution geometry. pixelSplat introduces an epipolar transformer over image pairs [62] but requires 80 GB VRAM (A100 / H100 class) at training time. Both lines rely on ShapeNet, CO3D, and RealEstate10K priors; correspondence signals vanish in texture-less regions [63], and single-view methods carry a structural risk of geometric hallucination that downstream classification cannot disentangle from genuine signal.

Cost-volume and dual-branch architectures. MVSplat constructs plane-sweep similarity volumes [64] and achieves 22 FPS at 10× fewer parameters than pixelSplat, reporting 26.39 PSNR on RealEstate10K with 12 M parameters. The authors target scene-level reconstruction [64]. GPS-Gaussian uses iterative disparity estimation with 3D correlation volumes [65], but targets dense, close-range human capture rather than sparse aerial capture. MVSplat itself fails on texture-less and reflective regions [63]. DepthSplat fuses a cost volume with monocular features from Depth-Anything-V2 [63], [66], reaching 27.47 PSNR on RealEstate10K (+1.08 dB over MVSplat) and retaining structure where stereo correspondence fails. **Critically for this thesis, DepthSplat supports a variable input view count N at inference [63].** This is the most important property that motivates its selection over the alternatives, instantiated in §5.5.1.

Transformer-based large reconstruction models. The Large Reconstruction Model (LRM) family regresses triplanes or per-pixel Gaussians directly from posed images [67], [68], [69], [70]. Inference is fast (GS-LRM reached 0.23 s on an A100, and LGM reached ∼5 s per object on four fixed views) [69], [70] but training costs are prohibitive in absolute terms: LGM trained on 32 A100 GPUs for four days; GS-LRM trained on 64 A100 GPUs for two days [69], [70]. The family is trained on Objaverse with unit-cube pre-scaling, fixed FOV (∼49.1° for LGM at radius 1.5) [69], and objects that fill the frame against blank backgrounds. Those preconditions are incompatible with sparse, wide-baseline aerial capture in which a target may subtend ten pixels of a texture-less sky.

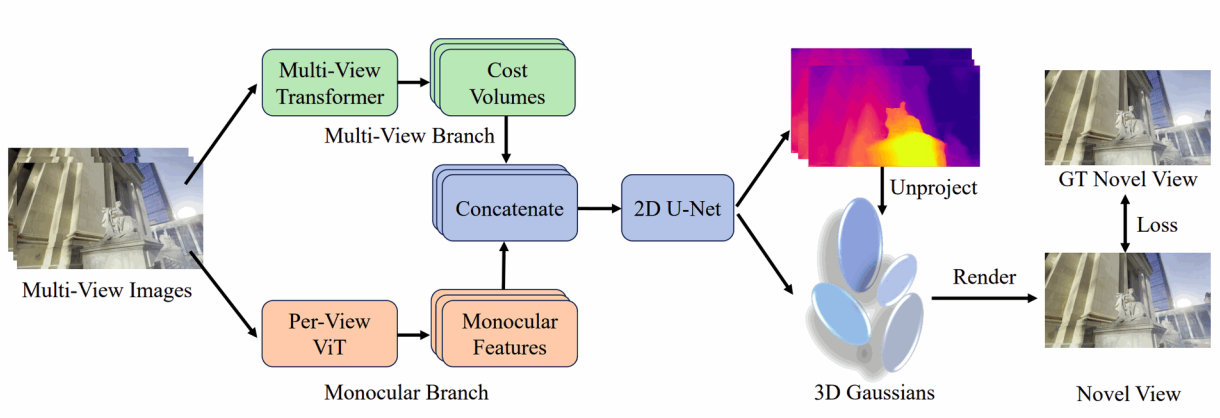


Figure 4.3: DepthSplat architecture overview [63].

Diffusion-based methods. Wonder3D learns cross-domain diffusion of normal and color maps [71]; Zero123++ diffuses six consistent novel views from a single image [72]. Diffusion synthesizes rather than measures geometry, which risks contaminating any downstream classification feature space with hallucinated structure that the classifier has no principled way to discount.

Positioning. Three properties separate DepthSplat from the rest of the feed-forward field for the purposes of this thesis. Vanilla 3DGS *can* be initialized with random points, but SfM initialization accelerates convergence and improves final quality, and random initialization is impractically slow for this thesis's real-time target [15]. Scene-trained feed-forward models degrade against texture-less sky [60], [63]. The LRM family is restricted to a fixed input view count. Among published open-source feed-forward families with variable N, DepthSplat best matches the sparse-wide-baseline regime; whether it suffices for texture-less distant small targets and symmetric quadcopter geometry is itself a research question, addressed empirically in the Methods and Experiments chapters 5.5.1 (Contribution 2).

4.3.3 Dynamic and 4D Gaussian Splatting

All of the optimization-based 4D Gaussian Splatting methods surveyed below are per-scene fits. 4DGS encodes a deformation field over a HexPlane encoder that predicts $(\Delta\mu, \Delta q, \Delta s)$ per Gaussian [73]; 4D-Rotor GS uses native XYZT primitives with geometric-algebra rotors and temporal slicing [74]; Hybrid 3D-4D GS demotes invariant Gaussians to a static branch [75]; 4DGS-1K reports 1000+ FPS rendering and a 41× storage reduction [76]. As a family, these methods set the dynamic-scene quality bar but cannot be used inside a real-time reconstruction pipeline because the per-scene optimization budget violates the latency requirement.

Across the feed-forward 4DGS line, reconstruction is per-timestamp with learned interpolation. L4GM regresses per-frame Gaussians in ∼1 s using temporal self-attention over monocular video, with a separate interpolation network densifying the sequence [77]; BTimer outputs a 3DGS snapshot in 150 ms and uses a learned novel-time enhancer to interpolate frames [78]. Both treat the 4D problem as reconstruction for novel views, not for classification.

Gap: existing 4D Gaussian methods ignore temporal motion as a downstream signal. Rigid drone dynamics and periodic wing-flap dynamics differ qualitatively, so the temporal channel that current 4D Gaussian methods discard is in fact informative for classification. The present thesis repurposes 4D Gaussian temporal evolution for discrimination rather than reconstruction quality (instantiated in §5.6.7; Contribution 3).

4.4 Multi-Object Tracking (MOT)

Multi-Object Tracking maintains identities across time, and tracking-by-detection dominates the literature. The canonical pipeline of SORT [79] pairs Kalman filtering with bipartite matching but fails under occlusion; DeepSORT fuses motion with appearance embeddings [80]; ByteTrack recovers low-confidence detections that other trackers discard [81]; OC-SORT mitigates occlusion via observation-centric momentum [82]. Lifting tracking to world coordinates is the natural multi-camera extension: MVTrajecter handles cross-view association [83], and

the WildTrack benchmark formalized multi-view tracking evaluation [84]. Set-prediction transformer trackers extend the DETR foundation [85]: TrackFormer treats tracking as set prediction [86], MOTRv2 extends the same idea [87]. In 3D, AB3DMOT applies Kalman filtering with 3D-IoU association on LiDAR [88]; CenterPoint introduces center-heatmap detection with velocity-based association on LiDAR [89]; PF-Track adds past-future query propagation for 3D tracking [90]; MUTR3D maintains 3D track queries shared across frames [91].

Gap: existing 3D MOT requires dense sensor coverage that sparse upward cameras cannot provide. Sparse upward-facing cameras preclude reliable appearance matching, and the LiDAR-centric tracking literature does not transfer to the RGB-only setting of this work. This thesis therefore defers identity-consistent 3D tracking to future work (§8.2) and treats classification per Gaussian-track snapshot. Accordingly, both the training and inference datasets place a single flying object in each captured scene, so multi-object capture, cross-view association, and identity-consistent tracking are deliberately out of scope here and form the immediate next step of this work (§8.2).

4.5 Classification from 3D Representations

4.5.1 Point Cloud Processing Architectures

3D Gaussian sets share the unstructured nature of point clouds, so the point-cloud architecture stack is the natural starting point. PointNet introduced per-point MLPs with symmetric aggregation [92]; PointNet++ added hierarchical local-pattern abstraction [93]; Point Transformer captured self-attention dependencies [94]; Point Transformer V3 is the serialized successor [95].

Classification on Gaussian representations. A small but growing line classifies *Gaussian* sets directly. ShapeSplat reconstructed ModelNet assets to 3DGS using vanilla 3DGS and learned to classify on the resulting appearance-plus-geometry representation [96]. Two recurring limitations persist: effective classification requires the full Gaussian parameter set (centroid proxies are insufficient), and existing architectures process only static Gaussian clouds. Gaussian-MAE adopts k-NN grouping that is not rotation-equivariant by construction; its classification performance derives from a learned Masked-Auto-Encoder latent space rather than from any geometric invariance [96]. The MAE-latent path achieves strong ModelNet numbers but does so by training a generative reconstruction objective whose latent space happens to be discriminative.

4.5.2 Set Processing and Gaussian Tokenization

Gaussian sets are unordered and of variable cardinality, which places them inside the set-learning literature. DeepSets is the canonical sum-pooled architecture [19]. Sum-pooling has limited expressivity for pairwise interactions: although DeepSets is a universal set-function approximator in the infinite-width limit, the finite-width sum-pooling that practical implementations use cannot represent arbitrary pairwise functions over the set elements [97], [98]. Set Transformer addressed the limitation directly by adding inducing-point self-attention, reducing complexity to $O(nm)$ [99]. Perceiver maps variable inputs to a fixed latent array via iterative cross-attention, enabling modality-agnostic processing [100].

Gap. Two related observations close the section. No established Gaussian-classification paradigm exploits temporal structure or the full Gaussian parameter set (appearance, opacity, and geometry jointly): ShapeSplat and Gaussian-MAE both rely on MAE latents and discard rotation-equivariance. Existing set architectures also ignore temporal evolution, classifying only static snapshots. The latter is the more important limitation and is the gap addressed in §5.6.7.

4.5.3 Temporal Sequence Modeling

Selective state-space models lead 3D and 4D point-cloud sequence modeling, reaching up to 92.6 % on ScanObjectNN-class benchmarks at linear $O(N)$ complexity versus transformer $O(N^2)$ that prohibits long sequences at edge memory budgets [20]. LSTMs suffer sequential bottlenecks and gradient instability over the frame counts of interest; TCNs are constrained by their fixed receptive fields; TimeSformer's factorized space-time attention [101] still creates prohibitive memory pressure at long horizons.

Mamba introduced input-dependent selective state spaces with a hardware-aware selective scan that reduces I/O cost to $O(N)$ [20]. Inference throughput is roughly 5× that of an equivalent transformer, and the architecture extrapolates to 1 M-token sequences. PointMamba serializes points along Hilbert curves before SSM

processing [102], and Mamba3D adds bidirectional SSM with Local Norm Pooling to reach 92.6 % on ScanObjectNN [103]. Mamba4D disentangles spatial and temporal processing, pairing DGCNN-style spatial encoders with a Mamba temporal stage, and reports +10.4 % over P4Transformer on long sequences with 87.5 % less GPU memory [104]. The compute envelope is concrete: P4Transformer uses ball-query spatio-temporal tubes at 44.10 M parameters and 35.54 GFLOPs (Table 6.19), while Mamba4D itself runs at 109.77 M parameters and 304.74 GFLOPs. Constant-state memory $O(D \cdot N)$ in selective SSM, which Mamba4D inherits from base Mamba, is what makes the family viable for edge deployment.

4.5.4 SE(3) Equivariance for 3D Data

Equivariance with respect to the Special Euclidean group SE(3), rotations and translations in 3D, bakes symmetry directly into the architecture. Standard augmentation provides no theoretical guarantees and forces the network to learn the symmetry from data, whereas equivariant layers share weights across rotations by construction.

The trade-off: expressivity versus efficiency. Tensor Field Networks introduced spherical-harmonic filters [18], the SE(3) Transformer extends the construction with attention over spherical harmonics [33], and Equiformer reaches state-of-the-art on molecular tasks via nonlinear message passing [105]. The cost is brutal: tensor products scale as $O(L^6)$ or $O(L^3)$ depending on filter order, making these models 10–100× slower than scalar networks and unsuitable for edge deployment at the operating envelope of this thesis. Vector Neurons offer a cheaper alternative by lifting scalar operations to 3D-vector operations with small overhead [29], and the VN-Transformer extends this with rotation-equivariant Frobenius attention, reaching 90.8 % on ModelNet40 with only ∼40 k parameters [106]. VN-MeanProject reduces the attention cost from $O(N^2C)$ to $O(M^2C')$, and the architecture supports early fusion of non-spatial attributes by extending $C \times 3$ to $C \times (3 + d_A)$ for opacity and color. This is what makes VN-Transformer a viable *spatial* encoder for Gaussian classification, adopted in §5.6.3 (Contribution 3); the same $C \times (3 + d_A)$ shape leaves room to fuse 3D tracking signals or external triangulation (microphones, radio signal) as future work (§8.2).

Gap: rotation equivariance is virtually unexplored for drone-bird detection. Current systems rely on CNNs or transformers that need massive labeled data and rotation augmentation to *approximate* the equivariance VN-Transformer guarantees by construction.

4.5.5 Motion Signatures for Drone-Bird Discrimination

Radar micro-Doppler established motion as a primary discriminator: drone rotors operate above 50 Hz while bird wing-flaps oscillate at 4–6 Hz [107]. Molchanov demonstrated 92 % classification accuracy from eigenpair micro-Doppler features [108]; Rahman pushed this to 99 % via a CNN over micro-Doppler spectrograms [109]. Vision-based methods have historically been dominated by appearance, but a recent line exploits temporal context: Akyon's LSTM/Transformer fusion improved bird F1 by 73 % over single-frame baselines [110], and the optical-flow + spatiotemporal fusion of Sun [45] is the natural 2D analog.

Gap and opportunity. Standard video frame rates resolve the wing-flap band (4–10 Hz) but miss the rotor band (> 50 Hz), and no prior work extracts motion signatures from 4D Gaussian representations. At 30 FPS, consumer video undersamples the rotor band, so visual systems must distinguish drones by the *absence* of avian motion rather than by the rotor signal directly. More importantly for this thesis, *no prior work extracts motion signatures from 4D Gaussian representations for drone-bird discrimination*: the literature handles motion either in 2D video [45], [110] or in radar micro-Doppler [107], [108], [109], never in 4D Gaussian temporal dynamics. MambaSplat-4D closes that gap (Contribution 3; temporal Mamba encoder in §5.6.7).

4.6 Chapter Summary and Research Gaps

Three pillars structured the survey: detection (paradigms, small-target methods, multi-view-3D gap), 3D reconstruction (classical, 3DGS, feed-forward, 4D dynamic), and temporal classification (point-cloud architectures, set processing, sequence modeling, SE(3) equivariance, motion signatures). Across all three, methods have been optimized either for close-range texture-rich capture or for single-view 2D imagery. The intersection that defines this thesis, namely sparse wide-baseline capture, texture-less sky, and small, spatially symmetric, dynamic targets (quadcopters and airplanes) covering very few pixels per view, exposes a coherent set of gaps that no single existing system addresses.

Table 4.1 maps each surveyed gap onto a thesis contribution.

Table 4.1: Summary of identified research gaps and thesis contributions.

Identified Research Gap	**Thesis Contribution**
Data. No public multi-camera aerial dataset combines class-balanced flying-object instances (bird, drone, helicopter, airplane) with synchronized wide-baseline capture and complete 3D ground truth (RGB, masks, depth, 3D trajectories, camera poses).	Contribution 1; full description in §1.3 and Stage 0 of the Methods chapter 5.
Reconstruction. Feed-forward Gaussian Splatting assumes textured scenes and bounded objects; it fails on sky backgrounds and wide baselines [60], [63].	Contribution 2; DepthSplat-OC architecture in §5.5.1.
Classification. No existing paradigm classifies 4D Gaussian sequences; point-cloud methods discard appearance and opacity; static-Gaussian methods discard motion.	Contribution 3; VN-Transformer spatial encoder in §5.6.3.
Dynamics. Temporal evolution of Gaussian parameters (deformation, rotation) is unexploited for discrimination. Prior literature establishes (i) multi-frame benefit for fine-grained classification via evidence accumulation [111], [112] and (ii) motion-signature discriminability for drone vs. bird via radar micro-Doppler and 2D-video baselines [107], [108], [109], [110]; this thesis extends the latter to feed-forward 4D Gaussian representations via MambaSplat-4D.	Contribution 3; temporal Mamba encoder in §5.6.7.

The Methods chapter 5 synthesizes these gaps into an end-to-end pipeline.

5 Methodology

This chapter presents the full method behind `AeroSplat-4D`: a four-stage pipeline that renders synthetic multi-view aerial scenes, consumes the simulator's ground-truth foreground masks, reconstructs each frame as a set of 3D Gaussians, and classifies the resulting temporal sequence under SO(3) invariance provable up to a bounded ε bias term. After the Gaussian-primitive notation (§5.1) and the system overview (§5.2), each stage is described in turn: synthetic dataset generation (§5.3), foreground masks from simulation (§5.4), feed-forward 3DGS reconstruction (§5.5), `MambaSplat-4D` rotation-invariant classification (§5.6: the core academic contribution, including the SO(3) equivariance formalism, feature-mode notation, and rotation-evaluation protocols), and end-to-end integration (§5.7). An RGB-only multi-view foreground segmentation module was prototyped during the project and is preserved as future work in Appendix C.

5.1 Preliminaries and Notation

This section defines the notation used throughout the chapter: how a single 3D Gaussian is parameterized and what it means for the classifier to be rotation-invariant. Readers fluent in 3D Gaussian Splatting can skim it; the only thesis-specific choices are the scalar-first quaternion convention and keeping color as a single view-independent value.

We adopt the standard 14-parameter Gaussian primitive of [15], defined in §3.2 as $\theta_i = \{\boldsymbol{\mu}, \mathbf{s}, \mathbf{q}, \alpha, \mathbf{c}^{\mathrm{SH}}\}$ with covariance $\boldsymbol{\Sigma} = \mathbf{R}_{\mathbf{q}} \operatorname{diag}(\mathbf{s}^2) \mathbf{R}_{\mathbf{q}}^{\top}$. Three thesis-specific conventions apply throughout. The quaternion convention is scalar-first (w, x, y, z). The higher-order spherical-harmonic bands are dropped, and the classifier uses the view-independent DC color $\mathbf{c}_{\mathrm{dc}} \in \mathbb{R}^3$ only. Together, the stored 14 parameters carry 13 continuous degrees of freedom; the unit-quaternion constraint accounts for the missing one, and a frame of N Gaussians is laid out as a scene tensor of shape $(B{\cdot}T, N, 14)$ in its flat form, or equivalently a key-wise dictionary in its structured form.

The pipeline enforces SO(3) invariance as a structural property of the classifier architecture. The formal equivariance and invariance definitions, the Gaussian rotation rule, the feature-mode vocabulary, and the rotation-evaluation protocols are developed in Stage 3 (§5.6) alongside the classifier that realizes them.

5.2 System Overview

`AeroSplat-4D` classifies a flying object (bird, drone, helicopter, or airplane) from synchronized multi-camera video alone, with no radar or active sensor: it reconstructs the target in 3D from several views, then classifies how that 3D shape moves over time. The full system is a four-stage pipeline: synthetic rendering, foreground-mask provision, feed-forward 3D reconstruction, and rotation-invariant temporal classification. Each stage exposes a typed interface, allowing downstream stages to swap implementations without touching upstream code.

The pipeline (Fig. 1.1) decomposes into four named stages. **Stage 0** generates synthetic multi-camera RGB, instance masks, and camera poses in NVIDIA Isaac Sim. **Stage 1** reads the simulator's ground-truth masks directly, replacing the RGB-only segmenter that the future-work appendix prototypes. **Stage 2** is the feed-forward 3DGS reconstruction module, `DepthSplat-OC` in the deployed configuration, with LGM retained as a

baseline. **Stage 3** is `MambaSplat-4D`, the rotation-invariant classifier that consumes the per-frame Gaussian sequences and emits a class label. Stages 0–2 are engineering contributions that enable end-to-end evaluation; Stage 3 is the academic contribution; **Stage 4** composes the four stages and benchmarks the full pipeline against 2D appearance-based baselines.

The full pipeline takes synchronized sparse multi-camera video as input and emits a single per-sequence class label as output. An orchestrator script chains rendering, segmentation, reconstruction, and classification in turn. Training is targeted at an RTX 5090, and edge deployment is targeted at the NVIDIA Jetson AGX Orin.

Three design principles run through the pipeline. We decouple geometry from semantics: Stage 2 reconstructs, and Stage 3 classifies, with no architectural cross-talk between the two. We enforce SO(3) invariance at the classifier rather than through data augmentation, which means rotation robustness is a structural property of the architecture and not a property of the training distribution. Finally, the per-frame latency budget drives the choice of a linear-time state-space model for the temporal encoder, as motivated in §3.3.

5.3 Stage 0 – Synthetic Dataset Generation

The pipeline requires densely annotated multi-view aerial video, which is not publicly available at the required scale and class balance. Stage 0 generates it synthetically in NVIDIA Isaac Sim, rendering each flying object from multiple calibrated cameras simultaneously, with exact ground truth (masks, depth, 3D trajectories, and calibration) at no labeling cost. The result is the `AeroSplat-4D` dataset that feeds every downstream stage.

5.3.1 Rendering Framework

Synthetic multi-camera video is generated with NVIDIA Isaac Sim [5], driven by Omni Replicator and YAML scenario configurations that are deep-merged from a shared base with per-scenario overrides. A headless batch orchestrator renders scenarios sequentially within a single Isaac Sim process, using NVIDIA's RTX renderer in ray-traced-lighting mode with 16-sub-frame accumulation and Replicator's per-step orchestration for production frames. Physics is disabled throughout; object kinematics is driven solely by USD keyframes. Scenario parameters such as flight direction and tree placement are deterministically seeded, so the rendered trajectories are reproducible at the parameter level, although the RTX integrator itself is not pixel-deterministic.

5.3.2 Synthetic Data and Annotated Simulators

Densely annotated multi-view aerial datasets are a recurring bottleneck for both 3D reconstruction and 4D classification. Sub-pixel ground truth from real multi-camera rigs is prohibitively expensive to collect at scale, and the public aerial datasets that do exist lack the scale, class balance, and full 3D truth required by the four-stage pipeline.

Simulator-based pipelines. A handful of established simulators address adjacent domains. CARLA [113] is a photorealistic driving simulator; AirSim [114] targets aerial robotics; Habitat [115] simulates embodied indoor agents; Kubric [116] is a procedural multi-modal synthetic-data generator; and BlenderProc [117] is a deterministic annotated-rendering pipeline built on Blender. NVIDIA Isaac Sim, together with Omniverse Replicator, complements this set with RTX path tracing and USD scene composition, the combination that the present work exploits.

Annotation modalities. A simulator-driven pipeline yields deterministic ground truth at no labeling cost. The exported modalities cover instance and semantic segmentation, depth, normals, optical flow, 2D and 3D bounding boxes, and 6-DoF object poses with exact intrinsics and extrinsics. All modalities are time-locked across the cameras of a rig, which is the property that lifts the rendered output from a synthetic dataset to a synthetic *multi-view* dataset.

Domain randomization and sim-to-real. Following the domain-randomization rationale of [118], [119], the pipeline varies two modalities at scenario-generation time: a sun light whose elevation is keyframed across the capture interval, producing time-of-day variation in two of the three scenario families; and flight-pose jitter, which uniformly perturbs azimuth and elevation to diversify viewpoints. Texture and material randomization are not yet implemented; extending DR coverage to these modalities is left to future work (Section 8.2).

Gap and positioning. No public dataset combines wide-baseline multi-camera animated flying-object captures with class-balanced data across bird, drone, helicopter, and airplane, and provides a full annotation suite at scale that supports feed-forward 3DGS reconstruction followed by 4D classification. To the best of our knowledge, no such dataset exists. This thesis contributes `AeroSplat-4D`, an Isaac Sim-based pipeline with $N \geq 3$ calibrated rigs, to close that gap.

5.3.3 Asset Library and Ground Truth

The asset library is class-organized into per-class USD libraries with skeletal animation, covering four flyer classes: bird, drone, helicopter, and airplane. Figure 5.1 previews the 30 bird assets, spanning realistic and stylized models in varied flight poses. USD skeletal animation drives the wing-flap kinematics of the bird assets, while drone, helicopter, and airplane motion is driven by USD keyframe transforms. Every rendered frame exports RGB, instance segmentation, and camera intrinsics and extrinsics; per-frame 3D position, 2D bounding box, and a visibility flag are written to the metadata. Per-object 6-DoF trajectories are also exported per frame, supporting the trajectory-based analyses introduced in Chapter 1. A depth pass can be enabled optionally; it is disabled by default to preserve throughput.

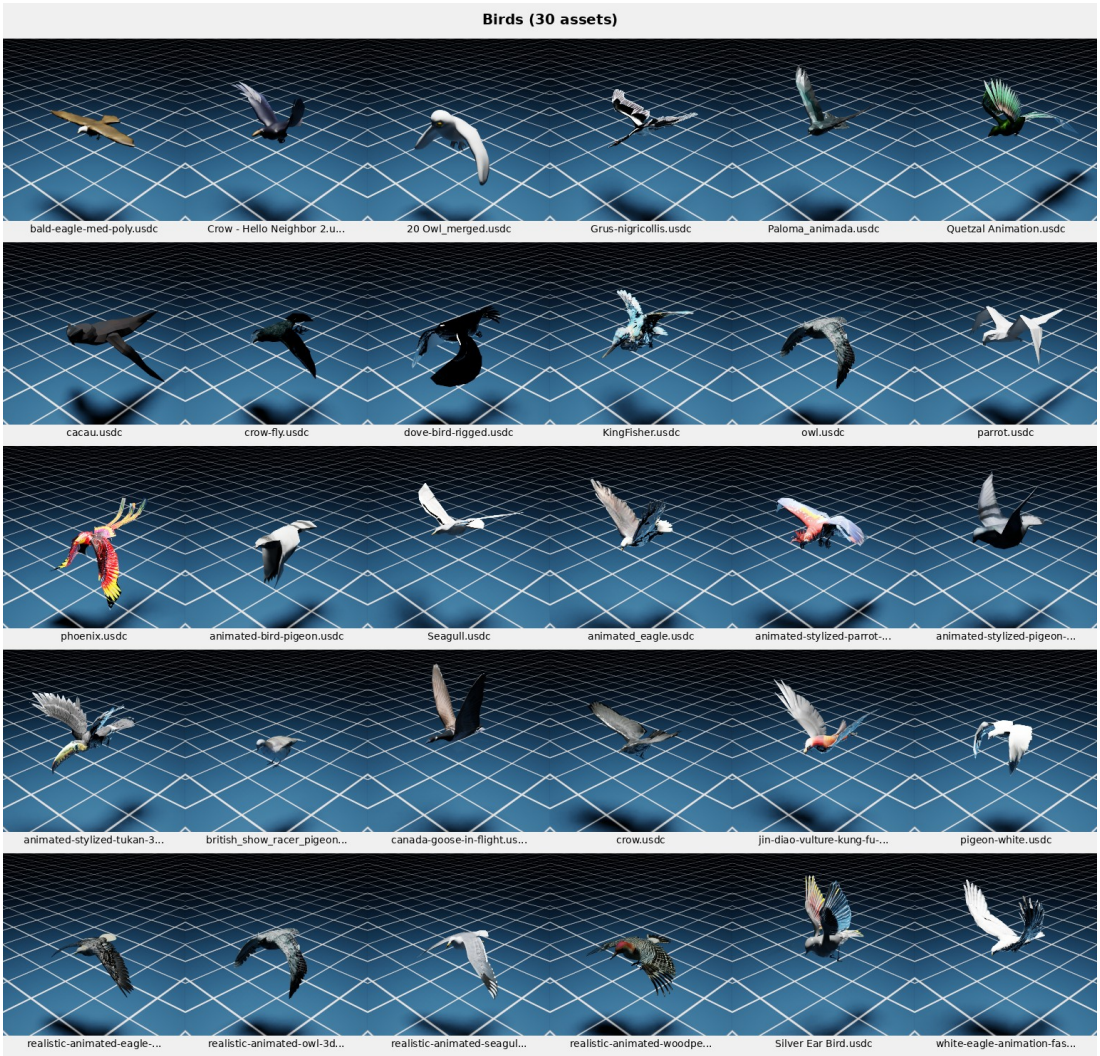


Figure 5.1: Asset library (bird preview). The full asset library is shown in Appendix B.

5.3.4 AeroSplat-4D Subsets

`AeroSplat-4D` comprises three camera-rig subsets, each isolating a different aspect of aerial sensing. All three carry the four-class label set (bird, drone, helicopter, airplane) and are reconstructed with `DepthSplat-OC` (§5.5.1). Sim-1 uses four calibrated cameras while Sim-2 and Sim-3 each use five; the rig geometry, scene

context, role, and split of each subset are summarized in Table 5.1, and Figure 5.2 shows synchronized frames from all three rigs.

Table 5.1: `AeroSplat-4D` subsets. All three are reconstructed with `DepthSplat-OC`. The "Role" column indicates whether the subset is split for training (in-distribution) or rendered only for OOD evaluation.

ID	Name	Camera Rig	Scene / Bg	Role	Split	GT
Sim-1	Distance-Sweep	Sphere, 4 cams, 10 radii (2–280 m)	White (no scene)	Train/Val/Test (ID)	50/25/25 id-disjoint (46/21/24)	Full†
Sim-2	Outdoor Circle	Circle, 5 cams @ 100 m radius, 37 m height	Rivermark, trees, sky	Train/Val/Test (ID)	50/25/25 id-disjoint	Full†
Sim-3	Line-Formation OOD	Line, 5 cams @ 50 m baseline, 8 m height	Rivermark, trees, sky, shadow	Test only (OOD)	24 held-out (8/8/4/4)	Full†

† Full = RGB, depth, instance masks, 3D trajectories, camera intrinsics and extrinsics, and ground-truth 3DGS parameters. `DepthSplat-OC` is our object-centric DepthSplat variant (§5.5.1); real-camera evaluation is reserved as future work (§8.2).

Sim-1: Distance-Sweep (in-distribution). Sim-1 isolates pure scene geometry against a white dome background. A spherical 4-camera rig samples ten radii from 2 to 280 m, the lower portion of the pipeline's target-distance envelope, making it the primary training dataset and the basis for the Stage-2 distance-degradation test. Object identities are split 50/25/25 into train, validation, and test partitions (46, 21, and 24 identities respectively), with no identity appearing in more than one partition.

Sim-2: Outdoor Circle (in-distribution). Sim-2 adds realism diversity by placing the target in an outdoor environment (scene downloaded from Nvidia named Rivermark) with an animated sun. The 5 cameras sit on a circle at 100 m radius and 37 m height, with a monitored-volume radius of 25–50 m as constrained in Chapter 1. Sim-2 follows the same identity-disjoint 50/25/25 split as Sim-1, since the asset library is shared across rigs.

Sim-3: Line-Formation (out-of-distribution). Sim-3 places a 5-camera rig along a 50 m linear baseline at 8 m height in an outdoor scene with cast shadows. It is rendered for held-out evaluation only; no training or validation partitions exist. The 24 test identities (8 birds, 8 drones, 4 helicopters, 4 airplanes) are the same as the held-out test set from Sim-1 and Sim-2 partitions, probing generalization to an unseen camera rig and background geometry.

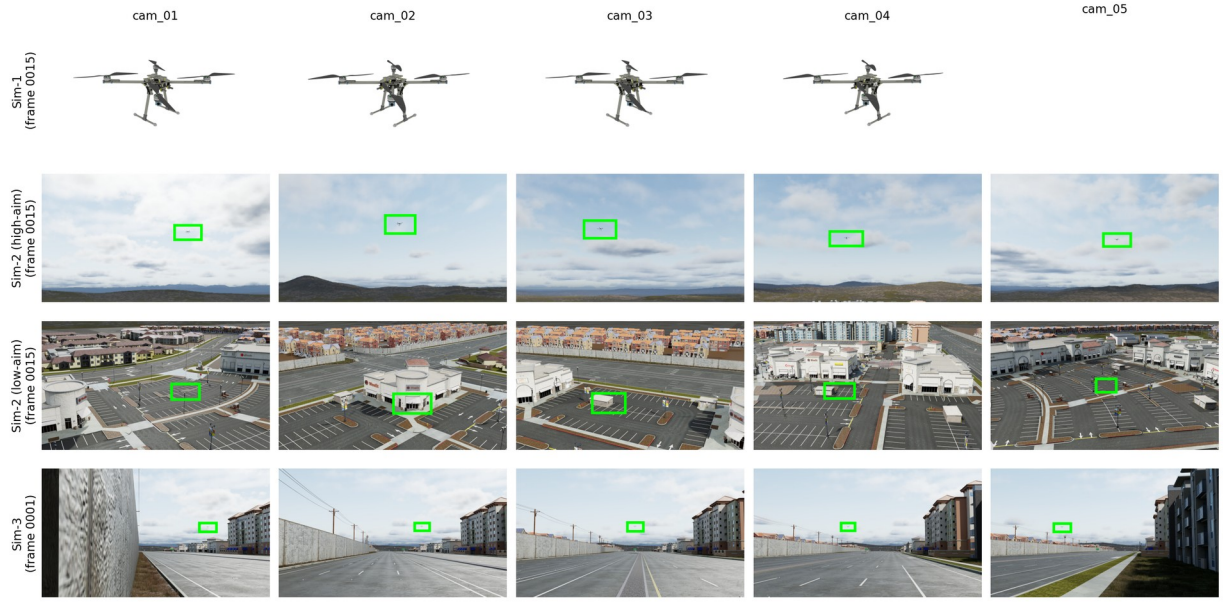


Figure 5.2: Multi-camera capture setup for `Sim-1`, `Sim-2`, and `Sim-3`. Synchronized cameras observe the scene from different viewpoints; the quadrotor is visible across multiple frames.

Stage usage. Stage 3 trains on the Sim-2 train and val partitions. In-distribution evaluation reports Sim-1-test and Sim-2-test separately, and Sim-3 acts as the out-of-distribution probe. Stage 2 uses Sim-1 as a distance-stratified reconstruction test set (§5.5.2), while Stage 3 uses all three subsets for classification (§5.6.9). The Stage-3 architectural ablations are run on the Sim-1 distance-sweep subset; the end-to-end head-to-head comparison (§6.6.1) evaluates all three regimes.

5.4 Stage 1 – Foreground Masks from Simulation

Before reconstruction, the target must be separated from the sky and background in each view; that is what a foreground mask provides. Stage 1 supplies per-camera 2D foreground masks to Stage 2. In this thesis, the masks are read directly from the Isaac Sim instance-segmentation annotator produced in Stage 0; no learned RGB-only segmentation is performed at inference time. The design of an RGB-only multi-view ray-march segmentation module that would close this gap was prototyped during the project and is preserved as future work in Appendix C.

5.4.1 Mask Source and Format

The masks are produced by Isaac Sim's instance-segmentation annotator, configured during the Stage 0 rendering pass. Two complementary signals are fused per pixel: a hard instance-ID mask that assigns each pixel to a specific object, and a sub-pixel alpha-coverage channel that captures partial occupancy at silhouette edges. The fusion unions both signals into a single binary foreground mask, stored as an 8-bit grayscale PNG per camera per frame at the native render resolution of the corresponding rig (Table 5.1). When the target leaves a camera's frustum, a per-frame visibility flag records its absence.

5.4.2 Downstream Use

Stage 2 consumes the binary masks in two ways. First, photometric reconstruction losses are gated on foreground pixels, so that background sky does not corrupt the Gaussian geometry. Second, where the reconstruction adapter supports it, the mask is forwarded as an additional input to a mask-conditioned encoder, allowing the network to attend selectively to the target object.

5.4.3 Rationale and Limitations

Reading the masks directly from the simulator decouples segmentation from the reconstruction–classification core. The noiseless training signal sets a clean upper bound on the downstream performance the pipeline can reach; in particular, mask consistency across views supervises the Gaussian geometry in the visual-hull style developed in §3.1.3. The convention is consistent with established synthetic-data practice: Kubric [116], BlenderProc [117], and Omniverse Replicator pipelines all rely on ground-truth instance masks rather than a learned segmenter at training time. The limitation is structural rather than incidental: a real deployment cannot read masks from the simulator, and a complete pipeline therefore requires an RGB-only foreground-extraction front-end. A candidate design that exploits temporal motion cues, multi-camera voxel-voting consensus ($\geq k$ agreeing views), and vote back-projection is documented in Appendix C and listed in §8.2.

5.5 Stage 2 – Feed-Forward 3DGS Reconstruction

This stage turns the masked multi-camera views of a single frame into a 3D model of the target, represented as a set of 3D Gaussians, in a single forward pass rather than the slow per-scene optimization of standard 3DGS; doing so in a single pass makes a real-time pipeline possible. `DepthSplat-OC`, the deployed reconstructor, adapts an existing feed-forward method to the hard regime of this task: small, distant objects seen against textureless sky from wide-baseline cameras.

5.5.1 DepthSplat-OC: Object-Centric Adaptation

Off-the-shelf feed-forward reconstructors are trained on textured, room-scale scenes and break down on a small object floating against blank sky; `DepthSplat-OC` is our object-centric adaptation that addresses this. `DepthSplat-OC` adapts DepthSplat [63] from scene-level to object-centric reconstruction through four coupled modifications: a mask-aware cost volume, sigmoid-bounded scales, a Dense Prediction Transformer (DPT) monocular residual, and a virtual-camera crop. Together, the four address the failure modes of the pre-trained

DepthSplat on textureless-sky foregrounds. These four coupled modifications constitute the second engineering contribution of the thesis (Contribution 2).

`DepthSplat-OC` is the deployed reconstructor for Stages 3 and 4 on the three `AeroSplat-4D` simulation subsets (Sim-1, Sim-2, and Sim-3), where the four (Sim-1) or five (Sim-2 and Sim-3) fixed rig cameras of each subset serve as context views. Its encoder parameter count is 38.12 M (encoder only; the rasterizer is excluded), as reported in Table 6.19. Published numbers for LGM [69] and other feed-forward reconstructors are included in Table 6.6 as literature baselines (RQ2).

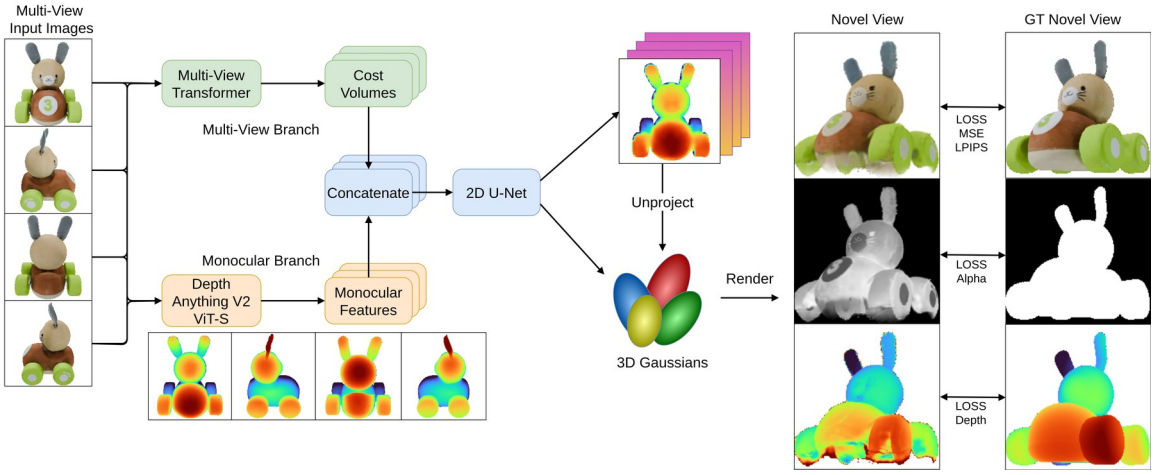


Figure 5.3: Illustration of `DepthSplat-OC` modifications. Modified from [63]

`DepthSplat-OC` is designed for sparse wide-baseline rigs and textureless regions as motivated in Chapter 1, and it is trained on the curated Objaverse subset whose geometry matches the downstream capture configuration (§4.3.2). The base architecture inherits DepthSplat's `MultiViewUniMatch` cost volume [120] plus a DPT-style dense-prediction upsampler [121]. The monocular branch is a Depth-Anything-V2 ViT-S backbone [66] fine-tuned end-to-end at a learning rate of 10^{-5}. The mask-aware cost volume zeroes out cost in background pixels and forces those pixels to the far plane. The final depth is the sum of the multi-view coarse depth and the DPT monocular residual, and the per-pixel Gaussian head emits opacity, offset, scale, rotation, and the SH-DC color.

Depth bounds are tightened from DepthSplat's scene-level range (0.1–1000 m) to the object-level range [0.55, 2.54] m used throughout training, and the view-sampling distribution is changed from sequential frames to a 360° Fibonacci sphere. The scales are bounded by a sigmoid, following the precedent set by GRM [122], which avoids the dead-zone failure mode of an unbounded scale parameterization at low opacities. These modifications give the adapter explicit support for object-centric reconstructions, which DepthSplat's scene-level pre-training does not natively support. The sigmoid bound itself is given by the equation below; the rescaled sigmoid maps the unconstrained raw scale $\mathbf{x}$ into a bounded magnitude:

$$\mathbf{s} = \sigma_{\max} \cdot \mathrm{sigmoid}(\mathbf{x}) + \sigma_{\min} \tag{5.1}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are the lower and upper bounds on the Gaussian scale magnitude.

Virtual-camera crop-and-upscale. At the operating ranges of this thesis (the 2–280 m Sim-1 sweep), aerial targets subtend only 8–60 px in the full frame, far below DepthSplat's training scale. For each frame, we tight-crop a 1.2× bounding-box window around the detection, resample it to 256 × 256 with Lanczos filtering, and override the intrinsics to the training-matched 50° field of view. To preserve multi-view epipolar consistency under off-center crops, a Rodrigues rotation aligns the crop-center ray with the optical axis, while the camera translation is preserved unchanged. The crop is driven entirely by the 2D detection bounding box and the camera intrinsics; no metric depth or distance measurements are required at inference time. The net effect is a virtual camera that views the target as if up to ∼85× closer (Fig. 5.4), which brings distant objects back into the regime `DepthSplat-OC` was trained on.

The near and far depth bounds 0.55 m to 2.54 m are derived empirically from the Objaverse training set as the 5th and 95th percentiles of per-scene minimum and maximum depths across 118,552 scenes with valid

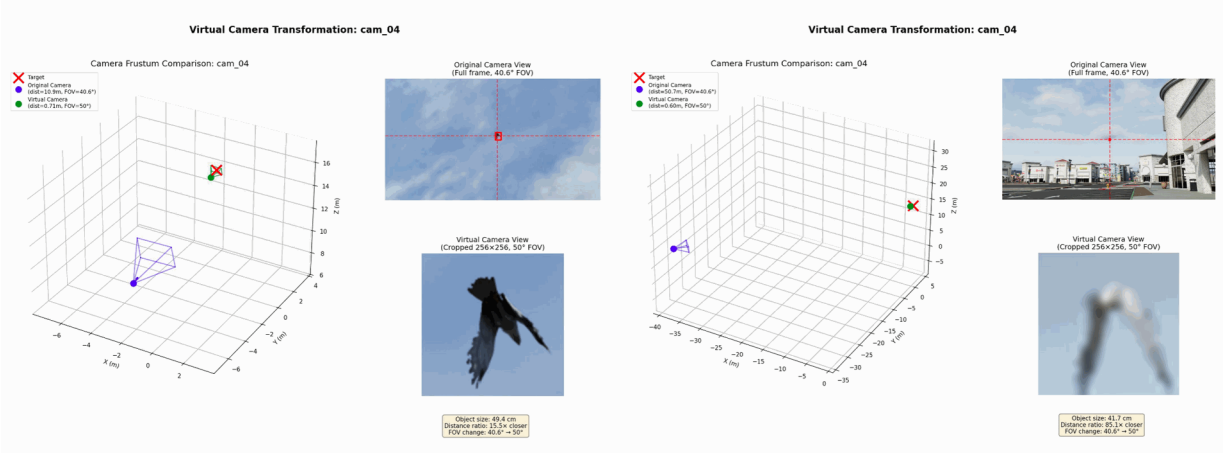


Figure 5.4: Virtual-camera crop-and-upscale on a representative camera. **Left:** near target at 10.9 m (object size 49.4 cm, 15.5× closer virtual distance). **Right:** far target at 50.7 m (object size 41.7 cm, 85.1× closer). In each case the original frame (top) is tight-cropped and Lanczos-resampled to 256 × 256 (bottom); the virtual camera is rotated so the crop-center ray aligns with the optical axis, lifting the effective field of view from 40.6° to the training-matched 50°. The metric distances and object sizes are ground-truth annotations shown for illustration only; the crop itself uses only the 2D bounding box and the camera intrinsics and requires no distance measurements.

rendered depth, drawn from the 122,506 curated training objects, expanded by a 10 % safety margin; because these object-scale bounds are fixed at inference time, they do not transfer directly to aerial 10 m to 100 m ranges.

5.5.2 Datasets

Stage 2 reconstruction draws on three datasets. The training set is a curated subset of Objaverse rendered in-house. Two cross-domain evaluation datasets, Google Scanned Objects [123] and Amazon Berkeley Objects [124], probe sim-to-real generalization. The in-domain evaluation set is the `AeroSplat-4D` Sim-1 distance-sweep subset (Table 6.8). The roles, sources, scales, and render protocols of the three datasets are consolidated in Table 5.2.

Training dataset: Objaverse. `DepthSplat-OC` is trained on a curated subset of Objaverse [125], filtered through the Objaverse++ quality annotations of [126]. Lin et al. show that training on a quality-filtered subset of Objaverse yields stronger 3D reconstruction than using the full corpus, as noisy, degenerate, or overly simplistic assets degrade learned geometry priors; local workstation storage constraints further reinforced this choice. The filtering retains objects with a quality score ≥ 2, high triangle density, and none of the degenerate flags (multi-object, scene-level, transparent, or single-color), yielding 122,506 objects from the 789,195 Objaverse++ candidates, a roughly 15% retention rate that preserves full-geometry, opaque, single-object meshes while keeping both realistic and scanned assets.

Objects are rendered in Blender 5.1 with three objects processed concurrently via a multiprocessing pool. Per object we render 32 views at 256×256 with white-composited RGB; cameras are sampled with azimuth $\mathcal{U}[0, 2\pi)$ and elevation $\mathcal{U}[-45°, +45°]$, the vertical field of view is fixed at 50°, and the camera radius is the per-asset optimal-distance solution that frames the object at the training scale, under four randomized SUN lights. Each view stores an RGB image, a binary foreground mask, metric depth, and the camera record (normalized intrinsics and the 3×4 world-to-camera matrix). The renders are split 90/10 into training and test partitions; the model is trained for 300k steps on the training partition.

Object-level evaluation: GSO and ABO. The checkpoint is evaluated on two held-out benchmarks, both disjoint from Objaverse. Google Scanned Objects [123] contributes the 1,046 objects in the local GSO mirror used for Stage-2 evaluation, and Amazon Berkeley Objects [124] contributes a 1,000-object subset of the product-imagery release. Both are scanned or captured from real objects, and both probe sim-to-real generalization with no overlap with the training set. The evaluation rig follows the LGM-style protocol of [69] with 10 Fibonacci target views and 4 cardinal context views at 20° elevation, rendered at 256×256 with white-

composited RGB. The camera poses are fixed across all evaluation runs, and the same harness drives all Stage-2 adapters (§5.5.4).

In-domain evaluation: AeroSplat-4D Sim-1 distance-sweep. The Sim-1 subset described in §5.3.4 and Table 5.1 is used at Stage 2 as a distance-stratified reconstruction probe. The sphere rig carries 4 cameras at the 10 radii from 2 to 280 m and isolates the controlled small-object regime that is not covered by GSO or ABO. The distance-degradation study (§6.4.3) sweeps the rig radius over the ten values $\{2, 4, 8, 16, 32, 64, 100, 140, 200, 280\}$ m and reconstructs each scene at the native 256^2 resolution.

Table 5.2: Stage-2 reconstruction datasets. The #Objects column lists curated training objects for Objaverse (each rendered as one 32-view scene) and physical objects for GSO and ABO. The `AeroSplat-4D` Sim-1 details are given in Table 5.1.

Dataset	Role	Source	#Objects	Render protocol
Objaverse (curated)	Train	Objaverse v1 + Objaverse++ filter	122,506	32 views, 256^2, white BG; 90/10 train/test; union of the score≥2 high-density, top-50k score-3, and high-quality subsets
GSO [123]	Eval (real)	Google Scanned Objects	1,046	10 Fibonacci targets + 4 cardinal contexts @ 20°, 256^2; disjoint from Objaverse
ABO [124]	Eval (real)	Amazon Berkeley Objects	1,000	Same rig as GSO; product-imagery domain
AeroSplat-4D Sim-1	Eval (in-domain)	Isaac Sim → DepthSplat-OC	see Tab. 5.1	Sphere rig, 4 cams, 10 radii (2–280 m), 256^2; distance-degradation probe

Per-dataset render galleries, RGB, mask, normals where rendered, and depth, for the Objaverse training set and the GSO, ABO, and Sim-1 evaluation sets are collected in Appendix A.1 (Figures A.1–A.4).

5.5.3 Training Objective (Objaverse Training)

`DepthSplat-OC` is trained on the curated Objaverse subset described above. The loss combines four terms: full-image RGB, LPIPS, mask, and depth, following the LGM and GRM convention of [69], [122]. The mask term replaces an earlier foreground-only loss that produced floating Gaussians; direct alpha supervision is more accurate and robust than the heuristic alpha estimation used by the legacy silhouette loss.

Training-time view sampling. The object-centric sampler draws 7 of the 32 rendered views per scene via farthest-point sampling, partitioned into 5 context views and 2 target views. The depth bounds are data-driven (near 0.55, far 2.54), matching the rendered objects' radius range.

Loss. The full-image MSE and LPIPS [127] terms are computed on the white-composited RGB; the mask term is an MSE on the rendered alpha against the ground-truth silhouette, in the LGM/GRM style; and the depth term is an L1 on foreground pixels with a background-to-far regularizer carrying an internal weight of 0.1. The composite objective is:

$$\mathcal{L}_{\text{DS-OC}} = \mathcal{L}_{\text{MSE}} + 0.05\,\mathcal{L}_{\text{LPIPS}} + 0.1\,\mathcal{L}_{\text{mask}} + 0.5\,\mathcal{L}_{\text{depth}} \tag{5.2}$$

where $\mathcal{L}_{\text{MSE}}$ is the full-image pixel MSE on white-composited RGB, $\mathcal{L}_{\text{LPIPS}}$ is the learned perceptual image-patch similarity, $\mathcal{L}_{\text{mask}}$ is the MSE between the rendered alpha and the ground-truth silhouette, and $\mathcal{L}_{\text{depth}}$ is the L1 depth loss with an internal background regularizer weighted at 0.1 for the background-to-far term.

Optimization. The training recipe adapts DepthSplat's [63] default configuration; architectural widths and batch size are halved to fit a single RTX 5090 (32 GB), while learning rates and the multi-view neighborhood are re-tuned for the object-centric setting. Optimization uses AdamW [128] with a main learning rate of 1.5×10^{-4} (original: 2×10^{-4}) and a separate 1×10^{-5} rate for the Depth-Anything-V2 ViT-S backbone [66] (original: 2×10^{-6}). The learning-rate schedule is a one-cycle policy with a 1% warm-up fraction (∼3,000 warm-up steps over the 300,000-step run) and a cosine anneal, with weight decay 0.01 and gradient clipping at 0.5. The batch size is 2 scenes per step (reduced from 4–14 for memory) over 300,000 steps in bf16-mixed precision, with UNet

gradient checkpointing and the CUDA rasterizer running as a float32 island. The cost-volume UNet and the Gaussian head are trained from scratch; only the monocular ViT-S is initialized from Depth-Anything-V2. The encoder hyperparameters are 64 depth candidates (halved from 128 for memory), a cost-volume UNet feature dimension of 64 (halved from 128), a multi-view matching neighborhood of 5 (increased from 2 to improve coverage on the wide-baseline Fibonacci rig), an SH degree of 2, one Gaussian per pixel, and sigmoid-bounded Gaussian point scales in $[10^{-6}, 10^{-2}]$.

5.5.4 Evaluation Harness

A three-step evaluation pipeline (rendering (adapted from [125], [129]), tensor serialization (adapted from [63]), and metric computation (following [31], [127])) is shared by all adapters; the datasets are those listed in §5.5.2 and Table 5.2. The protocol uses 10 Fibonacci target views per object and 4 cardinal context views at 20° elevation following the LGM convention of [69] (the LGM original input is 4 cardinal views at 0° elevation; the 20° lift is `AeroSplat-4D`-specific to improve spatial coverage of horizontal structures such as airplane wings and multirotor arms). The metrics are PSNR, SSIM [31], and LPIPS [127]; depth and normal metrics are reported where the modalities are available. The distance sweep places the camera rig at each of the 10 radii $\{2, 4, 8, 16, 32, 64, 100, 140, 200, 280\}$ m and evaluates reconstruction at the native 256^2 resolution, as developed in §6.4.3.

5.5.5 Rasterizer

The rasterizer is the `diff-gaussian-rasterization` module shipped with LGM [69], which exposes depth and alpha channels in both forward and backward passes to support mask supervision; we reuse it unchanged for `DepthSplat-OC`. A finite-value guard protects means, covariances, SH coefficients, opacities, and extrinsics, while extreme covariances and opacities are clamped to prevent tile-sort overflow.

5.5.6 Stage-3 Bridge: Batch Reconstruction

A batch reconstruction step converts each Isaac Sim video directory into per-frame Gaussian tensors that Stage 3 can consume directly. The step iterates over the four classes (bird, drone, airplane, helicopter) and the frames of each clip. The per-frame output is a serialized tensor consumed directly by the Stage-3 data loader.

5.6 Stage 3 – MambaSplat-4D Rotation-Invariant Classifier

Stage 3 is the core contribution of the thesis. At long range a flying object is too small for its appearance to be reliable, so `MambaSplat-4D` classifies it by how its reconstructed 3D shape moves over time rather than by how it looks in any single view, and the predicted class does not change when the whole scene is rotated: SO(3) invariance is built into the architecture rather than learned from augmented data. `MambaSplat-4D` maps a temporal sequence of 3D Gaussian Splats to a class label through a four-block pipeline that is SO(3)-invariant, provable up to a bounded ε bias term. All operations before the VN-In bridge are equivariant; the bridge is the sole invariance boundary; everything after is scalar.

Pre-pooling. Every frame is decimated offline to $N = 1024$ Gaussians before it reaches the classifier; the three-step decimation chain and its stochastic-subsampling scheme are detailed in §5.6.10. Because the token count is fixed, all spatial mixing is global self-attention over the 1024 tokens; no runtime farthest-point sampling, k-nearest-neighbor grouping, or hierarchical pooling is performed.

5.6.1 Design goals

Four design goals shape the classifier. We exploit all 14 Gaussian attributes rather than xyz alone, so that opacity, scale, quaternion, and DC color all contribute to discrimination. SO(3) invariance is provable up to the ε bias term, a structural property of the architecture, not robustness inherited from data augmentation. Per-frame invariance is intentional: 4DGS reconstruction noise and timing jitter can dominate sub-frame motion such as a bird's wing-beat, so we trade exact delta-T modeling for robustness to reconstruction-induced ordering noise. The temporal encoder is linear-time in T on variable-cardinality frames, and the full pipeline trains end-to-end.

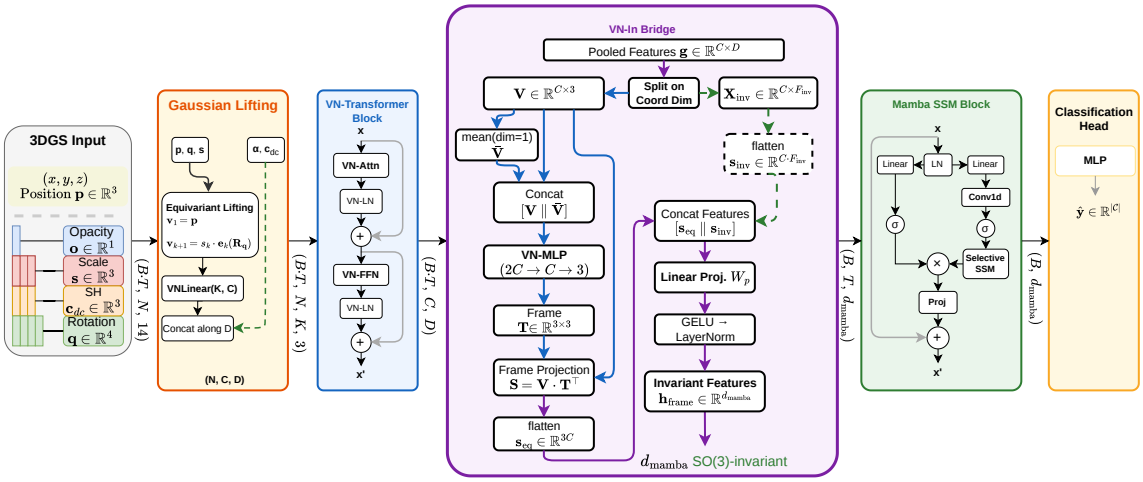


Figure 5.5: **`MambaSplat-4D` pipeline.** Each Gaussian is lifted into 4 equivariant vectors, processed by a VN-Transformer, pooled per frame, projected to invariant scalars via the VN-In Bridge, and fed to a Mamba SSM for temporal encoding.

5.6.2 Rotation convention

The formal definitions of equivariance (Eq. 3.13) and invariance (Eq. 3.14), the Gaussian rotation rule (Eq. 3.5), and the rotation-invariance of the Frobenius inner product are established in §3.3.2; we reuse them throughout this chapter with the notational shorthands $\mathbf{p}$ for the Gaussian center $\boldsymbol{\mu}$ and $\mathbf{c}_{\mathrm{dc}}$ for the DC color. Because the VN feature tensors have shape $\mathbf{V} \in \mathbb{R}^{C\times 3}$ with each channel stored as a *row* 3-vector, the per-vector rule $\boldsymbol{\mu} \mapsto \mathbf{R}\boldsymbol{\mu}$ acts on the coordinate axis by right-multiplication with the transpose, $\mathbf{V} \mapsto \mathbf{V}\mathbf{R}^\top$ (each row obeys $\mathbf{v}_c^\top \mapsto (\mathbf{R}\mathbf{v}_c)^\top = \mathbf{v}_c^\top \mathbf{R}^\top$). This is the batched form of the column-vector rule used in the lifting (Eq. 5.5). Haar-uniform rotations on SO(3) are drawn through the PyTorch3D random-rotations utility, which samples them from unit quaternions with Gaussian-distributed coordinates rather than from a uniformly drawn rotation angle [130], and is applied jointly to $(\mathbf{p}, \mathbf{q})$ per Eq. 3.5.

5.6.3 Preliminaries: Vector Neurons

Vector Neurons (VN) are the building block behind the classifier's rotation handling: every feature is a 3D vector that rotates together with the input instead of a plain number, which makes the network rotation-equivariant. Full rotation invariance is produced downstream, at the VN-In bridge. The core VN operators are a faithful port of the upstream VN-Transformer [29], [106]; our only additions are a VN-LeakyReLU activation and a mask-aware centroid. Features carry the shape $\mathbf{V} \in \mathbb{R}^{N\times C\times 3}$ in which each channel is a 3-vector. **VNLinear** mixes channels only and leaves the spatial axis untouched. **VN-ReLU** is the upstream activation: it learns vectors $(\mathbf{q}, \mathbf{k})$ and projects $\mathbf{q}$ perpendicular to $\mathbf{k}$ whenever $\langle \mathbf{q}, \mathbf{k}\rangle < 0$. We add **VN-LeakyReLU**(α) as $\mathbf{q} - (1-\alpha)(\mathbf{q}\cdot\hat{\mathbf{k}})\hat{\mathbf{k}}$ with $\alpha = 0.2$, used in the bridge only. **VN-LayerNorm** unit-normalizes the vectors, pipes the norms through a standard LayerNorm, and re-multiplies. A quasi-equivariant bias of magnitude $\varepsilon \approx 10^{-6}$ is included for accelerator stability, as specified by the linear-with-bias variant of VNLinear:

$$\mathbf{y}_{o,c} = \sum_i \mathbf{W}_{oi}\, \mathbf{x}_{i,c} + \varepsilon\, \hat{\mathbf{b}}_o \tag{5.3}$$

where $\mathbf{W}_{oi}$ is the learned channel-mixing weight, $\mathbf{x}_{i,c}$ is the input vector at input channel i and spatial token c, $\hat{\mathbf{b}}_o$ is a unit-norm learned bias direction, and ε is the small bias magnitude.

The Frobenius inner product is the foundation of rotation-invariant attention scores:

$$\langle \mathbf{V}, \mathbf{W}\rangle_F \;=\; \sum_{c,k} \mathbf{V}_{c,k}\, \mathbf{W}_{c,k} \;=\; \mathrm{tr}(\mathbf{V}^\top \mathbf{W}) \tag{5.4}$$

where the sum runs over the channel index c and the coordinate index k. Rotation invariance follows directly from $\langle \mathbf{V}\mathbf{R}^\top, \mathbf{W}\mathbf{R}^\top\rangle_F = \mathrm{tr}(\mathbf{R}\,\mathbf{V}^\top\mathbf{W}\,\mathbf{R}^\top) = \mathrm{tr}(\mathbf{V}^\top\mathbf{W}\,\mathbf{R}^\top\mathbf{R}) = \mathrm{tr}(\mathbf{V}^\top\mathbf{W})$, using the cyclic property of the trace and $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$, as established in §3.3.2.

5.6.4 Gaussian Lifting

Gaussian Lifting is the entry point of the classifier: it turns each 3D Gaussian into a small bundle of co-rotating 3D vectors plus rotation-independent scalars (opacity, color), so the spatial encoder that follows can process them equivariantly. The ablation studies vary which Gaussian attributes feed the classifier. We adopt the naming convention of [96], augmented with two intermediate modes that isolate the opacity and color contributions independently of the quaternion-driven axis weighting. Single letters denote individual attributes: C for the center, O for opacity, SH for the SH-DC color, S for scale, and R for the rotation quaternion, and a *mode* selects a subset of these attributes. Scale is equivariant whenever a quaternion is present (because the scale modulates the orientation axes) and invariant otherwise. Six modes are used throughout the thesis to isolate position, opacity, color, scale, and quaternion contributions in turn (see Table 5.3); for each mode, K denotes the number of co-rotating 3-vectors per Gaussian, d_{inv} the number of appended invariant scalars, and $D = 3 + d_{\text{inv}}$ the per-channel coordinate dimension.

Mode	Keys used	K	d_{inv}	D	Paper name
`C`	$\mathbf{p}$	1	0	3	$E(C)$
`C_O`	$\mathbf{p}, \alpha$	1	1	4	$E(C, O)$
`C_SH`	$\mathbf{p}, \mathbf{c}_{\text{dc}}$	1	3	6	$E(C, \text{SH})$
`C_S_R`	$\mathbf{p}, \mathbf{s}, \mathbf{q}$	4	0	3	$E(C, S, R)$
`C_O_S_R`	$\mathbf{p}, \alpha, \mathbf{s}, \mathbf{q}$	4	1	4	$E(O, C, S, R)$
`X`	all 14	4	4	7	$E(\text{X})$

Table 5.3: Six feature modes. K is the lifting-vector count, d_{inv} the injected-scalar count, D the per-channel coordinate dimension. Scale is equivariant when a quaternion is present (axis weighting) and invariant otherwise.

Gaussian Lifting converts each Gaussian into an equivariant vector bundle plus appended invariant scalars. The per-frame centers are mean-centered under a mask-aware centroid, $\mathbf{p}' = \mathbf{p} - \bar{\mathbf{p}}$. From each Gaussian we build $K \in \{1, 4\}$ co-rotating 3-vectors, with the choice of K set by the feature mode (Table 5.3). The scale modulates the magnitude of these vectors, while the quaternion orients the principal axes. Under an external rotation, every vector $\mathbf{v}_k$ transforms as $\mathbf{v}_k \mapsto \mathbf{R}\mathbf{v}_k$, so the lifting is equivariant; the scale scalars s_k are themselves rotation-invariant, and the lifted vectors $s_k\,\mathbf{R}_{\mathbf{q}}[:, k]$ are therefore rotation-equivariant. Invariant scalars, opacity α and DC color $\mathbf{c}_{\text{dc}}$, are appended along the coordinate dimension, giving an output of shape (N, C, D) with $D = 3 + d_{\text{inv}}$ and $d_{\text{inv}} \in \{0, 1, 3, 4\}$. The lifting itself is given by the equation below:

$$\mathbf{v}_1 = \mathbf{p}', \qquad \mathbf{v}_{k+1} = s_k\,\mathbf{R}_{\mathbf{q}}[:, k], \qquad k \in \{1, 2, 3\} \tag{5.5}$$

where $\mathbf{p}'$ is the centered Gaussian center, s_k is the scalar scale along axis k, and $\mathbf{R}_{\mathbf{q}}[:, k]$ is the k-th column of the rotation matrix derived from the quaternion $\mathbf{q}$.

The invariant scalars are then injected along the coordinate axis:

$$\mathbf{x} \leftarrow \left[\mathbf{x} \parallel \text{expand}([\alpha, \mathbf{c}_{\text{dc}}])\right] \in \mathbb{R}^{N \times C \times D}, \quad D = 3 + d_{\text{inv}} \tag{5.6}$$

where the bracket notation denotes concatenation along the coordinate dimension and expand broadcasts the invariant scalars across the C channels. Six feature modes are then realized by selecting different subsets of the lifted features per Table 5.3 above.

5.6.5 VN-Transformer Spatial Encoder

The spatial encoder mixes information across the Gaussians within a single frame and emits one rotation-equivariant descriptor per frame. It is a standard VN-Transformer: attention whose scores use a rotation-invariant inner product, so the descriptor rotates with the scene yet never depends on a chosen orientation.

The spatial encoder is a stack of four VN-Transformer blocks with $C = 14$ channels, $h = 4$ heads, and per-head dimension $d_h = 8$. Each block uses post-norm residual connections, attention followed by VN-FFN, each wrapped in VN-LayerNorm:

$$\mathbf{x} \leftarrow \mathrm{VNLN}\big(\mathrm{VNAttn}(\mathbf{x})\big) + \mathbf{x} \tag{5.7}$$
$$\mathbf{x} \leftarrow \mathrm{VNLN}\big(\mathrm{VNFFN}(\mathbf{x})\big) + \mathbf{x} \tag{5.8}$$

where VNLN, VNAttn, and VNFFN denote VN-LayerNorm, VN attention, and the VN feed-forward network. The attention scores themselves use the Frobenius inner product, which is rotation-invariant by construction:

$$a_{ij} \;=\; \frac{1}{\sqrt{d_h D}} \sum_{c=1}^{d_h} \sum_{k=1}^{D} Q_{i,c,k}\, K_{j,c,k} \tag{5.9}$$

where $Q_{i,c,k}$ and $K_{j,c,k}$ are the query and key entries at token i (resp. j), channel c, and coordinate k; the prefactor $1/\sqrt{d_h D}$ provides the standard temperature scaling. Multi-head reshaping keeps the heads as a separate tensor axis; within each head, the channel and coordinate dimensions are flattened to $(d_h \cdot D)$ for the softmax, which is then taken per query token. FlashAttention [131] is used on GPUs with compute capability sm ≥ 80, with a math-attention fallback whenever the padding mask is active. The feed-forward block follows the upstream VN-FFN form of [106]: VNLinear$(C, 4C) \to$ VN-ReLU$\to$ VNLinear$(4C, C)$. No positional encoding is added, since position enters through the lifted centroid $\mathbf{v}_1$. A masked weighted mean-pool followed by a learnable $\mathbf{W}_{\text{pool}} \in \mathbb{R}^{C\times C}$ remix produces the per-frame equivariant global descriptor of shape (B, C, D).

5.6.6 VN-In Bridge

This is the single step where the network stops tracking orientation, converting the per-frame descriptor into numbers that are unchanged by any rotation of the scene. The VN-In bridge is the sole invariance boundary of the pipeline: it projects pooled equivariant vectors onto a learned equivariant frame to produce rotation-invariant scalars. The pooled tensor is split into its equivariant and invariant parts along the coordinate dimension by the spatial encoder before reaching the bridge:

$$\mathbf{V} = \mathbf{g}_{:,:,0:3} \in \mathbb{R}^{C\times 3}, \qquad \mathbf{x}_{\text{inv}} = \mathbf{g}_{:,:,3:D} \in \mathbb{R}^{C\times d_{\text{inv}}} \tag{5.10}$$

where $\mathbf{V}$ collects the first three coordinate slots (the equivariant 3-vector part), and $\mathbf{x}_{\text{inv}}$ collects the remaining slots (the appended invariant scalars). A learned frame $\mathbf{T} \in \mathbb{R}^{3\times 3}$ is then generated from the equivariant features by a three-layer VN-MLP with VN-LeakyReLU at $\alpha = 0.2$; the final `VNLinear` emits three channels, so the frame is a triple of learned 3-vectors:

$$\mathbf{T} \;=\; \mathrm{VNLin}_{C\to 3}\big(\mathrm{LReLU}_{0.2}\big(\mathrm{VNLin}_{C\to C}\big(\mathrm{LReLU}_{0.2}\big(\mathrm{VNLin}_{2C\to C}([\mathbf{V}\|\bar{\mathbf{V}}])\big)\big)\big)\big) \tag{5.11}$$

where $\bar{\mathbf{V}}$ is the channel-mean of $\mathbf{V}$ and the bracket denotes concatenation along the channel dimension. The frame is then used to project the equivariant features into rotation-invariant scalars:

$$\mathbf{S} \;=\; \mathbf{V}\,\mathbf{T}^{\top} \;\in\; \mathbb{R}^{C\times 3} \tag{5.12}$$

where $\mathbf{S}$ is the resulting invariant scalar matrix. The invariance follows from the algebraic identity

$$(\mathbf{V}\mathbf{R}^{\top})(\mathbf{T}\mathbf{R}^{\top})^{\top} \;=\; \mathbf{V}\mathbf{R}^{\top}\mathbf{R}\,\mathbf{T}^{\top} \;=\; \mathbf{V}\mathbf{T}^{\top} \tag{5.13}$$

which holds *without* any orthonormality, non-degeneracy, or full-rank assumption on $\mathbf{T}$. The invariant scalars are then combined with the invariant features and projected to the Mamba hidden dimension $d_{\text{mamba}} = 256$ through a Linear–GELU–LayerNorm head:

$$\mathbf{h}_{\text{frame}} \;=\; \mathrm{LN}\big(\mathrm{GELU}\big(\mathbf{W}_p\,[\mathrm{flat}(\mathbf{S}) \,\|\, \mathrm{flat}(\mathbf{x}_{\text{inv}})] + \mathbf{b}_p\big)\big) \;\in\; \mathbb{R}^{d_{\text{mamba}}} \tag{5.14}$$

where $\mathbf{W}_p$ and $\mathbf{b}_p$ are the projection weights and bias, flat$(\cdot)$ vectorizes along the channel axis, and the bracket denotes concatenation. Alternative bridges, a vector-norm pool, a Gram-matrix invariant, a pairwise-distance invariant, and the VN-StdFeature head are ablated in §5.6.11 and Table 6.12.

5.6.7 Mamba Temporal Encoder

With each frame reduced to one rotation-invariant descriptor, the temporal encoder reads the sequence of those descriptors and summarizes how the object moves. It uses a Mamba state-space model rather than a Transformer because Mamba's cost grows linearly in the number of frames and lets the encoder process frames as they arrive, in line with the pipeline's per-frame latency budget.

Per-frame descriptors $\mathbf{h}_{\text{frame}}^{(t)} \in \mathbb{R}^{d_{\text{mamba}}}$ form a length-T sequence consumed by a unidirectional Mamba selective state-space model [20]. The linear-time $\mathcal{O}(T \cdot d)$ scaling replaces the quadratic attention $\mathcal{O}(T^2 \cdot d)$ of a standard Transformer [17] over the same sequence.

The bridge output is the per-frame input to the encoder; no extra projection layer is inserted. A sinusoidal positional encoding is added with a maximum sequence length of 512. The stack carries four Mamba blocks with pre-norm residuals and an intra-block dropout of 0.2 on `AeroSplat-4D` and 0.1 on MSR-Action3D. Dropout is the primary guard against overfitting on the small supervised training sets, acting alongside weight decay and (on `AeroSplat-4D`) label smoothing. The default hidden dimension is $d_{\text{mamba}} = 256$ and is swept over $\{64, 128, 256, 512\}$ in the Experiments ablation §6.5.6; the default depth (adapted from [104]) is $n_{\text{layers}} = 4$ and is swept over $\{1, 2, 4, 8\}$ in the depth ablation §6.5.7. The implementation uses the unidirectional selective SSM; the bidirectional variant (*bimamba*, which scans the sequence both forward and backward) is deliberately not used, so that frames can be processed causally as they arrive rather than requiring the full clip, in line with the streaming per-frame latency budget. Each Mamba block applies the standard residual structure:

$$\mathbf{r}^{(\ell)} = \text{DropPath}(\mathbf{h}^{(\ell-1)}) + \mathbf{r}^{(\ell-1)}, \quad \mathbf{h}^{(\ell)} = \text{Mamba}\big(\text{LN}(\mathbf{r}^{(\ell)})\big) \tag{5.15}$$

where $\mathbf{r}^{(\ell)}$ is the residual stream at layer ℓ, $\mathbf{h}^{(\ell)}$ is the post-Mamba hidden state, and LN is LayerNorm. The selective SSM kernel itself follows the standard SSM equation (Eq. 3.22) from the Background chapter: the parameters $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are input-dependent and $\mathbf{D}$ is held constant.

A final LayerNorm is followed by a mean-pool across the T frames, producing an output of shape (B, d_{mamba}). The mean-pool is preferred over a last-token aggregator so the gradient is spread across all frames, as ablated in Figures 6.11 and 6.12. The classification head is a small MLP, $\text{Linear}(256, 128) \to \text{ReLU} \to \text{Dropout} \to \text{Linear}(128, |\mathcal{C}|)$, with $|\mathcal{C}| = 4$ on `AeroSplat-4D`. Because all temporal operations act on scalars, the temporal stage is trivially SO(3)-invariant. The temporal pooling and the head are summarized by:

$$\mathbf{z} = \tfrac{1}{T} \sum_{t=1}^{T} \mathbf{h}_t^{(L)}, \qquad \hat{\mathbf{y}} = \text{MLP}(\mathbf{z}) \tag{5.16}$$

where $\mathbf{h}_t^{(L)}$ is the layer-L hidden state at frame t, $\mathbf{z}$ is the mean-pooled clip representation, and $\hat{\mathbf{y}}$ is the classifier logits.

Complexity. The temporal encoder runs in $\mathcal{O}(T \cdot d)$ versus the Transformer's $\mathcal{O}(T^2 \cdot d)$ [17], with constant memory per step.

5.6.8 Full-Pipeline Invariance

This subsection provides the formal guarantee behind the architecture: it shows that rotating the entire input leaves the output exactly unchanged in ideal arithmetic and changes it by at most a tiny, bounded ε on real hardware, so the predicted class does not flip. The full pipeline is SO(3)-invariant, provable up to a bounded ε bias term [106]. In this subsection, partition the pipeline into three parts: Part A (Gaussian Lifting through per-frame pooling), Part B (the VN-In bridge), and Part C (Mamba and the MLP head). Because every `VNLinear` acts on $C \times 3$ vector features, all bias-bearing operators lie in A $\cup$ B; no operator in Part C carries an ε contribution. The partition is therefore clean.

Exact case ($\varepsilon = 0$). With bias-free `VNLinear` in the [29] style, Part A is a composition of exactly SO(3)-equivariant maps: Gaussian Lifting ($\mathbf{v}_k \mapsto \mathbf{R}\mathbf{v}_k$ per the lifting equation above), Frobenius-score attention (scores invariant, values equivariant), and the VN-weighted pooling. Part B applies the algebraic identity $(\mathbf{V}\mathbf{R}^\top)(\mathbf{T}\mathbf{R}^\top)^\top = \mathbf{V}\mathbf{T}^\top$ already given in the Bridge equation, converting equivariance into invariance without

any orthonormality, non-degeneracy, or full-rank assumption on $\mathbf{T}$.[1] Part C acts on scalars. Hence $f(\mathbf{R} \circ \mathcal{G}^T) = f(\mathcal{G}^T)$ for all $\mathbf{R} \in \mathrm{SO}(3)$, exactly, in exact arithmetic:

$$f\big(\mathbf{R} \circ \mathcal{G}^T\big) \;=\; f\big(\mathcal{G}^T\big) \qquad \forall\, \mathbf{R} \in \mathrm{SO}(3) \tag{5.17}$$

where $\mathcal{G}^T$ denotes the temporal sequence of T frames of Gaussians and $\mathbf{R} \circ \mathcal{G}^T$ denotes the joint rotation of every Gaussian's $(\mathbf{p}, \mathbf{q})$ in every frame.

Approximate case ($\varepsilon > 0$). The trained checkpoints use `VN-LinearWithBias` of [106] with $\varepsilon = 10^{-6}$ in every Part-A `VNLinear`, adopted for accelerator stability; the bridge's frame-generating VN-MLP uses the bias-free variant ($\varepsilon = 0$), so no additional ε contribution enters there. Defining the violation metric $\Delta(f, \mathcal{G}, \mathbf{R}) = \|f(\mathbf{R} \circ \mathcal{G}) - f(\mathcal{G})\|_F$, each bias-bearing layer ℓ with C'_ℓ output channels satisfies $\Delta_\ell \le 2\varepsilon\sqrt{C'_\ell}$ per Proposition 3 of [106]; bias-free and scalar operators contribute zero. By the propagation bound of Proposition 4 of [106], the pipeline is $\varepsilon_{\text{total}}$-approximately invariant, with $\varepsilon_{\text{total}} \to 0$ as $\varepsilon \to 0$. The bridge frame $\mathbf{T}$ itself is then only ε-approximately equivariant, perturbing $\mathbf{V}\mathbf{T}^\top$ by an $\mathcal{O}(\varepsilon)$ residual.

Empirical bound. We do not instantiate a closed-form $\varepsilon_{\text{total}}$: Proposition 4 requires Lipschitz constants that [106] derives only for `VNLinear`, leaving VN-ReLU, VN-MultiHeadAttn, and the non-globally-Lipschitz VN-LayerNorm open. We measure Δ directly. A dedicated rotation-invariance diagnostic applies $N = 100$ Haar-uniform SO(3) rotations to $M = 32$ fixed test clips for each of the three training seeds (3200 passes per seed, 9600 in total), executes the full pipeline end-to-end in fp32 with TF32 disabled, and reports the per-clip worst case $\Delta_{\max} = \max_R \|f(\mathbf{R} \circ \mathcal{G}) - f(\mathcal{G})\|_\infty$. An identity-rotation self-check ($\mathbf{R} = \mathbf{I}$) bounds the cuBLAS-reduction noise floor separately, so that any reported $\Delta_{\max}$ above it is rotation-induced rather than numerical. The measured values are deferred to §6.5.9.

Decision invariance. Let $m_{\min} := \min_x \big(z_{(1)}(x) - z_{(2)}(x)\big)$ denote the worst-case predicted-class margin over the test set, where $z_{(1)}$ and $z_{(2)}$ are the largest and second-largest logits. Because each logit coordinate can drift by at most $\Delta_{\max}$ in either direction, the predicted class is invariant under every tested rotation whenever:

$$m_{\min} \;>\; 2\,\Delta_{\max} \tag{5.18}$$

where the inequality bounds the worst-case logit perturbation by twice the worst-case logit-coordinate drift. The empirical verification of this inequality, together with the corresponding arg max-flip count, is reported in §6.5.9.

Relation to prior work. The per-layer machinery of this analysis is inherited from the VN-Transformer of [106]: the `VN-LinearWithBias` operator, the per-layer bias bound $\Delta_\ell \le 2\varepsilon\sqrt{C'_\ell}$ (Proposition 3), and the propagation bound (Proposition 4). The contribution here is to lift that single-encoder result to a complete 4D classification pipeline. The clean A/B/C partition confines every bias-bearing operator to A∪B, so the temporal stage carries no ε term; the VN-In bridge is identified as the sole equivariant-to-invariant boundary and recovers the maximal $3C - 3$ invariants through the identity $(\mathbf{V}\mathbf{R}^\top)(\mathbf{T}\mathbf{R}^\top)^\top = \mathbf{V}\mathbf{T}^\top$ without any orthonormality, non-degeneracy, or full-rank assumption on $\mathbf{T}$; and the analytic bound is converted into the operational decision-invariance criterion $m_{\min} > 2\Delta_{\max}$ (Eq. 5.18), verified by the end-to-end empirical diagnostic on temporal Gaussian sequences (§6.5.9).

5.6.9 Datasets

`MambaSplat-4D` is evaluated on three datasets, summarized in Table 5.4. **ModelSplat** (ModelNet10-GS and ModelNet40-GS) is a static, single-frame benchmark of 3D Gaussian Splats created by the ShapeSplat project [96] from ModelNet meshes [132], using the official ModelNet train and test splits; it anchors the attribute and pre-training studies (§6.5.1). **MSR-Action3D** [133] is a real depth-camera benchmark of point-cloud sequences that validates the temporal rotation protocol on non-Gaussian data (§6.5.2). **`AeroSplat-4D`** is the primary contribution dataset: synthetic temporal Gaussian-Splat sequences from the Stage 2 `DepthSplat-OC` pipeline, carrying the four-class classification and all architectural ablations (§6.5.3 onward).

[1] Non-degeneracy of $\mathbf{T}$ governs *how many* of the $3C - 3$ invariants are recovered (§5.6), not invariance itself.

The three datasets differ along two axes, and every per-dataset deviation in the following subsections reduces to one of them. The *static-versus-temporal* axis ($T = 1$ versus $T = 24$) governs the temporal windowing and the aggregation rule. The *Gaussian-versus-point-cloud* axis (fourteen attributes versus coordinates only) governs the Gaussian-lifting input and the attribute ablation. No two datasets share both axes. `AeroSplat-4D` itself comprises Sim-1 and Sim-2, split into identity-disjoint train, val, and test partitions, plus Sim-3, a held-out out-of-distribution probe rendered for OOD evaluation only.

Table 5.4: Datasets used to evaluate `MambaSplat-4D`. "Modality" indicates whether each frame carries the full fourteen-attribute Gaussian primitive or coordinates only. Train/Test for `AeroSplat-4D` is reported as identity counts pooled across Sim-1 and Sim-2; Sim-3 contributes 24 OOD test identities only. Full subset details: Table 5.1.

Dataset	Type	Modality	Source	Classes	Train/Test	Pts/frame
ModelSplat-10	Static	3DGS (14-attr.)	ShapeSplat [96]	10	3991 / 908	1024
ModelSplat-40	Static	3DGS (14-attr.)	ShapeSplat [96]	40	9842 / 2467	1024
MSR-Action3D	Temporal	Point cloud	Depth camera (real)	20	subj. 1–5 / 6–10	2048
AeroSplat-4D	Temporal	3DGS (14-attr.)	Isaac Sim → DepthSplat-OC	4	50/25/25 id	1024

Per-dataset render galleries for the ModelSplat-10 and ModelSplat-40 lifted Gaussian-Splat assets and the MSR-Action3D point-cloud sequences are collected in Appendix A.2 (Figures A.5–A.7).

5.6.10 Preprocessing

Each dataset is converted offline into a fixed-cardinality point set that the Stage-3 trainer consumes directly; the paragraphs below describe the shared decimation core and the per-dataset deviations summarized in Table D.1 (Appendix D).

Farthest-point decimation. ModelSplat and `AeroSplat-4D` share one decimation pipeline adapted from the ShapeSplat training procedure of [96]. Each frame is reduced from its raw point count to $N = 1024$ points through a fixed three-step chain: a random pre-subsample to 8192 points, farthest-point sampling (FPS) [93] to 1200 points, and a final random subsample to 1024 points. Whereas Ma et al. apply this chain at runtime during training, we precompute it offline because farthest-point sampling is GPU-intensive and offline preprocessing yields a significant reduction in per-epoch training cost. The chain is run under deterministic seeding so the decimation is reproducible. To inject sampling stochasticity without runtime cost, ModelSplat stores 25 such decimations per object and the trainer cycles through them across epochs; `AeroSplat-4D` stores a single decimation per frame, as discussed below.

ModelSplat. ModelSplat uses the ModelNet10 and ModelNet40 3D Gaussian Splat representations from the ShapeSplat dataset [96], which lifts ModelNet meshes [132] into per-object PLY files carrying the full fourteen-attribute Gaussian primitive. The decimation chain converts each object into a single-frame ($T = 1$) tensor of 1024 Gaussians, and the official ModelNet train/test split of [132] is used unchanged.

MSR-Action3D. MSR-Action3D is a real depth-camera benchmark and does not share the 3DGS decimation core. Raw depth maps are back-projected into per-frame point clouds of variable cardinality, and each clip is zero-centered once by a single centroid taken over all of its frames, as in the sequence recipe of [134]. Subtracting one constant offset removes the action's absolute placement in the depth-camera frame while preserving the inter-frame motion that distinguishes one action from another, whereas a per-frame centroid would re-anchor every frame to the origin and cancel exactly that motion, leaving only within-frame deformation. Because the point counts vary per frame, decimation to $N = 2048$ points is deferred to load time and resampled afresh each epoch rather than precomputed. The data carry coordinates only, and the standard subject-wise split, subjects 1–5 for training and 6–10 for testing, is used.

AeroSplat-4D. `AeroSplat-4D` frames are the per-frame 3D Gaussian Splats produced by the Stage 2 reconstruction (`DepthSplat-OC`). Each frame passes through the shared decimation chain to 1024 Gaussians with a single stored version per frame. Unlike ModelSplat's 25 subsamples per static object, only one subsample per frame is stored because multiple subsamples of the same frame in a temporal sequence would cause the model to observe each spatial configuration multiple times per temporal step, introducing a spatial bias that

Table 5.5: Per-experiment Stage-3 training recipes. All recipes minimize classifier cross-entropy under a fixed seed; gradients are clipped to unit ℓ_2 norm for every recipe except MSR-Action3D's. Label smoothing (ε=0.1) is applied to `AeroSplat-4D` only.

Experiment	Dataset	Optimizer	LR	Schedule	Epochs	Batch
Attributes	ModelSplat-10/40	AdamW (def. β, wd 10^{-2})	10^{-3}	cosine, warmup 10 ep	200	64
Rotation	MSR-Action3D	SGD (m 0.9, wd 10^{-4})	10^{-2}	multistep $\times$0.1 @{20, 30}, warmup 10 ep	50	8
Main+abl.	AeroSplat-4D	AdamW (β 0.9/0.95)	10^{-3}	cosine, warmup 3 ep	50	8

would dominate the temporal signal during training. Source clips span 30 frames per (asset, distance, elevation) bin, from which the trainer takes 24-frame windows. The splits are identity-disjoint, with no object instance appearing in more than one of the training, validation, and test sets, in an approximate 50/25/25 ratio.

5.6.11 Training Pipeline

Stage 3 is trained by one supervised procedure that is reused unchanged across all three datasets; the paragraphs below describe that procedure and the per-dataset deviations required to reproduce the results of Chapter 6. A complete per-dataset overview of preprocessing, training, augmentation, baselines, and evaluation is consolidated in Table D.1 (Appendix D).

Supervised trainer. A single trainer maps a preprocessed sequence of T frames to a class label. The per-dataset differences reduce to two axes: static versus temporal ($T = 1$ for ModelSplat, $T = 24$ for MSR-Action3D and `AeroSplat-4D`) and Gaussian-versus-point-cloud input. Training runs in TF32 matrix multiplications on a compiled model graph, under a fixed seed and deterministic cuDNN kernels, so that every reported number is reproducible. After each epoch, the model is evaluated on the validation split, and the highest-accuracy checkpoint is retained: video-level accuracy for the temporal datasets, clip-level accuracy for the static one.

Optimization. Every recipe minimizes cross-entropy on the classifier logits; no auxiliary or contrastive term is active on the supervised path. Table 5.5 lists the per-experiment recipes. The two ModelSplat recipes and the `AeroSplat-4D` recipe are the default recipes, whereas the MSR-Action3D recipe mirrors the temporal baselines (same optimizer, schedule, and epoch budget) so that the rotation-protocol comparison isolates the architecture rather than the training regime.

Rotation augmentation. Rotation is the central augmentation and is tied directly to the evaluation protocol (Table 5.6). A model tested under z/SO(3) or SO(3)/SO(3) is trained with the matching train-side rotation, either an azimuthal z-rotation or a Haar-uniform SO(3) rotation, drawn once per clip. For the Gaussian datasets, the rotation is applied jointly to each Gaussian's position and orientation quaternion; for MSR-Action3D, which stores coordinates only, the rotation is applied to the points alone.

Point-set and temporal augmentation. Each frame enters the trainer as a fixed-cardinality point set: 1024 points for ModelSplat and `AeroSplat-4D`, and 2048 for MSR-Action3D. ModelSplat cycles round-robin through 25 precomputed farthest-point decimations per object across epochs, `AeroSplat-4D` reuses its single precomputed decimation per frame (§5.6.10), and MSR-Action3D draws a fresh random subsample each epoch. ModelSplat applies isotropic scale jitter in $[\frac{2}{3}, \frac{3}{2}]$, while `AeroSplat-4D` and MSR-Action3D apply per-axis anisotropic scale jitter in $[0.9, 1.1]$; ModelSplat additionally applies per-axis translation of ± 0.2. For the two temporal datasets, a contiguous 24-frame window of consecutive frames is taken from each longer clip by a deterministic sliding window, with a three-clip stride between window starts on `AeroSplat-4D`. No coordinate jitter or point-dropout augmentation is used (distinct from the network dropout of §5.6.7, which stays active).

Baselines. Four external methods are retrained for comparison. Two static point-cloud classifiers, VN-PointNet and VN-DGCNN, are compared against `MambaSplat-4D` on the ModelSplat attribute study, both of which derive from the rotation-equivariant variants of [29]. Two temporal point-cloud models, P4Transformer [135] and Mamba4D [104], are compared on MSR-Action3D and on the four-class `AeroSplat-4D` table (§6.5.3). Each baseline is retrained from scratch on the identical splits using its own published training code and task hyperparameters; on the MSR-Action3D table, the epoch budget and batch size are unified across all methods,

so that the rotation-protocol comparison isolates the architecture rather than the training regime. Gaussian-MAE [96] is excluded on principle: its k-nearest-neighbor grouping over raw coordinates presupposes a canonical orientation and is therefore not SO(3)-invariant, so retraining it under rotation augmentation would conflate architectural capacity with task difficulty.

Internal ablations. The remaining Stage-3 experiments are internal controls that hold the `AeroSplat-4D` recipe fixed and vary a single component. The bridge-method ablation (§6.5.4) replaces the VN-In bridge with a vector-norm pool, Gram-matrix invariants, pairwise-distance invariants, or a PCA canonical frame; the component ablation (§6.5.5) ablates the feature lifting, spatial encoder, bridge, and temporal modules in turn; and §6.5.6 onward vary the Mamba hidden width, the Mamba depth, the temporal window and stride, and the loss term, respectively.

5.6.12 Evaluation Protocol

Stage 3 is evaluated under controlled rotation perturbations to measure the SO(3) invariance that motivates the architecture. The rotation-evaluation protocols are defined first, followed by the per-dataset deviations.

We follow the three-protocol convention of [29], extended with one per-frame variant for the temporal setting. All protocols apply rotations jointly to $(\mathbf{p}, \mathbf{q})$ via the Gaussian rotation rule (Eq. 3.5). Five protocols are evaluated. The **none/none** identity protocol is a sanity baseline; **z/z** applies a z-axis rotation at both train and test time to probe azimuthal generalization; **z/SO(3)** trains under z-axis rotation but tests under full SO(3) and acts as the invariance stress test; **SO(3)/SO(3)** draws Haar-uniform rotations on both sides; and **z/SO(3)_pf** trains under z-axis rotation while drawing one independent SO(3) rotation *per frame* at test time, stressing correspondence-free temporal robustness. Train rotations are always drawn at clip level (one matrix per sequence); test rotations are clip-level except for the per-frame variant.

Protocol	Train rotation	Test rotation	Purpose
none/none	-	-	identity sanity
z/z	z-axis, one per clip	z-axis, one per clip	azimuthal generalization
z/SO(3)	z-axis, one per clip	SO(3), one per clip	invariance stress test
SO(3)/SO(3)	SO(3), one per clip	SO(3), one per clip	full generalization
z/SO(3)_pf	z-axis, one per clip	SO(3), *per frame*	correspondence-free test

Table 5.6: Rotation-evaluation protocols. Train rotations are clip-level (one matrix per sequence); test rotations are clip-level except `_pf`, which draws an independent SO(3) matrix per frame.

Rotation protocols. Every model is evaluated under the rotation protocols of Table 5.6: an unrotated sanity baseline, azimuthal z-rotation at train and test, the z/SO(3) invariance stress test, and full SO(3)/SO(3) generalization. The test rotations are drawn once per clip and applied jointly to the Gaussian position and orientation for the Gaussian datasets, and to the points alone for MSR-Action3D. Each configuration is repeated over three independent rotation draws (one for the component ablation, §6.5.5), and accuracies are reported as the mean and standard deviation across rotation trials and seeds.

Per-frame rotation. The two temporal datasets add a fourth, correspondence-free protocol in which an independent SO(3) rotation is drawn for every frame of a clip rather than one per clip. This stresses temporal robustness without assuming inter-frame orientation consistency. On static ModelSplat, the protocol is degenerate because the dataset has only a single frame and collapses to ordinary SO(3) evaluation.

Clip- and video-level accuracy. For the temporal datasets, each clip yields a softmax prediction, and clip-level accuracy is complemented by video-level accuracy, obtained by summing softmax probabilities over all clips of a video before taking the argmax. ModelSplat, having one frame per object, is reported at clip level only. Each configuration is averaged over three rotation trials (one for the component ablation, §6.5.5), and the video-level prediction is:

$$\text{pred\_video} = \arg\max\Big(\sum_{\text{clips}} \text{softmax}(\text{logits}_{\text{clip}})\Big) \tag{5.19}$$

where the inner softmax converts per-clip logits to a probability distribution and the outer argmax selects the dominant class across the video.

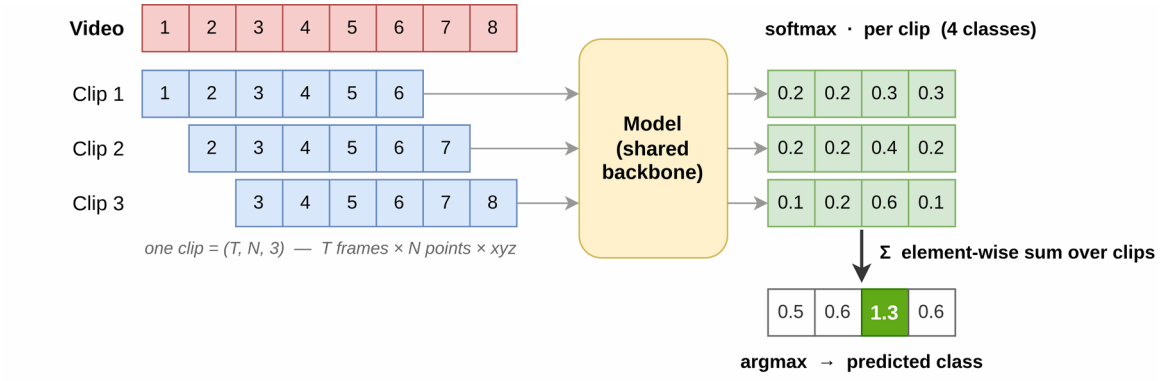


Figure 5.6: Illustration of per-clip versus video-level accuracy aggregation.

Invariance drift. Beyond classification accuracy, the residual deviation of the network output under rotation is measured directly: a batch of clips is passed through the model under many Haar-uniform rotations, and the spread of the resulting logits quantifies the quasi-equivariant drift bounded in §5.6.8. This confirms that the ε bias term leaves the predicted class unchanged in practice.

Out-of-distribution probe. `AeroSplat-4D` additionally evaluates on the Sim-3 subset, a line-formation outdoor scene whose 24 object identities are held out entirely from training. Sim-3 is rendered for evaluation only and probes generalization to an unseen camera rig and background, without any change to the trained model or to the rotation protocols above.

5.6.13 Loss Formulation

The default Stage-3 objective on `AeroSplat-4D` is a single-term clip-level cross-entropy with VN-DGCNN-style label smoothing; the ModelSplat attribute study (§6.5.1) and the MSR-Action3D rotation study (§6.5.2) use standard cross-entropy ($\varepsilon = 0$) to follow the point-cloud classification convention and match the temporal baselines (Table 5.5). The predicted distribution $\hat{p}_b \in \Delta^{K-1}$ is the softmax of the classifier logits $z_b \in \mathbb{R}^K$ produced from the temporally pooled Mamba sequence (§5.6.7); the ground-truth class is y_b, with K classes and a batch of size B. The smoothed target $\tilde{q}_b$ assigns $1-\varepsilon$ to the true class and spreads the remaining mass uniformly over the $K-1$ distractors, with $\varepsilon = 0.1$:

$$\tilde{q}_{b,c} = \begin{cases} 1-\varepsilon & \text{if } c = y_b, \\ \dfrac{\varepsilon}{K-1} & \text{otherwise,} \end{cases} \qquad \varepsilon = 0.1 \tag{5.20}$$

where $\tilde{q}_{b,c}$ is the smoothed target probability for sample b at class c and y_b is the integer ground-truth class label. The clip-level cross-entropy is then taken against the smoothed target:

$$\mathcal{L} = \mathcal{L}_{\text{CE-LS}} = -\frac{1}{B}\sum_{b=1}^{B}\sum_{c=1}^{K} \tilde{q}_{b,c} \log \hat{p}_{b,c} \tag{5.21}$$

where $\hat{p}_{b,c}$ is the predicted probability of class c for sample b and $\mathcal{L}_{\text{CE-LS}}$ denotes label-smoothed cross-entropy. The label-smoothing form follows the VN-DGCNN convention: the mass $\varepsilon/(K-1)$ is distributed only over the non-target classes, rather than spreading ε/K uniformly over all classes including the target.

5.6.14 Computational Complexity

Table 6.19 reports parameter counts, FLOPs, wall-clock latency, and peak VRAM for `MambaSplat-4D` against the P4Transformer and Mamba4D baselines on a single $T = 24$, $N = 1024$ sequence. The $E(C)$ feature mode realises 1.85 M parameters and the $E(X)$ mode realizes 1.88 M parameters, so the marginal cost of moving from

Table 5.7: Paradigm asymmetries between the 2D baselines and the 3DGS pipeline. Each row is an inherent consequence of the representation choice, not a confound.

Dimension	2D Baselines (B1–B4)	3DGS Pipeline
Input domain	2D RGB pixels	3D Gaussian primitives (14-dim)
Scene context	Full 2560×1440 scene (Sim-2)	Isolated object (N=1024, unit-sphere)
Temporal context	Single-frame + majority vote (B1, B4); detection + tracking (B2); multi-frame fusion (B3)	24-frame VN-Mamba SSM
Rotation invariance	None (viewpoint-dependent)	SO(3)-invariant by construction
Upstream cost	Direct sensor input	Stage 1 segmentation + Stage 2 reconstruction

Table 5.8: Comparison design: input, resolution, and training regime per baseline. All rows share the same identity-disjoint split and evaluation protocol. The *Tier* column indicates: **S**ystem-level or **C**lassifier.

Row	Method	Input type	Resolution	Pre-trained	Params	Temporal strategy	Tier
B1	WRN-YOLO	Full scene	960px	✓	73.6 M	Single-frame	S
B2	YOLOv7 + CSRT	Full scene	640px	✓	6.2 M	Detection + tracking	S
B3	FBOD-SV	Full scene	384×672	✓	28.6 M	Multi-frame fusion	S
B4	YOLO26-cls	Full scene	1440px	✓	∼1.4 M	Majority vote	C
Ours	MambaSplat-4D	3DGS point cloud	N=1024 Gaussians	×	1.88 M	24-frame Mamba SSM	C

$E(C)$ to $E(X)$ is only +0.03 M. The resulting model is approximately 23× smaller than P4Transformer [135] (44.10 M) and approximately 58× smaller than Mamba4D [104] (109.77 M), while retaining SO(3) invariance.

5.7 Stage 4 – End-to-End Integration

Stage 4 chains the four stages into one runnable pipeline and asks the bottom-line question: can a reconstruct-then-classify 3D pipeline match conventional 2D image pipelines on the same surveillance task? It is a comparison of two paradigms rather than a single-variable ablation (analogous to comparing LiDAR-based and camera-based perception in autonomous driving), and it also reports the per-stage latency that decides whether the system runs in real time.

5.7.1 Pipeline Composition and Comparison Design

Stage 4 composes the preceding stages into a single pipeline and benchmarks it against 2D appearance-based baselines. The executable chain runs Isaac Sim rendering, ground-truth mask extraction, feed-forward 3DGS reconstruction, and `MambaSplat-4D` classification in sequence; intermediate artifacts are reused across re-runs.

The Stage-4 evaluation is a *paradigm comparison*, not a single-variable ablation: it asks whether a 3DGS-lifting pipeline can match conventional 2D appearance-based pipelines when each operates in its natural configuration on the same surveillance task, which is analogous to comparing LiDAR-based and camera-based perception in autonomous driving. The 2D baselines receive full-scene images; the 3DGS pipeline receives reconstructed object geometry. Table 5.7 enumerates the resulting asymmetries, which are inherent to the paradigm difference. Where Stage 3 (§5.6.11) isolates the classifier architecture by comparing temporal point-cloud models on identical inputs, Stage 4 evaluates the representation choice by comparing complete pipelines.

The comparison spans two levels. *System-level* (B1–B3): deployed 2D detect-track-classify pipelines on full-scene images, measuring end-to-end capability. *Classifier-level* (B4 and Ours): a pure 2D image classifier and `MambaSplat-4D`, which remove the detection and tracking overhead of the system-level baselines by classifying localized inputs directly. The two classifiers differ in initialization (B4 from ImageNet-pretrained weights, our classifier from scratch) as well as in representation, so this contrast is interpreted at the system level rather than as a clean isolation of the representation choice. Table 5.8 details the input, resolution, and training regime per baseline.

All models share the same metric suite (accuracy, per-class F1, macro F1, and confusion matrices) but use aggregation rules matched to their output type. The 2D baselines (B1–B4) emit discrete per-frame class labels aggregated by majority vote with confidence-based tiebreaking; `MambaSplat-4D` emits per-clip logits and aggregates by the softmax-sum-then-argmax rule defined in §5.6.12. `MambaSplat-4D`'s classifier trains from scratch for 50 epochs on data derived from the same 46 training assets, whereas the 2D baselines (B1, B2, B4) use ImageNet/COCO-pretrained initialization; the comparison is therefore system-level rather than an isolation of representation from initialization (see §5.7.2). B4 uses AdamW (lr=10^{-3}, cosine schedule, 5-epoch warmup, weight decay 0.05, label smoothing 0.1). All models use the same identity-disjoint 50/25/25 train/val/test splits of §5.3.4 and a fixed seed of 42.

5.7.2 2D Baselines

We compare against four 2D baselines spanning the WOSDETC drone-versus-bird challenge lineage and the broader small-object detection literature [6]. **B1 WRN-YOLO** is a YOLOv5-style single-frame detector with a Wide-ResNet50-2 backbone [136]. **B2 YOLOv7 + CSRT** [137] pairs a YOLOv7-tiny detector with a CSRT tracker that bridges missed detections across frames. **B3 FBOD-SV** [138] is a multi-frame CSPDarknet53 fusion detector designed for small flying objects in surveillance video. **B4 YOLO26-cls** [139] is a pure image-classification head on a recent open-source YOLO backbone. B1, B2, and B4 are initialized from ImageNet/COCO-pretrained weights and B3 from its published backbone, after which all four are retrained on `AeroSplat-4D` with hyperparameters fixed to each method's published configuration rather than retuned. This is a system-level comparison and not an isolation of representation from initialization: both the 2D baselines and `MambaSplat-4D` rely on large-scale pretraining (ImageNet/COCO for the 2D detectors; Objaverse for the Stage-2 DepthSplat reconstructor), and the pipelines further differ in input resolution, object localization, temporal modeling, and multi-view fusion. We therefore report accuracy at the system level and analyze the dominant confound, effective object resolution, directly from the target's pixel footprint. At the Sim-2 operating distances, targets span roughly 7–120 pixels (median ∼45 pixels) along their longer axis, under 0.2% of the 2560×1440 frame area, placing all four baselines in a small-object regime.

5.7.3 Latency Profiling

Per-stage efficiency is measured under the protocol formalized in §5.6.14 and reported in full in Table 6.19. All latencies are the median of 30 timed runs after 5 warm-ups, with GPU-synchronized timing on an RTX 5090. Stage-2 latency covers the reconstruction-model forward pass only; the custom CUDA rasterizer is excluded because its cost depends on the target-camera configuration. Stage-3 latency is reported per-frame as the per-sequence latency divided by $T = 24$, which is valid because the temporal Mamba SSM is linear in T. Custom CUDA kernels (the rasterizer and farthest-point sampling) are not traced by the FLOP counter; pre-pool rows omit parameter and FLOP cells accordingly.

6

Experiments and Results

This chapter evaluates the `AeroSplat-4D` pipeline through four research questions, one per pipeline stage. RQ1 (§6.3) asks whether the synthetic dataset-generation stage produces class-balanced, geometrically diverse, fully annotated multi-camera sequences. RQ2 (§6.4) measures the quality of the feed-forward 3DGS reconstruction stage. RQ3 (§6.5), which is the core contribution of this thesis, evaluates rotation-invariant 4D classification. RQ4 (§6.6) reports end-to-end pipeline latency and accuracy. A shared experimental protocol (§6.1) and a dataset summary (§6.2) precede the four research questions, and every results table refers back to the Method chapter 5 for the architectural and training detail behind each experiment.

6.1 Reproducibility and Experimental Protocol

Every experiment in this chapter is produced under a single reproducibility envelope: a fixed seed, deterministic kernels, cached rotation matrices, and a fixed aggregation and hardware configuration. The per-experiment training recipes are stated once in the Method chapter 5 Table 5.5 and are not repeated here.

6.1.1 Determinism

Every configuration is trained under three random seeds (42–44), reduced to a single seed for the attribute ablation (§6.5.1) and the MSR-Action3D rotation study (§6.5.2), so that reported variance reflects training noise. A single seeding entry point makes each run bit-for-bit reproducible by fixing all RNG state and enabling PyTorch's deterministic-algorithm and cuDNN settings. Rotation matrices, a second source of run-to-run variation, are pre-generated once with a Haar-uniform SO(3) sampler [130] and cached, and every model within a table reads the same cache. Each clip is therefore assigned a fixed rotation independent of model or batch size, making rotation-robustness comparisons exactly paired.

6.1.2 Hardware and Software

All experiments run on a single NVIDIA RTX 5090 (32 GB VRAM) under CUDA 13.x and PyTorch 2.10, with `torch.compile` enabled for both supervised stages. Stage-2 reconstruction trains in bf16-mixed precision and Stage-3 classification in fp32 with TF32 matmuls. Latency figures report the median of 30 timed runs after 5 warm-up iterations (unless a caption notes otherwise), with CUDA synchronization around each measurement.

6.1.3 Baseline Retraining

All baselines are retrained on the same train, validation, and test splits as our own models, with hyperparameters matched to each baseline's original published configuration rather than re-tuned, so that accuracy differences reflect the architecture rather than per-experiment re-tuning. Their training recipes are given alongside ours in Table 5.5 (§5.6.11).

6.1.4 Evaluation Metrics

Every experiment reports against the three evaluation axes required in the Problem Setting chapter 2: classification, 3D reconstruction, and computational performance. This subsection gives the canonical definition of each metric and collects them in Table 6.1; the per-stage evaluation sections of the Method chapter 5 refer back to this catalog rather than redefining the metrics.

Classification is measured using overall accuracy, per-class precision and recall, F1 score, and confusion matrices, all reported across the four-class taxonomy so that class-specific biases remain visible rather than being absorbed into an aggregate score. Reconstruction is measured on rendered novel views: peak signal-to-noise ratio (PSNR) [140], in decibels, captures pixel-level fidelity, with higher values indicating lower mean-squared error against the ground-truth image; structural similarity (SSIM) [31] captures perceived similarity from luminance, contrast, and local structure on a 0-to-1 scale, with higher being better; and Learned Perceptual Image Patch Similarity (LPIPS) [127] captures perceptual distance through deep-network features, with lower being better. Computational performance is measured by latency, throughput, and memory footprint: latency is the end-to-end processing time, profiled under the median-of-runs protocol described in the hardware-and-software subsection above (§6.1.2); throughput is reported in frames per second on the target, or available hardware; and memory footprint is the peak GPU or CPU memory use.

Table 6.1: Evaluation metric catalogue across the three axes: classification, 3D reconstruction, and computational performance. Reconstruction metrics are computed on rendered novel views; the latency-measurement procedure follows the hardware-and-software protocol of §6.1.2.

Category	Metric	Description
Classification	Accuracy	Overall correct classification rate across the four classes
	Precision and recall	Per-class metrics for identifying class-specific biases
	F1 score	Harmonic mean of precision and recall
	Confusion matrix	Detailed error analysis over false positives and false negatives
3D reconstruction	PSNR [140]	Pixel-level fidelity of rendered novel views; higher is better
	SSIM [31]	Structural agreement between rendered and ground-truth views; higher is better
	LPIPS [127]	Perceptual distance between rendered and ground-truth images; lower is better
Computational	Latency	End-to-end processing time
	Throughput	Frames per second on target or available hardware
	Memory footprint	Peak GPU or CPU memory usage

6.2 Datasets

Each dataset is described alongside the stage that consumes it rather than cataloged here. The synthetic `AeroSplat-4D` dataset (Sim-1, Sim-2, and Sim-3, with their rig, scene, and split protocol) is defined in §5.3.4 (Table 5.1). The Stage-2 reconstruction pipeline is defined in §5.5.2 (Table 5.2): Objaverse [126] for pre-training, and Google Scanned Objects (GSO) [123], Amazon Berkeley Objects (ABO) [124], and the `AeroSplat-4D` Sim-1 distance-sweep corpus for evaluation. The Stage-3 classification datasets (ModelSplat-10/40, MSR-Action3D, and `AeroSplat-4D`) are defined in §5.6.9 (Table 5.4), with per-dataset preprocessing and augmentation in §5.6.10.

6.3 RQ1 – Synthetic Dataset Generation (Stage 0)

We ask whether `AeroSplat-4D` provides class-balanced, geometrically diverse, fully annotated multi-camera sequences for supervising Stages 2 and 3. The five experiments below characterize `AeroSplat-4D` along five complementary axes: dataset composition (§6.3.1), geometric coverage (§6.3.2), annotation consistency (§6.3.3), downstream reconstruction quality (§6.3.4), and temporal motion structure (§6.3.5). Sim-3 was rendered for its 24 held-out test identities only, so every Sim-3 statistic below is computed on that subset.

6.3.1 Dataset Composition

We first position `AeroSplat-4D` against established benchmarks (Table 6.2), then quantify its class balance and the integrity of its identity-disjoint splits (Table 6.3). Table 6.2 groups the comparison set into seven real airborne datasets, two synthetic airborne datasets (SynDroneVision [141] and UEMM-Air [142]), and ShapeSplat [96] as a 3DGS anchor, drawn from the broader anti-UAV detection and tracking landscape [143]. `AeroSplat-4D` is distinguished less by raw scale (AOT alone reaches 5.9 M frames against Sim-1's 54 600) than by completeness of supervision and multi-camera capture: where the real airborne datasets use a single camera (two for Anti-UAV) and provide 2D bounding boxes, occasionally with track ID or range, every `AeroSplat-4D` sequence carries RGB, mask, depth, 3D position, 6-DoF trajectory, intrinsics, extrinsics, and 3DGS ground truth across four synchronized cameras (Sim-1) or five (Sim-2, Sim-3) at 2560 × 1440 over a 2–380 m target-distance envelope.

Class balance and split integrity follow the established dataset documentation practice [144]. The normalized Shannon entropy $H/H_{\max}$ stays between 0.959 and 0.978 across all seven (Sim, split) cells of Table 6.3, confirming that no single class dominates. The Gini coefficient (0.722–0.735) reflects the intentional ∼2:1 ratio between the bird/drone classes and the helicopter/airplane classes, which are visible as roughly 9000 versus 4800 frames per class in the Sim-1 training split. However, this is by design rather than a sampling artifact.

Identity-disjoint split integrity is verified by a three-part leakage protocol [145]. First, USD-filename hash equality across splits confirms that the asset identities have no intersection. Second, the nearest-neighbor distance between DINOv2 [146] thumbnail embeddings is compared between the train→test and train→train pairs, so that a near-duplicate identity across splits would surface as an anomalously short distance. Third, a Uniform Manifold Approximation and Projection (UMAP) scatter colored by split (appendix) provides a visual cross-check.

Table 6.2: `AeroSplat-4D` positioned against related airborne, multi-view, and 3DGS benchmarks. The R/S column marks real versus synthetic capture; #cls, #id, #seq, #frm, and #cams give the number of classes, identities, sequences, frames, and cameras; Res. is the image resolution and Dist. the target-to-camera distance range (nominal design range for the `AeroSplat-4D` subsets). The GT-modality flags abbreviate RGB, mask (M), depth (D), 3D position (3D), 6-DoF trajectory (6-DoF), intrinsics (K), extrinsics ([R|t]), and 3DGS. BBox = 2D bounding box; N/R = not reported; – = not applicable.

Dataset	R/S	#cls	#id	#seq	#frm	#cams	Res.	Dist. (m)	GT modalities	License
Drone-vs-Bird [6]	R	1	N/R	77	85904	1	up to 3840 × 2160	N/R	BBox	Non-comm.
AOT [147]	R	4	N/R	4943	5.9M	1	2448 × 2048	0–700	BBox / ID / range	CDLA-Perm.
Anti-UAV [148]	R	1	N/R	318	580k	2	1920 × 1080 / 640 × 512	N/R	BBox / RGB / IR	Research
ARD-100 [149]	R	1	N/R	100	202467	1	1920 × 1080	N/R	BBox	CC BY 4.0
NPS-Drones [150]	R	1	N/R	50	70250	1	1920 × 1280	N/R	BBox / ID	Research
USC-Drones [151]	R	1	1	30	27000	1	1920 × 1080	N/R	BBox	Custom
FL-Drones [4]	R	2	N/R	40	12k	1	752 × 480	N/R	BBox	On-request
SynDroneVision [141]	S	1	13	N/R	140038	1	2560 × 1489	N/R	RGB / BBox	CC BY 4.0
UEMM-Air [142]	S	multi	N/R	N/R	120000	1	N/R	N/R	RGB / D / M / normals / IMU / text	Research
ShapeSplat [96]	S	87	65k	–	–	multi	N/R	–	3DGS / K / [R|t] / RGB	ShapeNet ToU
AeroSplat-4D (Sim-1)	S	4	91	1820	54600	4	2560 × 1440	2–280	RGB / M / D / 3D / 6-DoF / K / [R|t] / 3DGS	CC BY 4.0†
AeroSplat-4D (Sim-2)	S	4	91	182	16380	5	2560 × 1440	40–65	RGB / M / D / 3D / 6-DoF / K / [R|t] / 3DGS	CC BY 4.0†
AeroSplat-4D (Sim-3)	S	4	24	24	2880	5	2560 × 1440	80–380	RGB / M / D / 3D / 6-DoF / K / [R|t] / 3DGS	CC BY 4.0†

† CC BY 4.0 covers the released `AeroSplat-4D` artifacts (renders, instance masks, depth, 3D positions, 6-DoF trajectories, camera intrinsics/extrinsics, and 3DGS parameters). The third-party 3D models and the NVIDIA scene used during rendering are not redistributed and retain their original licenses.

Table 6.3: Class balance and identity-disjoint split integrity. Columns: asset, sequence, and total frame counts (#assets, #seq, #frames); the per-class frame counts Bird, Drone, Heli (helicopter), and Plane; the normalized Shannon entropy $H/H_{\max}$ (higher is more balanced); and the Gini coefficient (lower is more balanced). The identity-disjoint split per Sim is 50/25/25; see Appendix D.

Sim	Split	#assets	#seq	#frames	Bird	Drone	Heli	Plane	$H/H_{\max}$ ↑	Gini ↓
Sim-1	train	46	920	27600	9000	9000	4800	4800	0.966	0.727
Sim-1	val	21	420	12600	3600	4200	2400	2400	0.978	0.735
Sim-1	test	24	480	14400	4800	4800	2400	2400	0.959	0.722
Sim-2	train	46	92	8280	2700	2700	1440	1440	0.966	0.727
Sim-2	val	21	42	3780	1080	1260	720	720	0.978	0.735
Sim-2	test	24	48	4320	1440	1440	720	720	0.959	0.722
Sim-3	test	24	24	2880	960	960	480	480	0.959	0.722

6.3.2 Geometric Coverage

This experiment demonstrates that `AeroSplat-4D` provides sufficient spatial, scale, and multi-camera visibility coverage to supervise Stages 2 and 3. Every metric aggregates the per-frame ground-truth fields emitted by Stage 0.

Four figures characterize this coverage along complementary axes. Figure 6.1 shows the per-camera centroid heatmap on a log-density scale, following the WILDTRACK multi-view-detection convention [84]. Figure 6.2 reports the per-class object pixel-area distribution on the SODA small-object bins [152]. Figure 6.3 gives the per-class distance distribution, with dyadic-octave bins matched to the Sim-1 distance sweep, following large-scale benchmark practice [153]. Figure 6.4 plots the multi-camera visibility CDF, the probability that an object is visible in at least k cameras for $k = 1 \dots N_{\text{cam}}$. Together, they confirm that the dataset spans the spatial, scale, distance, and visibility regimes the downstream stages must handle.

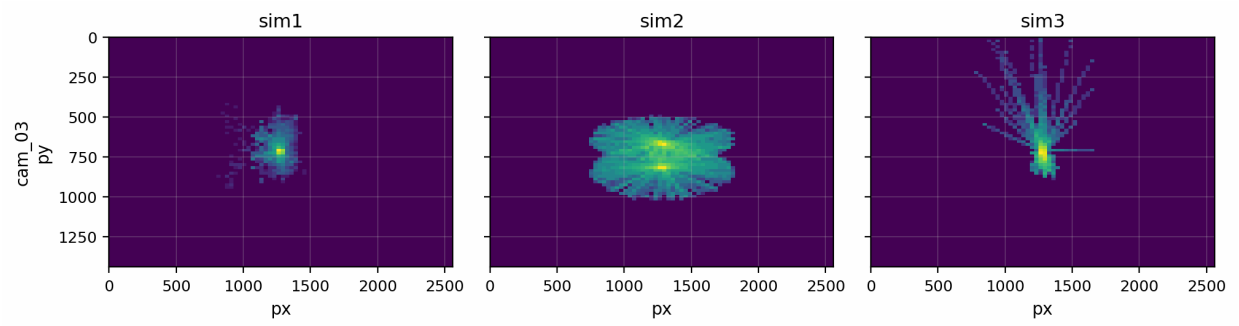


Figure 6.1: Per-camera log-density centroid heatmap for camera `cam_03` across Sim-1, Sim-2, and Sim-3. Sim-1 yields a tight centroid blob; the full camera-by-Sim grid appears in Appendix B.

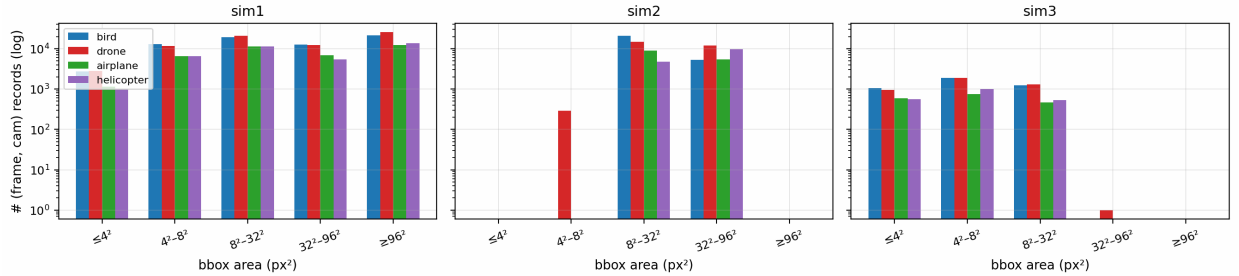


Figure 6.2: Per-class object pixel-area distribution on SODA-style log-scale bins, one panel per Sim with the four classes overlaid.

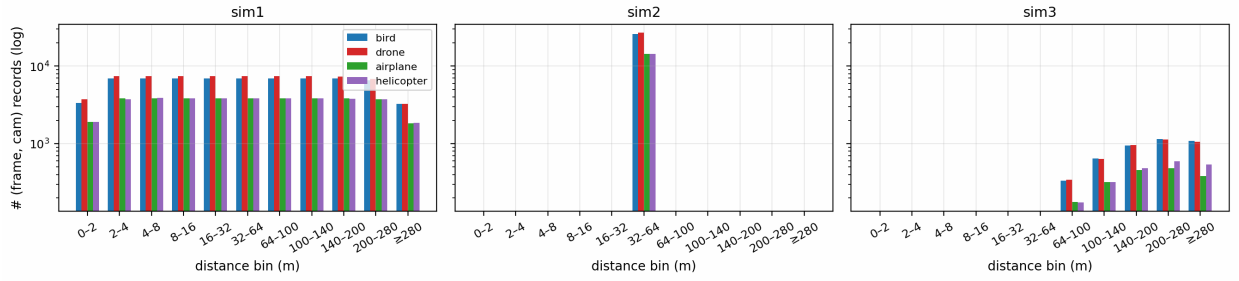


Figure 6.3: Per-class distance-to-camera distribution, with dyadic-octave bins matched to the Sim-1 distance sweep $\{2, 4, 8, 16, 32, 64, 100, 140, 200, 280\}$ m, one panel per Sim.

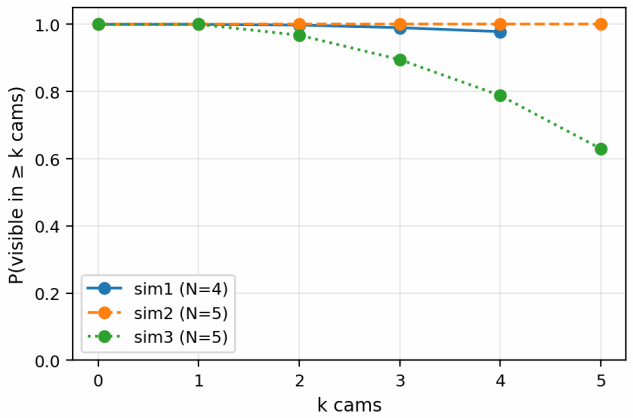


Figure 6.4: Multi-camera visibility CDF, the probability that an object is visible in at least k cameras for $k = 1 \dots N_{\text{cam}}$, one curve per Sim.

6.3.3 Annotation Consistency

Synthetic ground truth has no human annotators, so inter-rater agreement does not apply. Following synthetic-data generators such as Kubric [116] and the broader synthetic-data literature [154], Table 6.4 instead reports internal cross-modality consistency: residuals between the simulator's own mask, 3D-trajectory, and camera-pose outputs, which are authoritative by construction.

The table reports three residuals. First, mask-versus-projected-bbox IoU: the overlap between the rendered mask's tight box and a box centered on the projected 3D centroid. The mask box is not compared directly to the stored `bbox_2d`, which would be tautological, since the simulator emits `bbox_2d` as the mask's tight box by construction. Second, the re-projection residual: the pixel distance between the projected 3D centroid and the 2D box centroid. Third, trajectory smoothness, $\max_t \|X_{t+1} - X_t\|/\bar{v}$, which flags teleportation or discontinuities in the motion path.

Table 6.4: Internal cross-modality annotation consistency on `AeroSplat-4D`. Synthetic ground truth precludes claims of annotation errors; the table reports internal residuals between rendered masks, 3D trajectories, and camera poses. Each metric is given as median (p5–p95).

Sim	Mask vs proj. bbox IoU ↑	Re-proj. residual (px) ↓	Traj. smoothness max $\Delta X/\bar{v}$ ↓
Sim-1 (all splits)	0.739 (0.273–1.000)	3.34 (0.53–105.49)	2.62 (1.85–4.13)
Sim-2 (all splits)	0.768 (0.349–0.967)	2.92 (0.60–11.78)	1.07 (1.01–1.47)
Sim-3 (test ids only)	0.500 (0.000–1.000)	1.38 (0.37–27.61)	1.01 (1.01–1.05)

The Sim-1 re-projection residual has a heavy tail (p95 $\approx$ 105 px against a $\approx$ 3 px median). This follows from the distance sweep: at 200–280 m a target subtends fewer than 5 pixels, so a sub-degree offset between the projected and mask centroids maps to a residual of roughly 100 px at full resolution. Sim-2 and Sim-3, which span narrower distance ranges, keep tighter p95 values. The Sim-3 mask-versus-bbox IoU drops to a p5 of 0.000 because the line-formation rig produces partial-visibility frames, where the projected bounding box extends outside the camera frustum while the rendered mask is clipped at the frame boundary.

6.3.4 Downstream-Reconstruction Quality

This experiment bounds the operating regime of Stage 2 reconstruction and surfaces its long-tail failure cases, complementing the controlled distance-degradation sweep of §6.4.3.

Figure 6.5 reports reconstruction quality (PSNR, SSIM, LPIPS) binned by ground-truth pixel area and by distance, reusing the bins of §6.3.2. Figure 6.6 gives the per-frame Gaussian-count distribution, which guards against a misreading of the first figure: a low Gaussian count at long distance follows from a small projected object, not from a reconstruction failure.

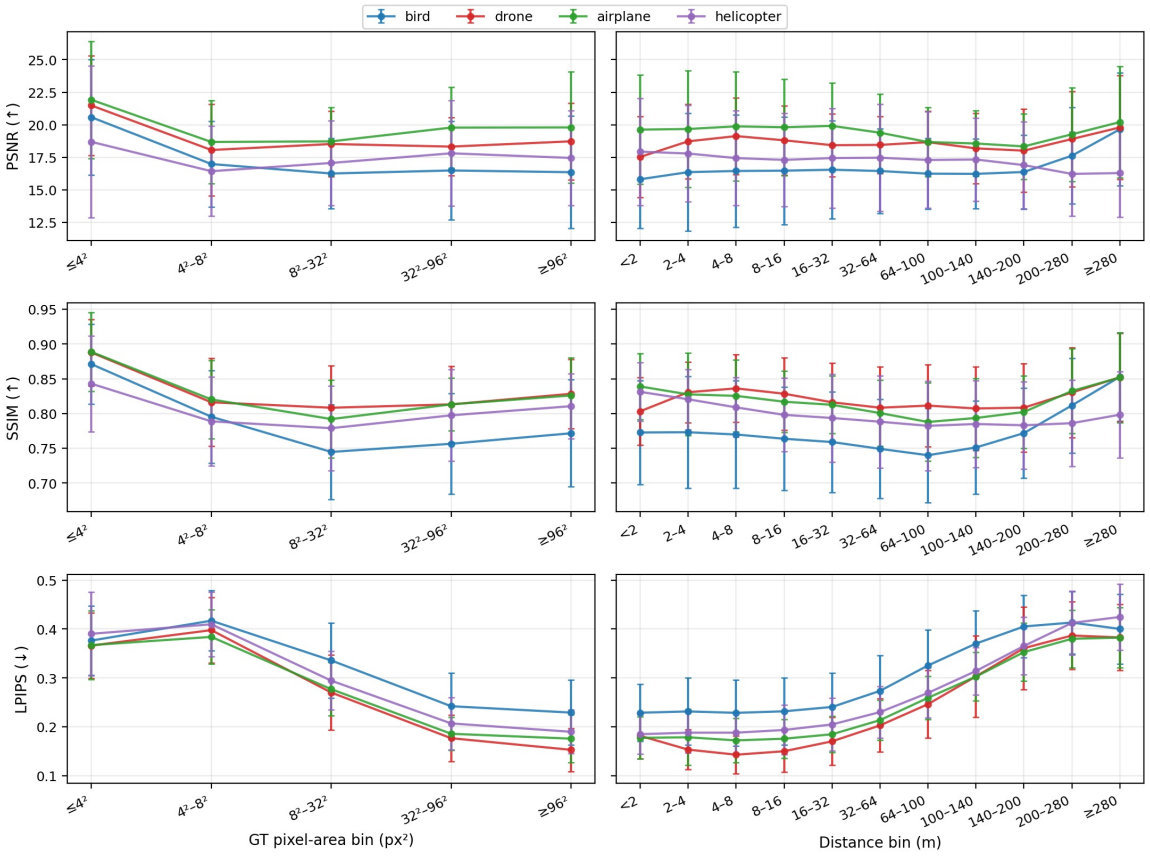


Figure 6.5: Reconstruction quality binned by ground-truth pixel area (left) and distance (right), Sim-1 only, `DepthSplat-OC` reconstruction with per-class color.

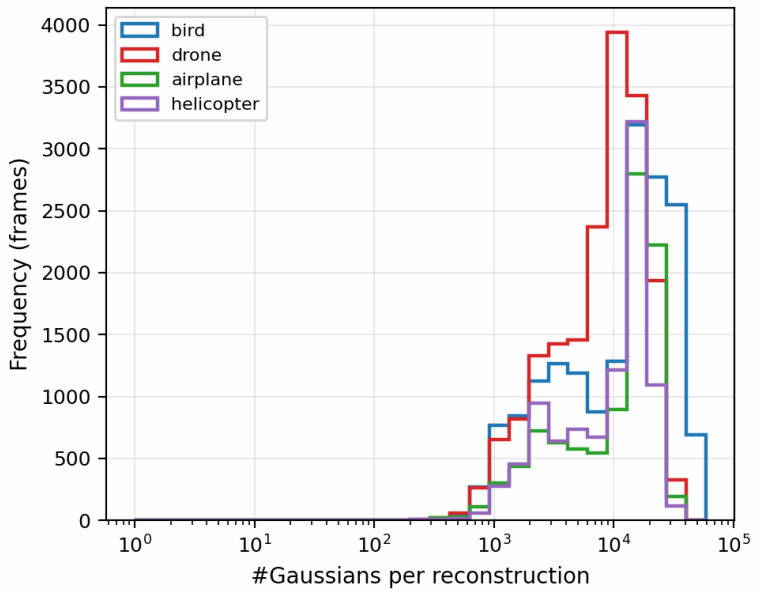


Figure 6.6: Per-frame Gaussian-count distribution from Stage-2 reconstructions on Sim-1, log x-axis, per-class color.

6.3.5 Temporal Characterization

Per-class motion signatures distinguish the four classes in time. We compute a symmetric pairwise Chamfer distance matrix over the farthest-point-sampled (FPS) 3DGS point clouds of all frame pairs in each 30-frame sequence. Chamfer distance, the mean of the symmetric nearest-neighbor distances between two unordered point sets, is order-invariant, making it appropriate for 3DGS reconstructions, which have no canonical point ordering. The distance-to-frame-1 curve $d(\cdot, f_1)$ measures how the reconstructed geometry departs from a reference pose, so periodic motion appears as recurring minima.

The analysis covers 1,616 complete (category, distance, elevation, asset) sequences across ten camera distances from 2r to 280r. For each sequence, the 30×30 Chamfer matrix is computed, and the two elevation viewpoints ($\pm 20°$) are averaged to reduce per-viewpoint noise. Per-category, per-distance aggregation then yields the mean and $\pm 1\sigma$ envelope of the $d(\cdot, f_1)$ curves across assets. The airplane and helicopter classes contribute only 16 assets each, compared to 29 birds and 30 drones, so their inter-asset estimates are noisier, and those two classes have wider $\pm 1\sigma$ bands. A frequency-domain FFT provides corroboration: each DC-removed 30-sample curve maps to 16 one-sided bins ($k = 0 \ldots 15$) spaced at $1/30 \approx 0.033$ cycles per frame, from DC ($k = 0$) to the Nyquist frequency ($k = 15$, 0.5 cycles per frame), the same grid for every distance band. This resolution is coarse, so the FFT corroborates the time-domain periods rather than supporting standalone spectral claims.

Figure 6.7 shows the 4r (close-range) results. Birds give the strongest, most coherent signal, with V-shaped dips at an approximate period of 12 frames, a dominant low-frequency FFT peak at bin 2, and a $\pm 1\sigma$ band tight relative to the signal across all 29 bird assets. Drones oscillate at roughly 8 frames with wider inter-asset spread, and helicopters at roughly 16 frames. Airplanes give the weakest signal: the estimated period of roughly 22 frames is unreliable, since the FFT lacks a dominant low-frequency peak with multiple competing modes, and the $\pm 1\sigma$ bandwidth exceeds the oscillation amplitude of the mean curve, reflecting heterogeneous animation styles.

As camera distance increases, the temporal signal progressively degrades. At close range (2r–8r), per-category signatures are clearly distinguishable; at medium range (32r–64r), the Chamfer-distance amplitudes attenuate but category-level separation remains visible; beyond roughly 100r, the curves flatten to near-constant values and temporal structure becomes unrecoverable. Birds are entirely absent from the dataset at 280r, where the few remaining assets (n=5–6 per class) give only small-sample, unreliable estimates. This distance-dependent signal loss complements the reconstruction-quality degradation measured in the distance-degradation sweep (§6.4.3) and establishes a practical operating-range ceiling for any downstream temporal classifier. The full ten-distance sweep is reported in Appendix B.3.

The per-category periodicity motivates the temporal-context sweep (§6.5.8): a classifier ingesting fewer frames than a class's cycle period under-samples its motion signature, a risk that is greatest for the longer-period classes (birds near 12 frames, helicopters near 16). The distance-dependent signal loss further motivates the Mamba [20] classifier over single-frame baselines, since extended temporal context can offset weaker per-frame signal at longer ranges. Per-class scale, quaternion, opacity, and color distributions are reported in Appendix B.

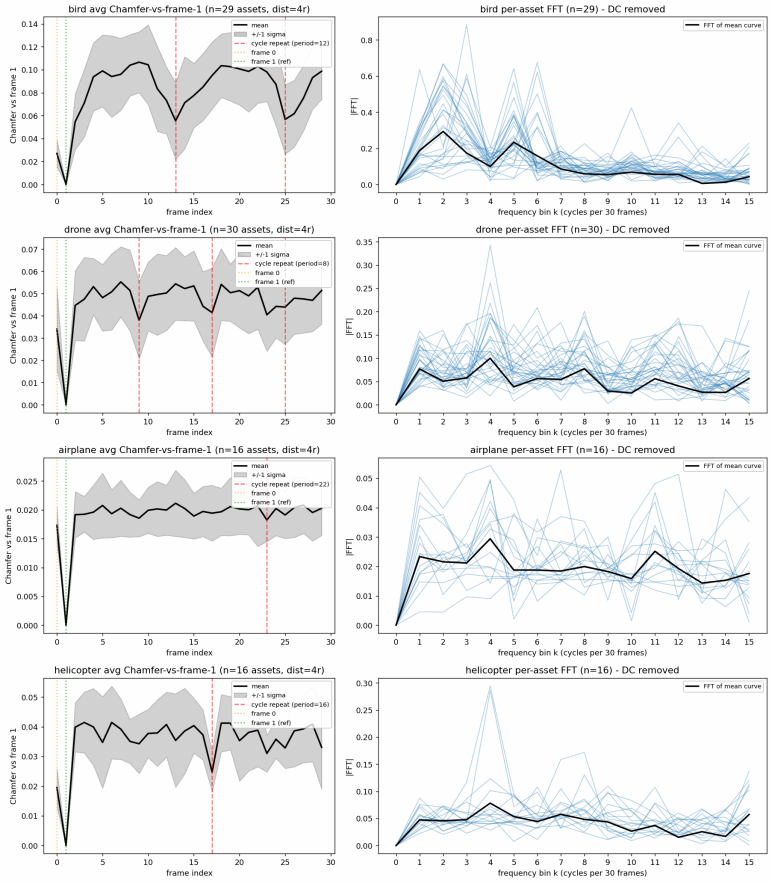


Figure 6.7: Per-category mean Chamfer distance to frame 1 (left) and per-asset FFT magnitude (right) at camera distance 4r. Each row is one category; the shaded band is $\pm 1\sigma$ across assets; dashed red lines mark the detected cycle-repeat period. FFT is computed on DC-removed 30-sample curves (16 frequency bins). Asset counts: bird n=29, drone n=30, airplane n=16, helicopter n=16. The remaining nine distances appear in Appendix B.3.

6.4 RQ2 – 3DGS Reconstruction (Stage 2)

We evaluate two feed-forward 3DGS reconstructors (LGM [69] and our `DepthSplat-OC`) on two standard object-centric benchmarks, GSO and ABO. LGM is the baseline; `DepthSplat-OC` is the deployed Stage-2 model used for the Sim-1, Sim-2, and Sim-3 reconstructions behind every downstream Stage-3 and Stage-4 result. Reconstruction quality is first benchmarked at standard close range (§6.4.1, §6.4.2), an ablation isolates the design choices of `DepthSplat-OC` (§6.4.1), and a distance-degradation sweep on the `AeroSplat-4D` Sim-1 corpus stresses small-object performance (§6.4.3); the textureless-sky aerial regime is exercised by that sweep and by the downstream Stage-3 and Stage-4 evaluations on Sim-1.

6.4.1 DepthSplat-OC Component Ablation

This experiment ablates four `DepthSplat-OC` configurations on GSO and ABO. The full run uses ground-truth alpha as encoder mask conditioning; `DepthSplat-OC` derives from the DepthSplat feed-forward reconstructor [63], and the depth-backbone tensors inside the checkpoint were initialized from a stock Depth Anything v2 ViT-S [66] and then jointly trained for 300k steps alongside the cost volume and the Gaussian head. The three ablation rows individually disable mask conditioning and the joint training of the depth backbone. This is achieved by overwriting its tensors with the raw stock Depth Anything v2 weights at inference, and doing so for both at once.

Mask conditioning is the dominant single factor: relative to the full model, it accounts for +2.50 dB PSNR on GSO and +2.90 dB on ABO, and removing it also costs roughly 0.07 SSIM and nearly doubles LPIPS. Reverting the depth backbone to the raw stock Depth Anything v2 weights costs a further +2.09 dB on GSO and +1.68 dB on ABO. This is evidence that the cost volume and the Gaussian head are tightly coupled to

the backbone features they were trained against, rather than evidence that the backbone could be improved by simply swapping in a different depth checkpoint. The two effects are sub-additive: dropping both the mask and the joint backbone training costs +3.09 dB on GSO and +3.53 dB on ABO, less than the sum of the individual deltas, indicating that the two factors partially substitute for one another. The GSO and ABO orderings agree across all four configurations.

Table 6.5: `DepthSplat-OC` component ablation on GSO and ABO, under the same four-context-view / ten-Fibonacci-target protocol as Table 6.6. Mask conditioning and the DPT monodepth branch are described in §5.5.1. "Full model" is the deployed `objaverse_white` checkpoint as-is; the "stock DAv2 backbone" rows revert the depth-backbone tensors to the raw stock weights at inference, ablating the joint backbone training rather than substituting a separately fine-tuned checkpoint (§8.2).

	GSO			ABO		
Configuration	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Full model (deployed)	**23.21**	**0.867**	**0.122**	**23.53**	**0.872**	**0.145**
w/o mask (full-frame input)	20.71	0.798	0.203	20.63	0.796	0.233
stock DAv2 backbone (joint training ablated, GT mask)	21.12	0.812	0.193	21.85	0.835	0.201
stock DAv2 backbone + no mask (lower bound)	20.12	0.786	0.224	20.00	0.787	0.257

6.4.2 Architecture Comparison (Standard Range)

This experiment compares reconstructors at the standard range, reporting PSNR, SSIM, and LPIPS for GSO and ABO at 256^2 with four cardinal context views at 20° elevation and ten Fibonacci target views, following the protocol of §5.5.2. Table 6.6 includes only baselines evaluated at 256^2; results reported at 512^2 by LGM [69] (23.79 PSNR on 100 GSO objects), M-LRM [155] (25.23 PSNR), and GRM [122] (30.05 PSNR) are not directly comparable due to the 4× pixel-count advantage and differing evaluation subsets, and are therefore omitted from the table. The aerial, textureless-sky stress test on Sim-1 is deferred to §6.4.3.

`DepthSplat-OC` was trained with five context views (§5.5.1); the four-view evaluation row is therefore slightly out-of-distribution and may undercount the model's capability. At four context views, it reaches 23.21 PSNR on GSO and 23.53 on ABO, already improving on the closest-protocol LGM baseline [69] on PSNR and SSIM, with LPIPS comparable on GSO and better on ABO. At the training-matched five-view operating point, performance rises to 24.65 on GSO and 24.50 on ABO. This is a further ∼1.3 dB gain. Figure 6.8 sweeps the full camera-count range on GSO and ABO: PSNR and SSIM climb steeply from two to five views while LPIPS falls, then all three metrics flatten beyond five, so we notice diminishing returns at six and eight views. This is likely because the model is trained only on 5 context views. A varying context view input at training should be performed in future work 8.2.

GS-LRM [70], the only 256^2 baseline with a matched evaluation protocol, posts substantially higher numbers (29.59 PSNR on GSO). However, this gap should be read alongside the training-compute comparison in Table 6.7: GS-LRM's 256^2 model was trained for ∼3,072 GPU-hours on 64 A100 GPUs (its 256-resolution pretraining stage) with ∼730 K Objaverse objects and a 300 M-parameter transformer, whereas `DepthSplat-OC` was trained for ∼27 GPU-hours on a single RTX 5090 with ∼122 K objects: roughly two orders of magnitude less compute. The present result is therefore best read as a proof-of-concept: a lightweight, single-GPU reconstructor that already surpasses LGM and serves the downstream classification pipeline of this thesis, rather than a claim of state-of-the-art reconstruction quality. Future work should scale the training regime (multi-GPU runs, longer schedules, and dedicated checkpoints for different context-view counts) to close the gap with higher-compute baselines. Baseline numbers in Table 6.6 are quoted from the cited sources under each source's own protocol; re-evaluation under our exact protocol was not feasible for this thesis, and minor protocol deviations are documented in the footnotes.

Table 6.6: Architecture comparison at standard range (256^2 input). GSO has 1030 scenes; ABO has 1000 scenes (the subset from `abo_eval_1000.txt`). Only baselines evaluated at 256^2 are included. Method details are in §5.5.1 and the evaluation harness in §5.5.4. Baseline numbers are quoted from the cited sources under that source's protocol; deviations are noted in the footnotes. See Table 6.7 for the training-compute comparison.

			GSO			ABO		
Model	In Res	Ctx	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LGM [69][a]	256^2	4	21.44	0.832	0.122	20.79	0.813	0.158
GS-LRM [70][b]	256^2	4	**29.59**	**0.944**	**0.051**	**28.98**	**0.926**	**0.074**
DepthSplat-OC[c]	256^2	4	23.21	0.867	0.122	23.53	0.872	0.145
	256^2	4	23.89	0.878	0.110	23.92	0.880	0.137
	256^2	5	24.65	0.890	0.100	24.50	0.889	0.129
	256^2	6	24.76	0.891	0.100	24.60	0.890	0.129
	256^2	8	24.72	0.891	0.100	24.49	0.889	0.130

[a] LGM numbers from GS-LRM [70], Tab. 1, evaluated at 256^2 with four input views; the protocol closest to ours.
[b] GS-LRM [70], Tab. 1, object-level model: 256^2, four input views.
[c] Evaluated under the same protocol as Table 6.5 (10 Fibonacci targets); the context-sweep rows below use 6 fixed targets, which accounts for the ∼0.7 dB PSNR difference at matched context-view count. Trained with five context views; the four-view row is slightly out-of-distribution.

Table 6.7: Training-compute comparison for the 256^2 baselines in Table 6.6. GPU-hours are computed as (number of GPUs)×(wall-clock training time). LGM numbers from [69]; GS-LRM numbers from [70]; the GS-LRM figure is the cost of its 256^2 pretraining stage, which produces the 256^2 checkpoint compared in Table 6.6, while the released model adds one further day of 512^2 fine-tuning; `DepthSplat-OC` details in §5.5.1.

	LGM	GS-LRM	**DepthSplat-OC**
GPU	32× A100 80 GB	64× A100 40 GB	1× RTX 5090 32 GB
CUDA cores / GPU	6,912	6,912	21,760
Total CUDA cores	221,184	442,368	21,760
Aggregate VRAM	2,560 GB	2,560 GB	32 GB
Wall-clock training	∼4 days	∼2 days	∼27 h
GPU-hours	∼3,072	∼3,072	∼27
Training objects	∼80 K	∼730 K	∼122 K

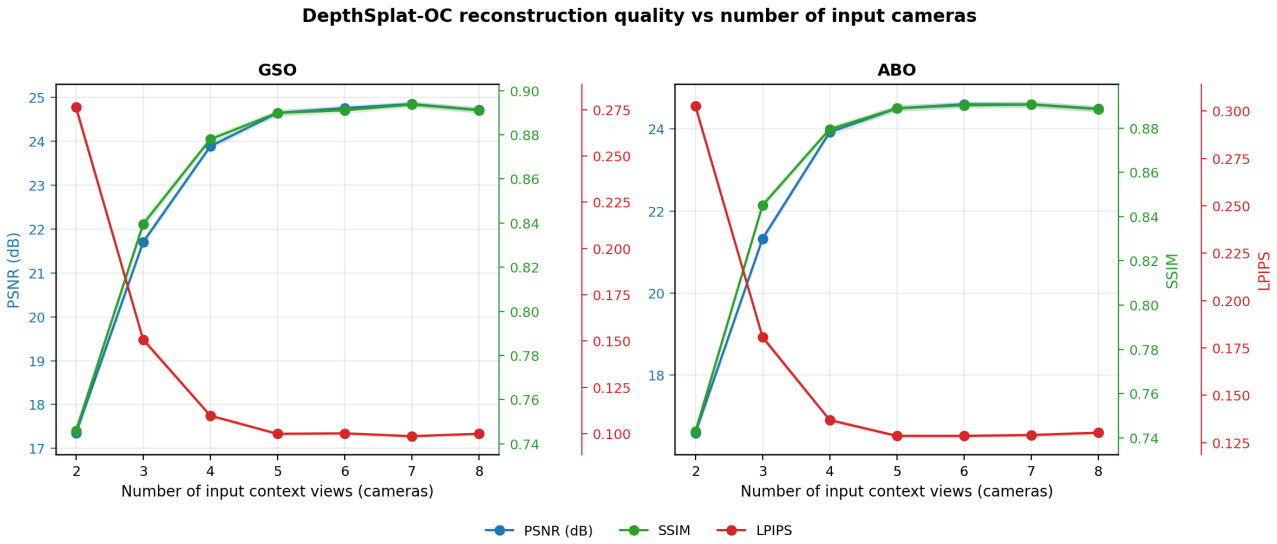


Figure 6.8: `DepthSplat-OC` reconstruction quality versus number of input context views on GSO (left, 1030 scenes) and ABO (right, 1000 scenes). Each panel plots PSNR↑ (blue, left spine), SSIM↑ (green, right spine), and LPIPS↓ (red, second offset right spine) for $N \in \{2, 3, 4, 5, 6, 7, 8\}$ context views, with ±1 SEM bands. PSNR and SSIM climb steeply from N=2 to N=5 while LPIPS falls, then all three saturate for $N \geq 5$. The model accepts an arbitrary number of input cameras without retraining; it was trained with N=5, so N=2 and N=3 are out-of-distribution and lose roughly 7–8 dB PSNR. Targets are the six fixed Fibonacci views of the context-sweep protocol, so absolute scores align with the context-sweep rows of Table 6.6 rather than the ten-target headline row.

6.4.3 Distance Degradation on AeroSplat-4D

This experiment runs a controlled-distance sweep on Sim-1 to stress the small-object, textureless-sky regime that GSO and ABO cannot handle. The goal is to characterize per-view `DepthSplat-OC` reconstruction quality as a function of camera distance, under the evaluation protocol of §5.5.2. The dataset is the Sim-1 distance sweep rendered in IsaacSim, comprising 24 curated assets (8 birds, 8 drones, 4 airplanes, and 4 helicopters) at elevations of ±20° and a fixed 45° azimuth, over frames 10–19 (ten consecutive frames per sequence). The sweep covers ten distances, $\{2, 4, 8, 16, 32, 64, 100, 140, 200, 280\}$ m, matched to the dyadic-octave bins of §6.3.2.

For each (sequence, frame, view) tuple, the `DepthSplat-OC` PLY Gaussians are loaded in the kiui Y-up frame and the four input cameras are re-rendered at 256^2 with the Gaussian renderer, then compared against ground-truth tight crops to obtain PSNR, SSIM, and LPIPS-VGG. `DepthSplat-OC` uses the parity-patched tight crop at a 1.5 m camera radius (§5.5.2). Each (distance) cell aggregates roughly 7360 evaluations (480 sequences × 10 frames × 4 views) with minor frame dropout at 200 and 280 m.

LPIPS rises monotonically with distance beyond roughly 32 m. PSNR and SSIM, by contrast, are approximately flat up to 64 m and then weakly *increase* at 200–280 m as the sky fraction of the crop grows and dominates the pixel-wise scores. This is behavior that should be read together with the pixel-area histogram (Figure 6.2). This apparent uplift is a degenerate-crop artifact rather than a genuine quality gain: at long range the object shrinks to a few pixels of ground-truth signal, the upscaled 256^2 crop becomes nearly uniform sky, and the reconstruction collapses to a near-white field. Because both the ground truth and the prediction approach a constant background, PSNR and SSIM saturate at high values, while LPIPS, being perceptual and content-sensitive, still captures structural loss. Points at ≥ 140 m are therefore shown in gray in Figure 6.9 and should be interpreted with caution.

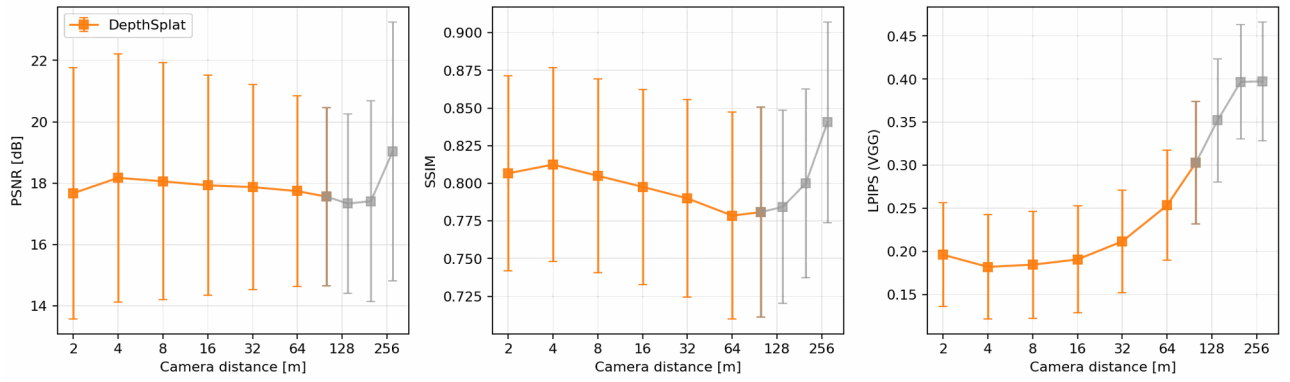


Figure 6.9: `AeroSplat-4D` Sim-1 distance sweep, frames 10–19. Per-view PSNR↑, SSIM↑, and LPIPS↓ versus camera distance for `DepthSplat-OC`. Markers give the mean across all (sequence, frame, view) tuples per cell (∼7360 evaluations); error bars give ±1 std. Points at ≥ 140 m (gray) are dominated by the degenerate-crop artifact discussed in the text.

Table 6.8: `DepthSplat-OC` reconstruction quality versus distance on `AeroSplat-4D` Sim-1 (frames 10–19). Each cell is the mean over roughly 7360 (sequence, frame, view) tuples; std and intermediate distances are shown in Figure 6.9. Evaluation protocol: §5.5.2.

Distance	PSNR↑	SSIM↑	LPIPS↓
2 m	17.67	0.807	0.196
16 m	17.93	0.798	0.191
64 m	17.74	0.779	0.253
140 m†	17.34	0.784	0.352
280 m†	19.04	0.841	0.397

†Degenerate-crop regime; see text.

6.4.4 DepthSplat-OC Preprocessing

Before they can be consumed by the classifier, the Stage-2 3DGS reconstructions are preprocessed. A full 3DGS reconstruction contains far too many Gaussians to feed directly into the `MambaSplat-4D` pipeline, so we apply the same decimation procedure used by ShapeSplat [96], described in §5.6.10. Figure 6.10 illustrates the effect of this preprocessing on the toucan test object.

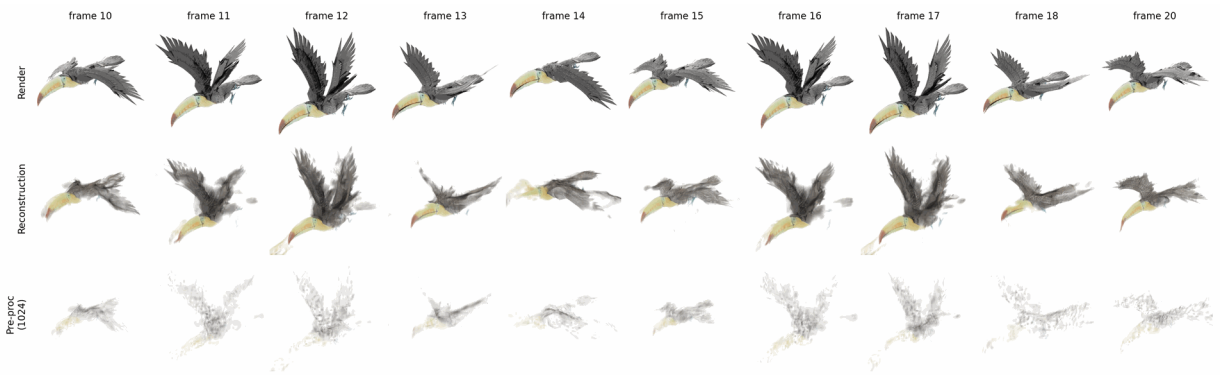


Figure 6.10: `AeroSplat-4D` pre-processing pipeline applied to the toucan test object, one column per frame. Top row: the ground-truth render. Middle row: the raw `DepthSplat-OC` 3DGS reconstruction. Bottom row: the matching preprocessed 1024-point tensor. The reconstruction and preprocessed rows are rendered through the differentiable Gaussian rasterizer.

6.5 RQ3 – 4D Classification (Stage 3)

RQ3 is the core contribution of this thesis: the eight experiments below exercise the design choices behind `MambaSplat-4D`, and a final diagnostic (§6.5.9) verifies its rotation-invariance guarantee end-to-end. Across these experiments, we vary the Gaussian feature mode, the rotation protocol, the invariant bridge, the individual pipeline components, the Mamba hidden dimension d_{mamba} and depth n_{layers}, and the amount of temporal context.

6.5.1 Gaussian-Attribute Ablation

This experiment ablates which Gaussian attributes carry discriminative signal, crossing the six feature modes (Table 5.3) with the three rotation protocols (Table 5.6) on ModelSplat-10 and ModelSplat-40 [132]. We compare our VN-In bridge (§5.6.6) against the VNStdFeature bridge (§5.6.11), with VN-DGCNN and VN-PointNet [29] at $E(C)$.

Centroids alone break the VN-In bridge: $E(C)$ collapses to 42.8% on ModelSplat-10 under z/z, because non-uniform centroid distributions render the learned frame rank-deficient. Adding spherical-harmonic color is the largest single-attribute gain, lifting z/z accuracy on ModelSplat-10 by 47 percentage points ($E(C) \rightarrow E(C, \text{SH})$, 42.8 to 89.8). VN-In $E(X)$ is its bridge's best at 90.4% on ModelSplat-10 and 86.8% on ModelSplat-40. VNStdFeature is far less mode-sensitive, already exceeding 88% at $E(C)$ and peaking at 92.6%. VN-DGCNN ($E(C)$, 92.3%) and VN-PointNet stay competitive on ModelSplat-10.

Table 6.9: Gaussian-attribute ablation on ModelSplat-10 / ModelSplat-40. Overall accuracy (%) ↑ across three rotation protocols (§5.6.12, Table 5.6). Feature modes follow Table 5.3. Bridges: VN-In (§5.6.6) and VNStdFeature (§5.6.11). Within each bridge block (VN-In and VNStdFeature), bold marks the best value in each column.

Bridge	Feature Set	MN10			MN40		
		z/z	z/SO(3)	SO(3)/SO(3)	z/z	z/SO(3)	SO(3)/SO(3)
VN-In (ours)	$E(C)$	42.8	43.3	51.5	73.4	73.1	60.2
	$E(C, O)$	73.5	73.8	86.5	80.1	80.6	80.5
	$E(C, \text{SH})$	89.8	90.1	88.4	85.5	85.7	85.5
	$E(C, S, R)$	77.9	77.6	74.3	72.3	71.8	71.3
	$E(O, C, S, R)$	87.7	87.6	86.2	85.5	85.6	85.9
	$E(X)$	**90.4**	**90.3**	**91.0**	**86.8**	**86.5**	**86.3**
VNStdFeature	$E(C)$	88.8	88.8	87.8	83.7	83.7	85.1
	$E(C, O)$	89.3	89.3	88.9	85.5	85.5	85.9
	$E(C, \text{SH})$	**92.6**	**92.6**	**92.0**	88.1	88.1	88.6
	$E(C, S, R)$	92.2	92.2	78.3	88.1	88.1	89.2
	$E(O, C, S, R)$	92.2	92.2	92.0	89.3	89.3	**89.3**
	$E(X)$	91.9	91.9	91.3	**89.4**	**89.4**	88.9
VN-DGCNN [29]	$E(C)$	92.3	92.3	91.6	88.1	88.1	88.4
VN-PointNet [29]	$E(C)$	86.5	86.7	89.0	79.9	80.0	81.5

6.5.2 Temporal Rotation Protocol (MSR-Action3D)

This experiment evaluates the four rotation protocols (Table 5.6) on MSR-Action3D [133], comparing `MambaSplat-4D` $E(C)$ against P4Transformer [135] and Mamba4D [104]. Both baselines collapse under the per-frame protocol $z/\text{SO}(3)_{\text{pf}}$, dropping to 5.2% and 10.3% (essentially chance). `MambaSplat-4D` stays robust at 60.2 / 60.1 / 60.1 / 66.9, its near-identical z/z and $z/\text{SO}(3)_{\text{pf}}$ values confirming per-frame rotation invariance; the higher SO(3)/SO(3) value reflects the heavier SO(3) training augmentation rather than any eval-time effect. MSR-Action3D carries point coordinates only, with no Gaussian attributes, so `MambaSplat-4D` runs in its positions-only $E(C)$ mode (Table 5.3); the absent opacity, color, scale, and quaternion channels cap its peak accuracy below the attribute-rich `AeroSplat-4D` experiments. Its absolute accuracy is therefore not directly comparable to those experiments, which also differ in task, training recipe, and encoder width. The within-table rotation-protocol contrast remains valid, since all three models share one evaluation protocol.

Table 6.10: Rotation protocol on MSR-Action3D. Mean accuracy (%) over 10 rotation trials per cell. Protocols follow Table 5.6, §5.6.12. Bold marks our method (`MambaSplat-4D` $E(C)$), not the per-column best.

Model	$z/\mathrm{SO}(3)_{\mathrm{pf}}$	z/z	$z/\mathrm{SO}(3)$	$\mathrm{SO}(3)/\mathrm{SO}(3)$
P4Transformer [135]	5.2	75.3	15.3	77.7
Mamba4D [104]	10.3	81.2	13.7	86.8
MambaSplat-4D $E(C)$	**60.2**	**60.1**	**60.1**	**66.9**

6.5.3 4-Class AeroSplat-4D

This experiment runs four-class temporal classification on `AeroSplat-4D` ($T = 24$, Sim-1 distance-sweep corpus, held-out test split). The P4Transformer and Mamba4D baselines consume xyz only; we report `MambaSplat-4D` at $E(C)$ and $E(X)$ (Table 5.3), over three seeds and three rotation trials per cell.

`MambaSplat-4D` $E(X)$ is the most robust model, holding 70–71% across every rotation protocol with std ≤ 1.8, and its z- and SO(3)-trained results coincide. The full attribute set carries real signal: $E(X)$ beats $E(C)$ by about 8 percentage points across all protocols. The baselines collapse under rotation: Mamba4D wins the clean z/z cell at 74.9% but loses 35–37 points moving to $z/\mathrm{SO}(3)$ or $z/\mathrm{SO}(3)_{\mathrm{pf}}$, and P4Transformer never recovers above 36.5% once rotation enters. SO(3) augmentation recovers some baseline robustness, Mamba4D rising to 58.2% on SO(3)/SO(3), but it still trails $E(X)$ by roughly 13 points there. Per class under SO(3)/SO(3), helicopter is hardest for the baselines (0.13 for P4Transformer, 0.30 for Mamba4D); $E(X)$ recovers it to 0.63 and stays the most balanced model across all four classes.

Table 6.11: Four-class `AeroSplat-4D`, `DepthSplat-OC` 3DGS (Sim-1 distance sweep, held-out test split, $T = 24$). Mean accuracy (%) $\pm$ std over 3 seeds $\times$ 3 rotation trials. Per-class accuracy is reported at SO(3)/SO(3), averaged across seeds and trials. Training recipe: §5.6.11, Table 5.5. Bold marks our method (`MambaSplat-4D` $E(X)$), not the per-column best.

Model	$z/\mathrm{SO}(3)_{\mathrm{pf}}$	z/z	$z/\mathrm{SO}(3)$	$\mathrm{SO}(3)/\mathrm{SO}(3)$	Acc Bird	Acc Drone	Acc Plane	Acc Heli
P4Transformer [135]	32.6 ± 0.5	59.8 ± 4.9	36.3 ± 1.5	36.5 ± 5.2	0.52	0.37	0.35	0.13
Mamba4D [104]	38.2 ± 0.5	74.9 ± 1.1	39.5 ± 2.1	58.2 ± 4.8	0.70	0.59	0.66	0.30
MambaSplat-4D $E(C)$	61.6 ± 6.0	61.5 ± 5.9	61.6 ± 6.2	63.7 ± 1.2	0.78	0.69	0.42	0.50
MambaSplat-4D $E(X)$	**70.2 ± 1.5**	**70.2 ± 1.5**	**70.2 ± 1.5**	**71.2 ± 1.8**	**0.85**	**0.71**	**0.57**	**0.63**

6.5.4 Bridge-Method Ablation

This experiment swaps six bridge variants inside the `vn_mamba` pipeline while holding the spatial encoder, the Mamba SSM, and the classifier head fixed. Each bridge maps the VN feature tensor $V \in \mathbb{R}^{B\times C\times 3}$ to an SO(3)-invariant scalar tensor that the temporal Mamba then consumes. We compare VN-In (ours) against four fixed-form invariants defined in §5.6.11: L2-norm pooling (C invariants), Gram dot-products ($C(C{+}1)/2$), pairwise channel distances ($C(C{-}1)/2$), and a PCA canonical frame ($3C{-}3$), plus the VNStdFeature head (§5.6.11), which applies a *learned* per-point canonical frame before pooling. VN-In (§5.6.6) likewise realizes the full $3C{-}3$ invariant subspace, but through a *learned*, data-adapted frame rather than a fixed eigendecomposition; all six satisfy the invariance identity of Eq. 5.13.

All six variants train *without* rotation augmentation on Sim-1 and are evaluated under the none, z, SO(3), and per-frame SO(3) protocols, so the bridge alone must absorb test-time rotations. Because every bridge is rotation-invariant by construction, the four rotation columns are identical per variant, and Table 6.12 reports a single column per split. VN-In is the strongest at 80.0%/79.8% (val/test). The decisive comparison is the PCA frame at 64.6%/63.6%: it shares VN-In's exact $3C{-}3$ output dimension and algebraic shape, so the 15.4-point gap isolates the value of a *learned* frame over a fixed eigendecomposition. The lower-dimensional invariants (Gram 66.6%, Distance 65.8%, L2-norm 63.9%) discard relative-orientation information and trail further. The VNStdFeature head (71.0%/71.5%) closes part of this gap with its learned per-point frame but remains 9.0 points below VN-In at more than twice the parameter count (3.90M versus 1.88M). Both the full invariant dimensionality and a learned frame thus contribute, and a fixed frame can squander the extra dimensions, as PCA's slight deficit to the pairwise variants shows.

Table 6.12: Bridge-method ablation on `AeroSplat-4D` (`DepthSplat-OC` 3DGS, Sim-1 distance sweep, $T = 24$). All variants are trained *without* rotation augmentation, with 3 seeds. Video-level accuracy (%) ↑ on the validation and test splits, mean ± std over 3 rotation trials; the value is identical across rotation protocols, since every bridge is rotation-invariant by construction, so a single column is shown per split. Bridge definitions: §5.6.6.

Bridge Method	Invariants	val ↑	test ↑	#Params
L2-norm pooling	C of $3C-3$ max	63.9 ± 2.5	62.9 ± 3.1	1.87M
Gram (pairwise dot)	$\mathcal{O}(C^2)$ pairwise	66.6 ± 0.8	65.7 ± 4.1	1.90M
Pairwise distances	$\mathcal{O}(C^2)$ pairwise	65.8 ± 6.2	62.8 ± 8.9	1.89M
PCA canonical frame	canonicalised $3C-3$	64.6 ± 2.7	63.6 ± 2.3	1.88M
VNStdFeature	$3C$ (per-point learned frame)	71.0 ± 3.5	71.5 ± 1.9	3.90M
VN-In Bridge (ours)	$3C-3$ **(full, learned)**	**80.0 ± 2.4**	**79.8 ± 3.7**	**1.88M**

6.5.5 Component Ablation

This experiment removes or swaps one component at a time from the full `MambaSplat-4D` pipeline (the `full_X` variant), trained without rotation augmentation on Sim-1. The variants fall into three groups, each isolating one architectural claim:

- **Features (5 variants):** $E(C)$, $E(C, O)$, $E(C, \mathrm{SH})$, $E(C, S, R)$, and $E(C, O, S, R)$, isolating how much signal lives in color versus opacity, spherical harmonics, scale, and rotation (lifted-channel construction in §5.6.6).
- **Spatial (1 variant):** *w/o translation invariance* (`spatial.translation_invariant=false`), isolating the contribution of translation-invariant spatial features while leaving rotation invariance intact, the latter conferred downstream by the VN-In bridge.
- **Temporal (4 variants):** *w/o temporal (last frame)*, *w/o Mamba SSM (bridge + mean pool)*, *w/o Mamba (bridge + last frame)*, and *w/o positional encoding*, isolating the temporal aggregator and its positional encoding.

Every variant retained in Table 6.13 preserves the SO(3)-invariant VN-In bridge, so its four rotation columns are numerically identical up to evaluation noise, a built-in sanity check that confirms the invariance claim of `full_X`, including the *w/o translation invariance* row, whose rotation invariance survives the loss of translation invariance precisely because it is conferred by the bridge rather than by the spatial coordinates. The non-equivariant scalar-Mamba configuration that would break this is confounded with the temporal architecture and parameter budget, and is therefore examined separately in §6.5.4 rather than here.

Before presenting the component ablation, we justify the temporal readout aggregator held fixed across this component ablation and the capacity and context sweeps that follow: mean pooling over the Mamba output sequence rather than last-token pooling (distinct from the *w/o temporal* spatial-only row, which removes the SSM entirely). Our prior intuition favored last-token pooling, since the Mamba SSM is sequential and its final hidden state has nominally already integrated every earlier frame. The embedding geometry in Figure 6.11 appears to confirm this: last-token pooling drives a larger intra-minus-inter cosine gap (0.612 ± 0.153 versus 0.247 ± 0.051), greater centroid L_2 separation, and a more negative inter-class similarity. The validation metrics in Figure 6.12 contradict that reading: although both variants reach near-perfect training accuracy, last-token pooling's validation loss turns upward after roughly epoch 10 while its accuracy plateaus lower (0.741 ± 0.061) and far noisier than mean pooling (0.791 ± 0.024), and its final validation macro F_1 trails (0.711 ± 0.070 versus 0.770 ± 0.020). The inflated embedding gap is therefore memorization rather than generalization: concentrating the readout on a single timestep lets the final state overfit, whereas averaging over all timesteps acts as a regularizer, yielding a smaller but honest gap, lower and more stable validation loss, and higher accuracy at roughly a third of the seed variance. We accordingly adopt mean temporal pooling as the default aggregator in all subsequent tables, despite its contradicting our initial intuition.

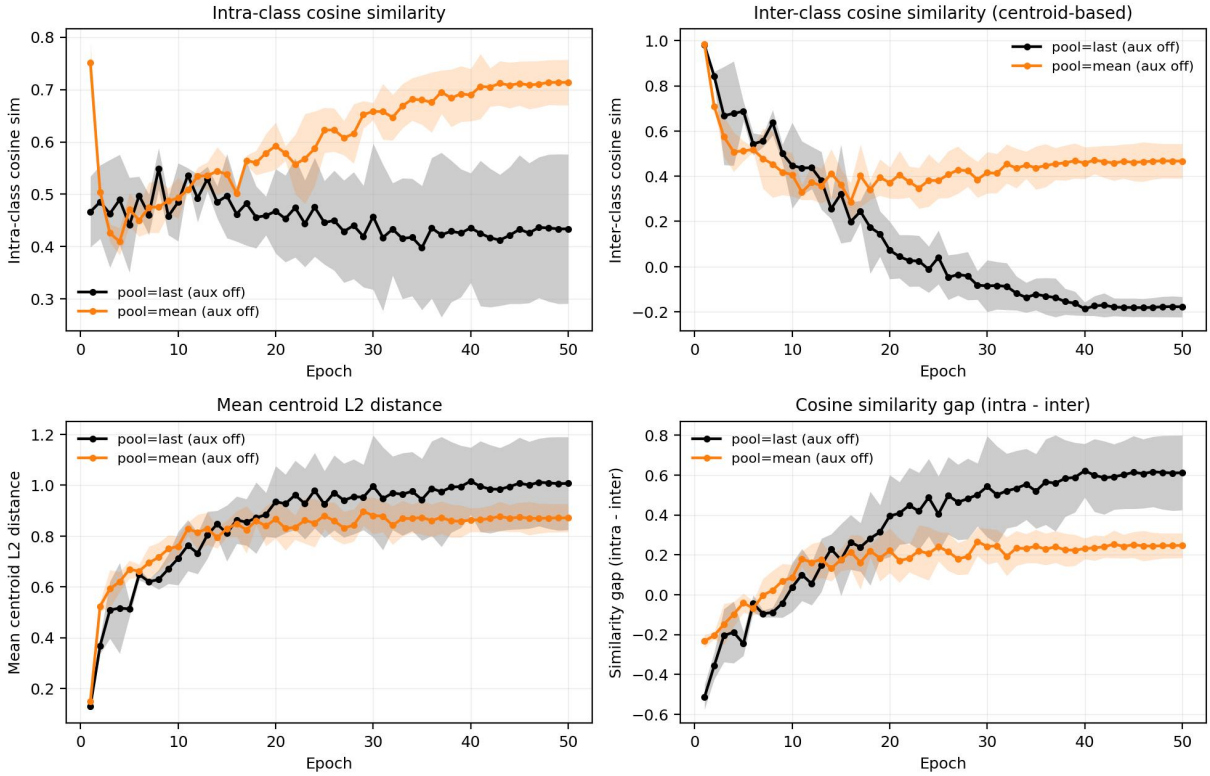


Figure 6.11: Embedding quality over 50 epochs for last-token versus mean temporal readout pooling over the Mamba output sequence (auxiliary loss off; 3-seed mean $\pm 1\sigma$). Panels: intra-class cosine similarity, centroid-based inter-class cosine similarity, mean centroid L_2 distance, and the intra-minus-inter cosine gap. Last-token pooling produces the larger geometric separation (final gap 0.612 ± 0.153 versus 0.247 ± 0.051), as shown in Figure 6.12, but this does not generalize.

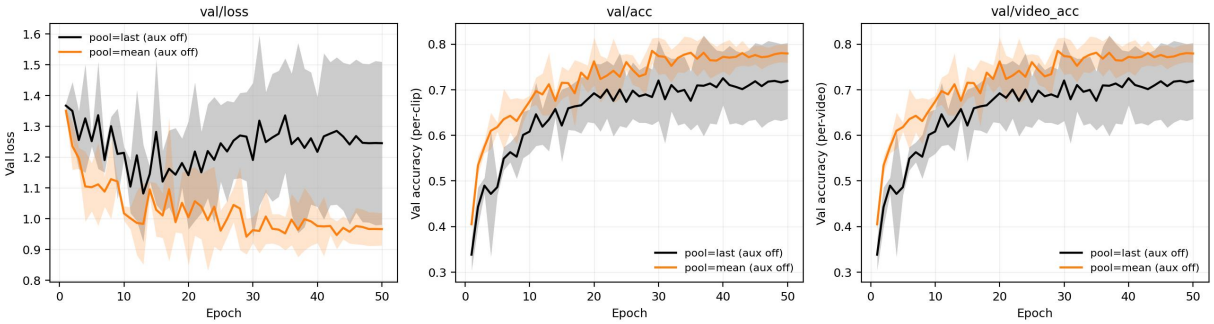


Figure 6.12: Validation core metrics over 50 epochs for the same two readout-pooling variants (3-seed mean $\pm 1\sigma$): validation loss, per-clip accuracy, and per-video accuracy. Last-token pooling's validation loss rises after the early epochs (overfitting) and its accuracy plateaus at a lower level (0.741 ± 0.061) with markedly higher seed-to-seed variance than mean pooling (0.791 ± 0.024).

Table 6.13: `MambaSplat-4D` component ablation on `AeroSplat-4D` (`DepthSplat-OC` 3DGS, Sim-1 distance sweep, $T = 24$). All variants are trained *without* rotation augmentation, with 3 seeds (42, 43, 44) × 1 rotation trial. Video-level accuracy (%) ↑ on the validation and test splits under four rotation protocols; mean ± std across seeds. All variants here retain the SO(3)-invariant VN-In bridge, so every row produces identical numbers across the four rotation columns by construction, providing a sanity check on the invariance claim. Training recipe: §5.6.11, Table 5.5.

		Val				Test				
Group	Configuration	none	z	SO(3)	$\mathrm{SO(3)_{pf}}$	none	z	SO(3)	$\mathrm{SO(3)_{pf}}$	#Params
Baseline	**Full MambaSplat-4D (X)**	$\mathbf{80.8 \pm 2.8}$	$\mathbf{80.8 \pm 2.8}$	$\mathbf{80.8 \pm 2.8}$	$\mathbf{80.8 \pm 2.8}$	$\mathbf{78.8 \pm 4.1}$	$\mathbf{78.8 \pm 4.1}$	$\mathbf{78.8 \pm 4.1}$	$\mathbf{78.8 \pm 4.1}$	**1.88M**
Features	$E(C)$	66.1 ± 7.2	66.1 ± 7.1	66.1 ± 7.2	66.0 ± 7.1	66.8 ± 7.8	67.1 ± 7.4	66.9 ± 7.6	66.7 ± 7.9	1.85M
Features	$E(C, O)$	59.7 ± 11.0	59.7 ± 11.0	59.7 ± 11.0	59.7 ± 11.0	61.3 ± 7.4	61.3 ± 7.3	61.3 ± 7.4	61.3 ± 7.4	1.86M
Features	$E(C, \mathrm{SH})$	56.4 ± 2.9	56.4 ± 2.9	56.4 ± 2.9	56.4 ± 2.9	44.2 ± 2.5	44.2 ± 2.5	44.2 ± 2.5	44.2 ± 2.5	1.87M
Features	$E(C, S, R)$	78.5 ± 3.4	78.5 ± 3.4	78.5 ± 3.4	78.5 ± 3.4	73.1 ± 3.7	73.1 ± 3.7	73.1 ± 3.7	73.1 ± 3.7	1.85M
Features	$E(C, O, S, R)$	75.1 ± 6.5	75.1 ± 6.5	75.1 ± 6.5	75.1 ± 6.5	77.5 ± 6.6	77.5 ± 6.6	77.5 ± 6.6	77.5 ± 6.6	1.86M
Spatial	w/o translation invariance	78.2 ± 1.6	78.2 ± 1.6	78.2 ± 1.6	78.2 ± 1.6	77.2 ± 0.4	77.2 ± 0.4	77.2 ± 0.4	77.2 ± 0.4	1.88M
Temporal	w/o temporal (last frame)	60.9 ± 1.0	60.9 ± 1.0	60.9 ± 1.0	60.9 ± 1.0	59.1 ± 0.6	59.1 ± 0.6	59.1 ± 0.6	59.1 ± 0.6	0.33M
Temporal	w/o Mamba SSM (bridge + mean pool)	78.4 ± 5.5	78.4 ± 5.5	78.4 ± 5.5	78.4 ± 5.5	77.3 ± 4.5	77.3 ± 4.5	77.3 ± 4.5	77.3 ± 4.5	0.13M
Temporal	w/o Mamba (bridge + last frame)	66.5 ± 5.5	66.5 ± 5.5	66.5 ± 5.5	66.5 ± 5.5	62.2 ± 2.8	62.2 ± 2.8	62.2 ± 2.8	62.2 ± 2.8	0.13M
Temporal	w/o positional encoding	77.8 ± 5.6	77.8 ± 5.6	77.8 ± 5.6	77.8 ± 5.6	74.3 ± 2.6	74.3 ± 2.6	74.3 ± 2.6	74.3 ± 2.6	1.88M

Figure 6.13 shows validation-accuracy convergence for the full pipeline and every ablation variant over 50 epochs. Most variants train to a stable plateau by roughly 30 to 40 epochs, confirming that the Table 6.13 comparison is between converged models rather than under-trained ones. The exception is the non-equivariant scalar-Mamba configuration (*w/o VN-In bridge*), which collapses to chance early in training; consistent with the note above, it is examined separately rather than tabulated here.

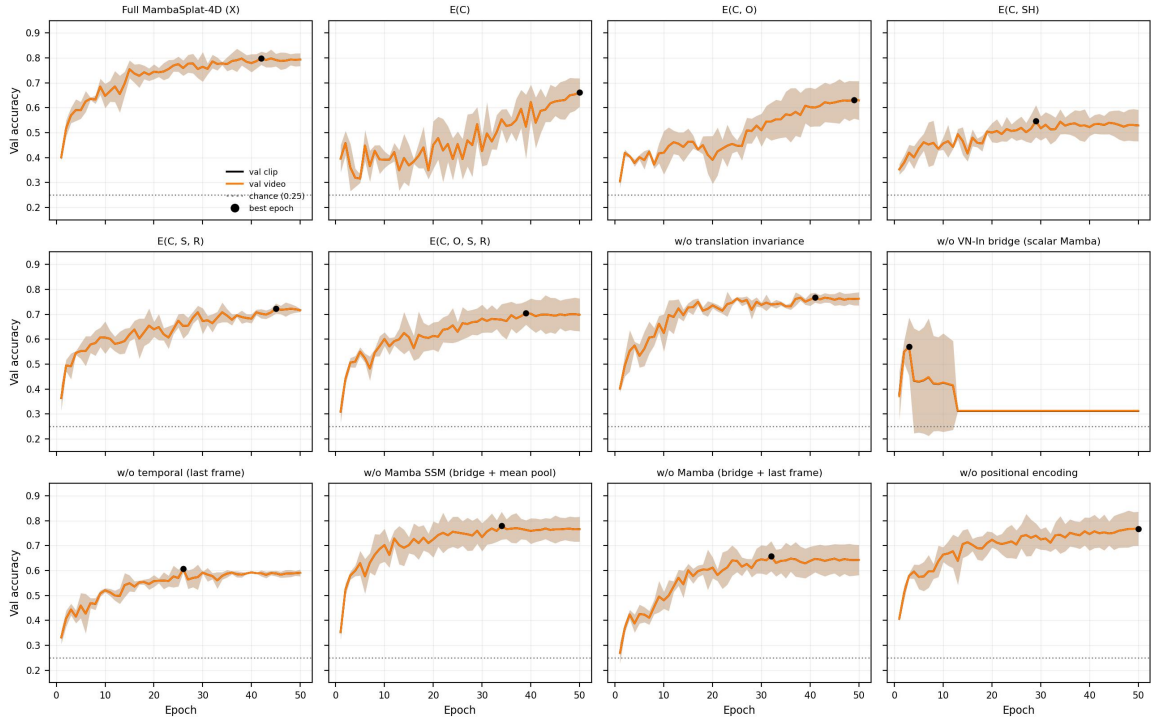


Figure 6.13: Validation-accuracy convergence over 50 epochs for the full `MambaSplat-4D` pipeline and its ablation variants on Sim-1, one panel per variant. Each panel plots clip-level and video-level validation accuracy (per-seed mean with shaded $\pm 1\sigma$ band, chance at 0.25), with the best epoch marked. Most variants converge and plateau by roughly 30 to 40 epochs; the scalar-Mamba variant (*w/o VN-In bridge*) collapses to chance early in training.

Unlike the other tabulated variants, the spatial-only *w/o temporal* baseline converges but then overfits. By epoch 50 it reaches 98.2 % training accuracy against only 59.1 % test accuracy, a train-to-test gap of nearly 40 points, roughly twice that of the full pipeline (100 % train, 78.8 % test) or the *w/o Mamba SSM* mean-pool

variant (100 % train, 77.3 % test). The signature is unambiguous in the loss: its validation loss bottoms near epoch 15 and then climbs by about 21.5 % while the training loss keeps falling, and its validation accuracy peaks around epoch 26 (the best epoch marked in Figure 6.13) before drifting down, whereas the two mean-pool variants show no comparable upturn. The cause is the readout rather than capacity: the identity head classifies from the most recent frame alone, discarding the averaging that the pooled variants apply over all T frames. That temporal mean acts as a variance-reducing regularizer, the same mechanism identified for last-token versus mean readout pooling above (Figures 6.11 and 6.12); without it, the head fits a single low-signal per-frame embedding well enough to memorize the training identities but not to generalize, the more so as no rotation augmentation is applied here.

Three of these variants, the full `full_X` pipeline, *w/o Mamba SSM* (VN-In bridge + mean pool), and *w/o temporal* (last frame), isolate the temporal stack and are analyzed qualitatively in Figures 6.14–6.19 on the test split, aggregated over seeds 42–44. The total temporal contribution is large: relative to the spatial-only *w/o temporal* baseline (59.1 % test), the full pipeline adds roughly 20 percentage points (78.8 % test). Almost all of this gain is already realized by the VN-In bridge with simple mean pooling, *w/o Mamba SSM* reaching 77.3 %, so the Mamba SSM itself adds only ∼1.5 points on clean reconstructions, within the seed-to-seed standard deviation (±4). The value of temporal context is therefore unambiguous, but on clean data the choice of temporal aggregator (Mamba SSM versus mean pool) is not separable from noise; the distance-resolved study below stresses exactly this distinction.

Figure 6.14 traces accuracy as the number of observed frames grows. The full pipeline and the mean-pool variant both climb from low single-frame accuracy to ∼78 % as context accumulates, while the spatial-only baseline, whose identity head classifies from the most recent frame alone (so its curve reflects per-frame spatial quality rather than accumulation), stays flat near 60 % regardless of clip length. The Mamba variant trails the mean-pool variant until roughly ten frames are observed, indicating that the SSM only repays its parameters once enough temporal context is available. At the shortest prefixes, the full pipeline even trails the static single-frame baseline (0.32 versus ∼0.60 at one frame), since the SSM has no temporal context to exploit yet, and overtakes both the baseline and the mean-pool variant only once roughly ten frames accumulate.

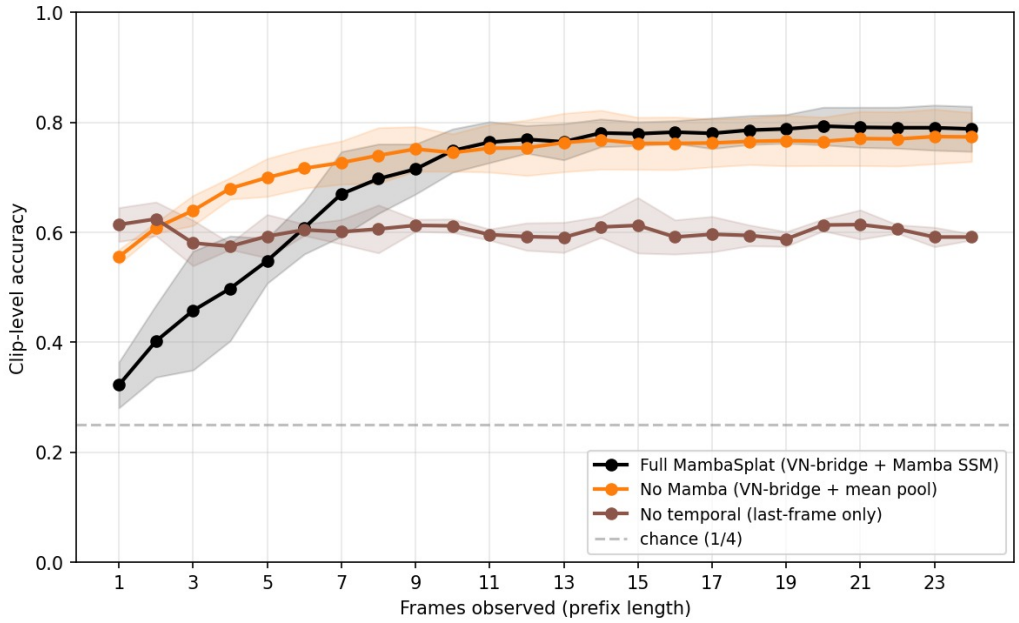


Figure 6.14: Clip-level accuracy versus the number of observed frames (prefix length 1 . . . 24) on the `AeroSplat-4D` Sim-1 test split, 3-seed mean $\pm 1\sigma$. The *w/o temporal* (last-frame) curve reflects per-frame spatial quality, not accuracy accumulating over time.

Figures 6.15 and 6.16 resolve the contribution by camera distance. Both temporal variants track each other closely and stay well above the single-frame baseline across the near-to-mid range, consistent with the per-frame signal degradation characterized in §6.3.5 and §6.4.3: at 140 m the full pipeline and mean-pool reach 0.75 and 0.72 against 0.54 for the single-frame baseline. Beyond roughly 140 m the temporal advantage shrinks rather than vanishing, and the ordering between the two aggregators inverts: at 200–280 m mean pooling matches or

slightly edges out the Mamba SSM (0.55 versus 0.48 at 200 m, 0.42 versus 0.36 at 280 m) while both still beat the single-frame baseline (0.32 and 0.28). This is the degraded-geometry counterpart to the clean-data finding above: on heavily degraded reconstructions, the SSM no longer recovers its parameters from a simple mean. The (frames × distance) difference panel makes the trade-off explicit: at very short prefixes the spatial-only baseline is competitive, but once context accumulates the temporal variants dominate at almost every distance up to the long-range floor.

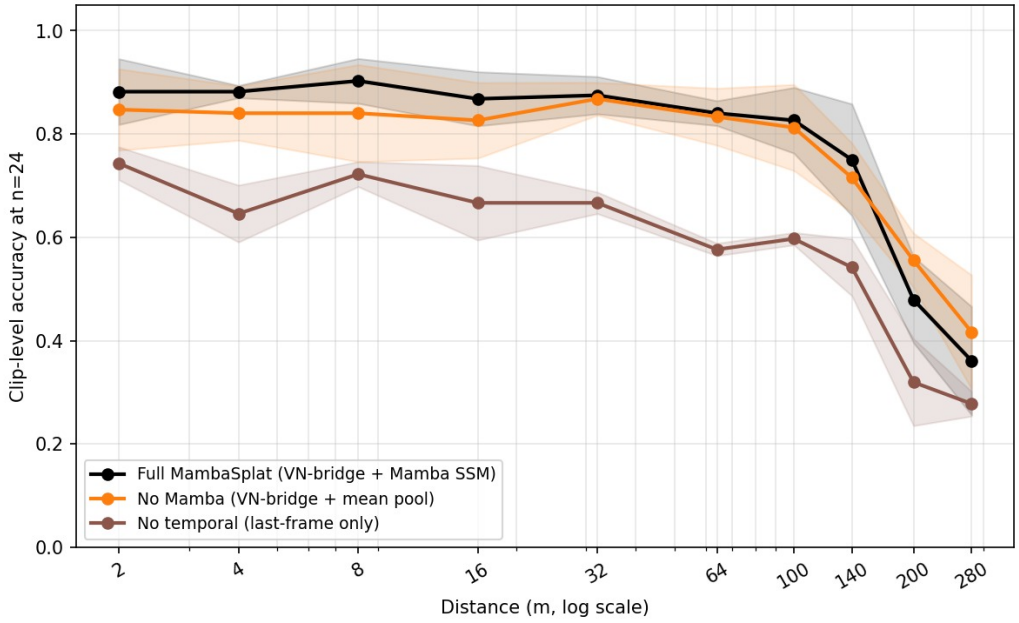


Figure 6.15: Clip-level accuracy versus camera distance at $T = 24$ (log x-axis), 3-seed mean $\pm 1\sigma$. Distances match the Sim-1 dyadic-octave sweep of §6.3.2.

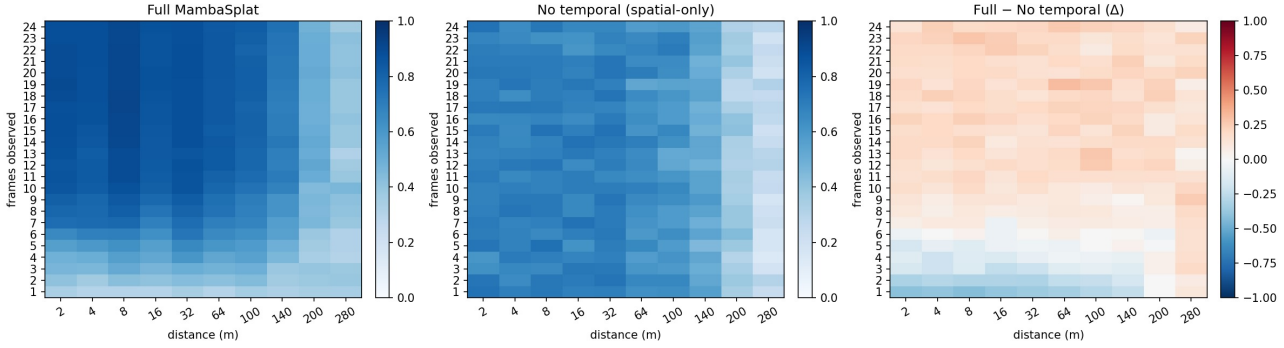


Figure 6.16: Accuracy over (frames observed × distance), seeds pooled. Left: full pipeline; center: spatial-only baseline; right: their difference (red = temporal helps). Temporal context helps once enough frames accumulate and up to the long-range floor.

Per-class accuracy (Figure 6.17) is consistent with the four-class results (§6.5.3) in that helicopter remains the hardest class for the spatial-only baseline (0.47) and is recovered most by the temporal stack (to 0.78, the largest per-class gain of any class). The per-class view also resolves the aggregate near-tie between the Mamba SSM and mean pooling into a double dissociation: relative to the mean-pool aggregator, the SSM helps helicopter (0.78 versus 0.71) and slightly cedes airplane (0.76 versus 0.83), while bird and drone are unchanged within noise on the all-distance average. The marginal ∼1.5-point SSM effect of Table 6.13 is therefore not a uniformly small gain but the average of two opposing per-class effects of roughly ±7 points: the SSM concentrates its capacity on the hardest class. With three seeds and a single rotation trial the $\pm 1\sigma$ bands of both contrasts overlap, so this redistribution is suggestive rather than established. Drone is the class for which temporal context helps least, since its spatial-only baseline is already the highest (0.675) and its temporal ceiling the lowest (0.745). The

confusion matrices (Figure 6.18) show that the helicopter recovery is structural rather than a uniform diagonal lift: without temporal context, the helicopter is misread mainly as bird (0.24) or drone (0.23) and only rarely as airplane (0.06), and accumulating frames removes most of that off-diagonal mass. Figure 6.19 resolves the accuracy–distance curves per class, one colored line per temporal variant, and shows the aggregator ordering is class- and distance-dependent rather than uniform. Mean pooling matches or edges the full pipeline once geometry degrades for three of the four classes: on the airplane from the mid-range outward, and on the drone and helicopter at long range. Bird is the exception: the Mamba SSM is the better aggregator at range, the full pipeline holding ∼0.08 above mean pooling at 100–280 m while their near-range accuracies coincide. This is the one class whose all-distance average (Figure 6.17) hides a distance-resolved effect, and it is consistent with the temporal characterization (§6.3.5): birds carry the strongest, most coherent low-frequency periodic signal, exactly the temporal structure an order-aware SSM can exploit but a mean pool discards, and the cue that matters most once per-frame spatial quality is degraded.

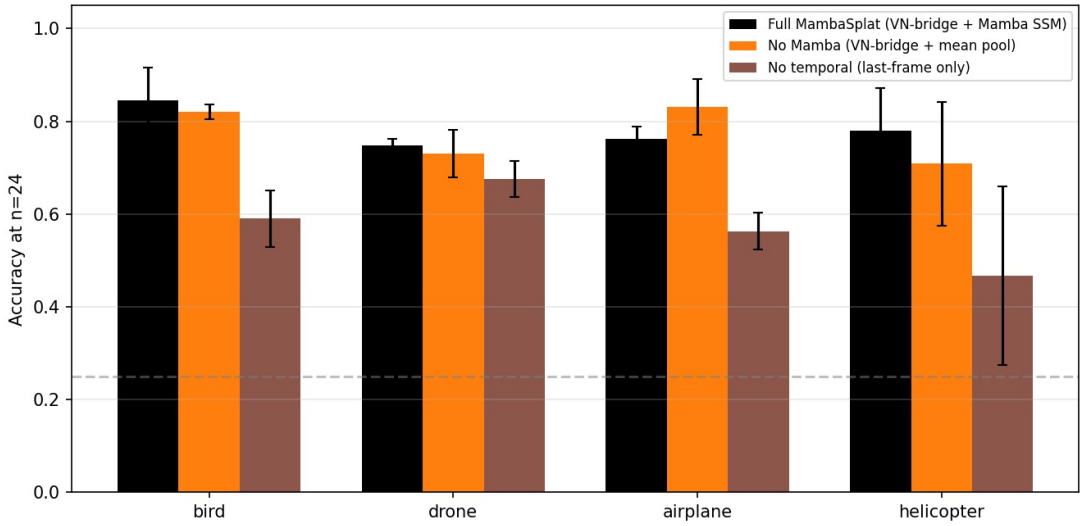


Figure 6.17: Per-class accuracy at $T = 24$ on the test split, 3-seed mean $\pm 1\sigma$.

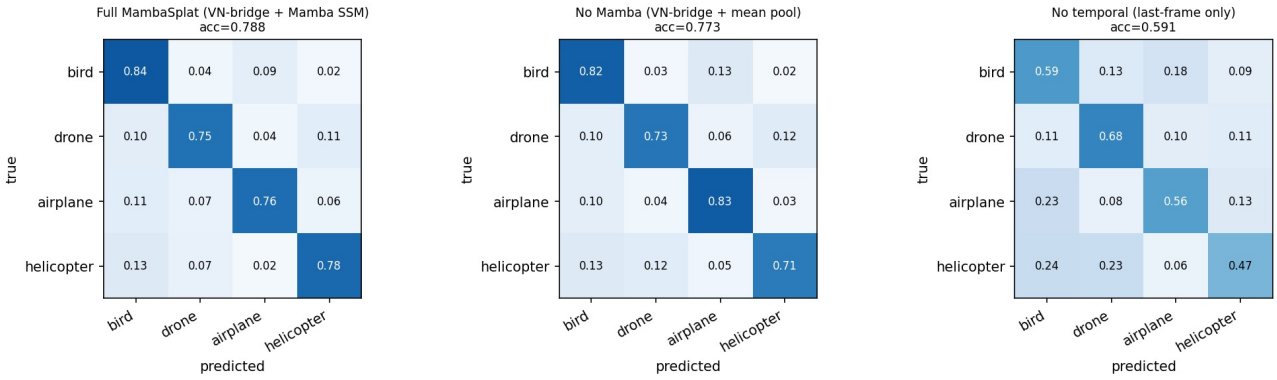


Figure 6.18: Row-normalized confusion matrices at $T = 24$ on the `AeroSplat-4D` Sim-1 test split (SO(3) evaluation, video-level, seeds 42–44 pooled). Left: full `MambaSplat-4D`; center: *w/o Mamba SSM* (VN-In bridge + mean pool); right: *w/o temporal* (last frame). Without temporal context, the helicopter is chiefly confused with a bird (0.24) and a drone (0.23); as temporal context accumulates, the helicopter diagonal rises from 0.47 to 0.78 and the off-diagonal mass collapses.

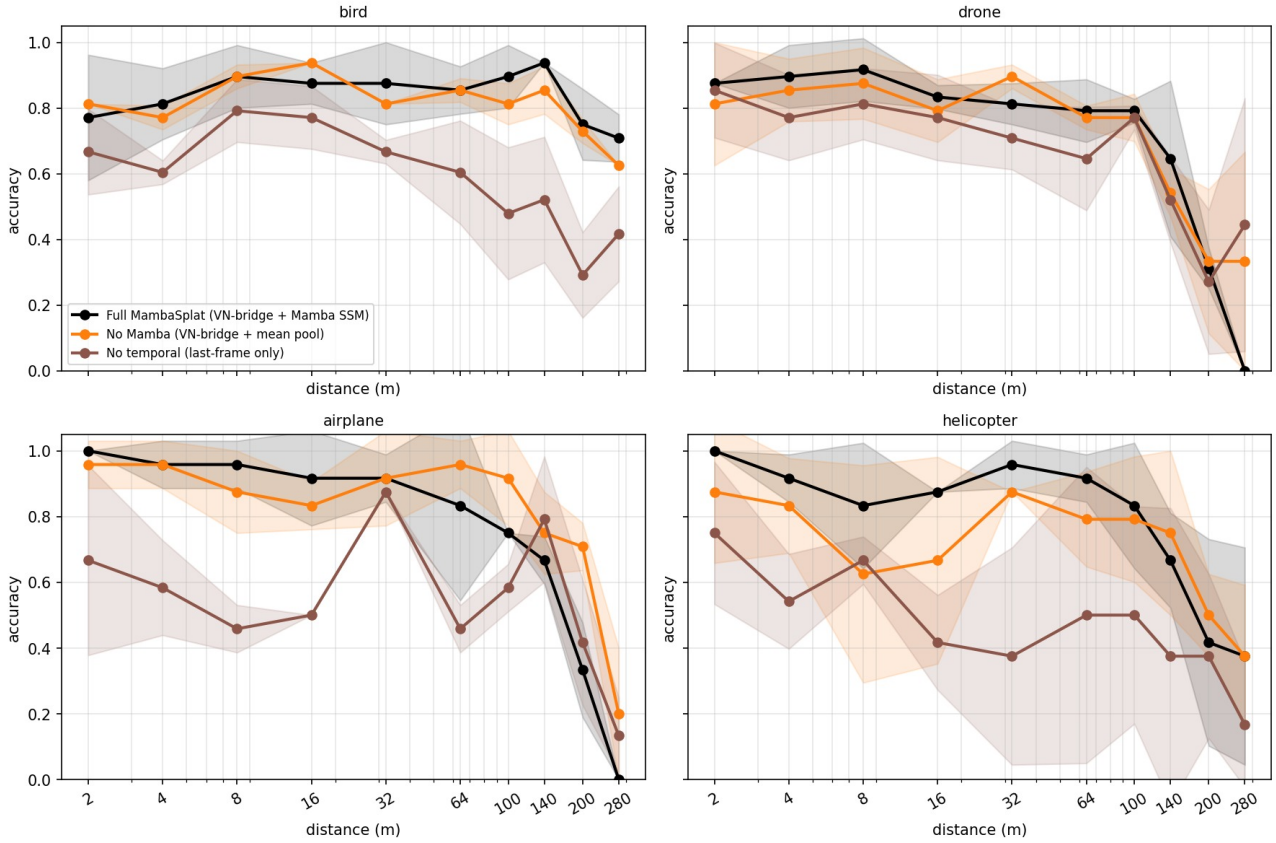


Figure 6.19: Clip-level accuracy versus camera distance at $T = 24$ (log x-axis), resolved per class, 3-seed mean $\pm 1\sigma$; one line per temporal variant (full pipeline, *w/o Mamba SSM*, *w/o temporal*). The per-class counterpart of Figure 6.15.

Temporal-order sensitivity (frame shuffling)

To separate whether the model uses frame *order* or only the unordered set of per-frame features, we re-evaluate the `full_X DepthSplat` checkpoints (seeds 42–44, $K = 10$ random permutations per clip) with the $T = 24$ input shuffled and compare against the in-order baseline across the distance sweep (Figure 6.20). The effect is class-dependent: shuffling costs the bird class 0.125 ± 0.051 in clip accuracy, a band that excludes zero, while drone, airplane, and helicopter are unchanged within noise (drops of -0.019 ± 0.024, 0.008 ± 0.029, and 0.011 ± 0.025). The bird separation persists across the near-to-mid range and matches T1.5a (§6.3.5): birds carry the strongest, most coherent low-frequency signal, an approximate 12-frame wing-flap period, exactly the ordered structure that shuffling destroys and a mean pool would discard. Order sensitivity is not evidence that the other classes are non-temporal, since a single repeated frame collapses every class (single-frame clip accuracy of 0.631, 0.159, 0.000, and 0.407 for bird, drone, airplane, and helicopter, against in-order accuracies of 0.76 to 0.84): the model accumulates across frames for all four classes and additionally respects their order only where periodic structure exists to exploit.

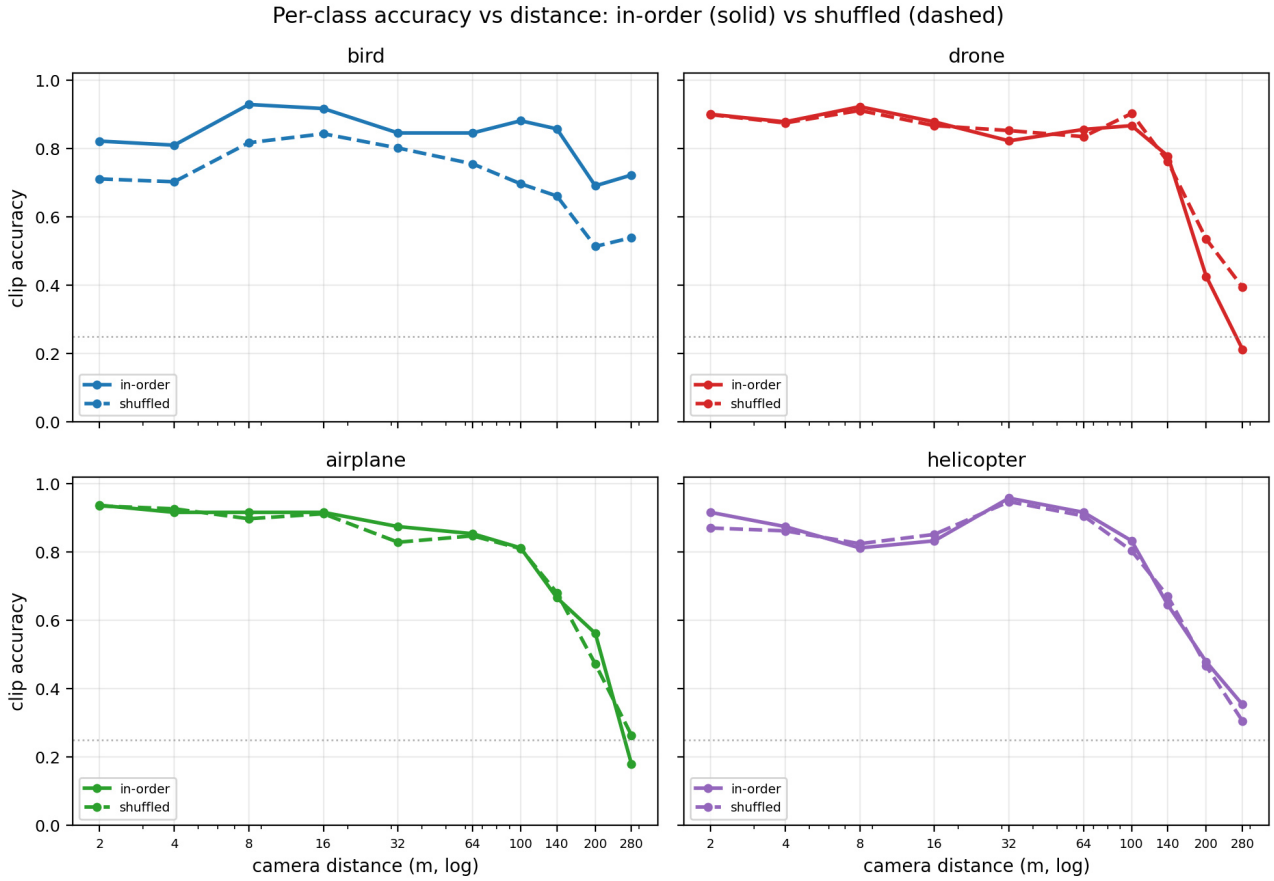


Figure 6.20: Per-class in-order (solid) versus shuffled (dashed) clip accuracy across the Sim-1 distance sweep at $T = 24$ (log x-axis), 3-seed mean over $K = 10$ shuffle permutations, one panel per class; the dotted line marks the four-class chance level (0.25). The in-order and shuffled curves separate only for birds, and only across the near-to-mid range where the coherent ∼12-frame wing-flap periodicity of (§6.3.5) supplies ordered structure to exploit; the other three classes overlap at every distance. The noisy far-range bins (200–280 m) reflect the small per-class asset counts there.

6.5.6 Bridge-to-Mamba Hidden Dimension Sweep

This experiment sweeps the Mamba hidden dimension $d_{\text{mamba}} \in \{64, 128, 256, 512\}$ on Sim-1 to quantify the accuracy, parameter-count, and latency trade-off at the invariant bottleneck, holding every other component fixed at the §5.6.11 defaults so that differences *across* variants isolate d_{mamba} alone. Because all four variants preserve the SO(3)-invariant VN-In bridge, the none, z, SO(3), and per-frame SO(3) protocols match within each variant up to evaluation noise, so a single accuracy column is shown per split. Accuracy peaks at the default $d_{\text{mamba}} = 256$ (78.8% val, 77.3% test); the narrower 64 and 128 bottlenecks are marginally lower (77.8% and 77.0% val), and the widest 512 collapses to 74.3% val, even as parameters grow from 0.21 M to 7.04 M; latency stays near 17.9 ms throughout. Accuracy is therefore essentially flat from 64 to 256 and only degrades once the bottleneck is widened to 512, so 256 is the accuracy-efficiency sweet spot. The efficiency columns are deterministic with respect to d_{mamba} and are reported once per variant.

Table 6.14: Bridge-to-Mamba hidden-dimension d_{mamba} sweep on `AeroSplat-4D` (`DepthSplat-OC` 3DGS, Sim-1 distance sweep, $T = 24$). All variants are trained *without* rotation augmentation, with 3 seeds (42, 43, 44) × 3 rotation trials. Video-level accuracy (%) ↑ on the validation and test splits; mean ± std across seeds. The model is rotation-invariant by construction, so the four rotation columns (none, z, SO(3), SO(3)_{pf}) are numerically identical within each variant, and a single accuracy column is shown per split. #Params and per-sequence latency are deterministic with respect to d_{mamba} and are reported once per variant (efficiency protocol: §6.1.2).

d_{mamba}	val ↑	test ↑	#Params (M)	Latency/seq (ms) ↓
64	77.8 ± 3.5	76.2 ± 1.0	0.21	17.89
128	77.0 ± 0.7	75.8 ± 1.6	0.56	17.95
256 (default)	**78.8 ± 2.2**	**77.3 ± 2.3**	1.88	17.97
512	74.3 ± 5.2	73.4 ± 2.9	7.04	17.94

6.5.7 Mamba Depth Sweep

This experiment sweeps the Mamba depth $n_{\text{layers}} \in \{1, 2, 4, 8\}$ on Sim-1 to quantify the temporal-capacity requirement, holding $d_{\text{mamba}} = 256$ and every other component fixed at the §5.6.11 defaults so that differences across variants isolate n_{layers} alone. As in the hidden-dimension sweep (§6.5.6), the SO(3)-invariant VN-In bridge makes the four rotation protocols match within each variant up to evaluation noise, so a single accuracy column is shown per split. Accuracy peaks at the default four layers (80.3% val, 77.9% test); one and two layers are lower (77.3% and 76.5% val) and eight is no better (76.8% val), while parameters grow from 0.57 M to 3.63 M and latency stays near 17.8–18.2 ms. Temporal depth helps up to four layers; beyond that, it adds parameters without accuracy. The efficiency columns are deterministic with respect to n_{layers} and are reported once per variant.

Table 6.15: Mamba depth n_{layers} sweep on `AeroSplat-4D` (`DepthSplat-OC` 3DGS, Sim-1 distance sweep, $T = 24$). All variants are trained *without* rotation augmentation, with 3 seeds (42, 43, 44) × 3 rotation trials. Video-level accuracy (%) ↑ on the validation and test splits; mean ± std across seeds. Rotation-invariant by construction, so the four rotation columns are numerically identical within each variant, and a single accuracy column is shown per split. #Params and per-sequence latency are deterministic in n_{layers} and reported once per variant (efficiency protocol: §6.1.2).

n_{layers}	val ↑	test ↑	#Params (M)	Latency/seq (ms) ↓
1	77.3 ± 6.5	76.2 ± 6.9	0.57	17.78
2	76.5 ± 1.9	76.4 ± 2.3	1.00	17.84
4 (default)	**80.3 ± 1.4**	**77.9 ± 1.6**	1.88	17.97
8	76.8 ± 5.6	75.0 ± 2.4	3.63	18.24

6.5.8 Temporal-Context Sensitivity

This experiment runs two joint sweeps on Sim-1 to quantify how much temporal context `MambaSplat-4D` needs: a sequence-length sweep $T \in \{4, 8, 12, 16, 24\}$ at $\Delta t = 1$, and a frame-interval sweep $\Delta t \in \{1, 2\}$ at $T = 12$, for six variants in total (the $T = 12$, $\Delta t = 1$ point is shared). Every other component is held fixed at the §5.6.11 defaults, with $T = 24$, $\Delta t = 1$ the recipe default carried over from the preceding Stage-3 experiments.

Unlike the width and depth sweeps (§6.5.6, §6.5.7), which fix the training T and sweep architecture, this experiment *trains* at each $(T, \Delta t)$, so the clips-per-video count varies: a 30-frame source clip at a training stride of 3 yields roughly $\lfloor (30 - (T-1)\Delta t)/3 \rfloor + 1$ clips, and smaller T sees more clips per epoch. All six variants preserve the SO(3)-invariant VN-In bridge, so the four rotation columns match within each variant up to evaluation noise, and the efficiency columns are deterministic in $(T, \Delta t)$ and reported once per variant.

Table 6.16: Temporal-context sensitivity on `AeroSplat-4D` (`DepthSplat-OC` 3DGS, Sim-1 distance sweep). All variants are trained *without* rotation augmentation, with 3 seeds (42, 43, 44) × 3 rotation trials. Video-level accuracy (%) ↑ on the validation and test splits under four rotation protocols; mean ± std across seeds. Rotation-invariant by construction, so the four rotation columns should be numerically identical within each variant up to seed noise; the sweep dimensions are sequence length T (top block, $\Delta t = 1$) and frame interval Δt (bottom block, $T = 12$). The temporal span is $(T-1)\Delta t + 1$ frames. Latency is the median over 50 runs after 5 warm-up iterations. Rotation protocols follow §5.6.12, Table 5.6; efficiency protocol: §6.1.2.

			Val				Test						
T	Δt	Span	none	z	SO(3)	$SO(3)_{pf}$	none	z	SO(3)	$SO(3)_{pf}$	#Params (M)	GFLOPs	Latency/seq (ms) ↓
Varying sequence length ($\Delta t = 1$):													
4	1	4	79.4 ± 5.4	79.4 ± 5.4	79.4 ± 5.4	79.4 ± 5.4	76.2 ± 3.6	76.2 ± 3.6	76.2 ± 3.6	76.2 ± 3.6	1.88	17.26	4.2
8	1	8	78.1 ± 1.7	78.1 ± 1.7	78.1 ± 1.7	78.1 ± 1.7	74.7 ± 0.2	74.7 ± 0.2	74.7 ± 0.2	74.7 ± 0.2	1.88	34.51	7.0
12	1	12	78.2 ± 5.0	78.2 ± 5.0	78.2 ± 5.0	78.2 ± 5.0	77.3 ± 6.5	77.3 ± 6.5	77.3 ± 6.5	77.3 ± 6.5	1.88	51.77	9.5
16	**1**	**16**	**81.0 ± 1.6**	**81.0 ± 1.6**	**81.0 ± 1.6**	**81.0 ± 1.6**	**80.1 ± 3.5**	**80.1 ± 3.5**	**80.1 ± 3.5**	**80.1 ± 3.5**	1.88	69.03	12.3
24 (default)	1	24	77.9 ± 3.8	77.9 ± 3.8	77.9 ± 3.8	77.9 ± 3.8	76.8 ± 2.3	76.8 ± 2.3	76.8 ± 2.3	76.8 ± 2.3	1.88	103.54	18.0
Varying frame interval ($T = 12$):													
12	**1**	12	78.2 ± 5.0	78.2 ± 5.0	78.2 ± 5.0	78.2 ± 5.0	77.3 ± 6.5	77.3 ± 6.5	77.3 ± 6.5	77.3 ± 6.5	1.88	51.77	9.5
12	2	23	75.4 ± 6.1	75.4 ± 6.1	75.4 ± 6.1	75.4 ± 6.1	71.5 ± 6.2	71.5 ± 6.2	71.5 ± 6.2	71.5 ± 6.2	1.88	51.77	9.5

Accuracy is non-monotonic in the sequence length, and the standard-deviation bands overlap heavily. Within the $\Delta t = 1$ sweep, it peaks at $T = 16$ (80.1 % test), while the carried-over default $T = 24$ is mid-pack (76.8 % test): temporal context beyond roughly 16 frames yields no accuracy gain, yet costs about 50 % more compute (69.03 versus 103.54 GFLOPs, 12.3 versus 18.0 ms per sequence). $T = 16$ is therefore the accuracy–efficiency sweet spot, and the $T = 24$ default carried over from the preceding experiments sits within the overlapping noise band rather than at the optimum. Widening the frame interval to $\Delta t = 2$ (a 23-frame span at $T = 12$) lowers accuracy to 71.5 % test from the 77.3 % of the matched-compute $\Delta t = 1$ point, indicating that denser temporal sampling is preferable to a longer but sparser span.

6.5.9 Full-Pipeline SO(3) Invariance Drift

This experiment is the end-to-end empirical verification of the invariance proof (§5.6.8). We evaluate `MambaSplat-4D` $E(X)$ and $E(C)$ (centroid-only lifting, otherwise identical) on the `AeroSplat-4D` Sim-1 test split, exposing each seed (42, 43, 44) to $N = 100$ Haar-uniform SO(3) rotations [130] over $M = 32$ fixed test clips (9600 rotated forward passes per model: 100 rotations × 32 clips × 3 seeds), in fp32 with TF32 disabled. We report the cross-seed worst-case drift $\Delta_{\max}$ and minimum predicted-class margin $m_{\min}$ against the verdict bands of §5.6.8 (HOLDS at $\leq 10^{-4}$, MARGINAL in $(10^{-4}, 10^{-2}]$, LEAK above 10^{-2}).

Table 6.17: Full-pipeline SO(3) invariance drift on the `AeroSplat-4D` Sim-1 test split (100 Haar rotations × 32 clips × 3 seeds). $E(X)$ realizes the exact-equivariance bound predicted by §5.6.8; $E(C)$ exceeds it by three orders of magnitude, because centroid-only lifting renders the learned frame $\mathbf{T}$ rank-degenerate and amplifies the ε-bias residual.

Model	ε	$\Delta_{\max}$	$m_{\min}$	$m_{\min}/(2\Delta_{\max})$	flip rate	verdict
MambaSplat-4D $E(X)$	10^{-6}	8.09×10^{-5}	0.173	1067×	0.0000	HOLDS
MambaSplat-4D $E(C)$	10^{-6}	2.93×10^{-1}	7.30×10^{-6}	1.25×10^{-5}	0.0008	LEAK

The two feature modes give opposite verdicts, both confirming the theory. For $E(X)$ the bound holds: $\Delta_{\max} \leq 10^{-4}$, the safety ratio $m_{\min}/(2\Delta_{\max}) \approx 1070\times$ satisfies the decision-invariance inequality (Eq. 5.18), and zero argmax flips occur across all 9600 passes. For $E(C)$ the invariance leaks: $\Delta_{\max} \approx 0.29$ exceeds the LEAK threshold, the margin collapses to $m_{\min} \approx 7.3 \times 10^{-6}$, and flips occur at a rate of 8×10^{-4}. This is the non-degeneracy footnote of §5.6.8 materializing: centroid-only lifting yields a rank-deficient $\mathbf{T}$ that amplifies the ε-residual benign for $E(X)$. An identity-rotation self-check confirms the drift is rotation-induced rather than reduction-order noise (fp32 floor $\leq 3.1\times10^{-6}$ for $E(X)$, exactly zero for $E(C)$, bit-exact under the identity); the $E(X)$ drift sits two orders of magnitude above this floor, consistent with the ε-bias propagation of Eq. 5.3.

6.6 RQ4 – End-to-End System (Stage 4)

RQ4 benchmarks the full pipeline against the 2D baselines of the IJCNN 2025 WOSDETC Drone vs Bird Detection Challenge [6], evaluated on `AeroSplat-4D` across the three regimes of §5.3.4 (Table 5.1): Sim-1

ID, Sim-2 ID, and Sim-3 OOD. The contribution of mask-gated photometric supervision is quantified at the reconstruction level in Table 6.5; end-to-end latency on target hardware is reported in §6.6.2 and §6.6.3.

Evaluation basis. All methods are compared on the same identity-disjoint split. The in-distribution regimes (Sim-1, Sim-2) are scored on the held-out test split, and the Sim-3 OOD probe is scored on its test split, which is the only split it provides. Our rows use the `DepthSplat-OC` reconstruction pipeline (§5.5.1), and the 2D baselines and our classifier share the evaluation harness of §5.7.1.

6.6.1 2D-Baseline Comparison

This experiment compares progressive capabilities, walking from single-frame 2D detection through detect-and-track and multi-frame 2D fusion to our multi-view 3D-and-temporal pipeline, each evaluated across the three regimes. The four 2D baselines (WRN-YOLO [136], YOLO26-cls [139], YOLOv7 [137] with a CSRT tracker [156], and FBOD-SV [138]) and their capability tiers are defined in §5.7.2.

All six rows are populated: the four 2D baselines and our two feature modes, $E(X)$ (all 14 Gaussian attributes) and $E(C)$ (centroids only), each evaluated across the three regimes.

Table 6.18: Progressive-capability comparison against 2D baselines on `AeroSplat-4D` under three evaluation regimes (Sim-1 ID, Sim-2 ID, Sim-3 OOD), plus macro-F1 on Sim-3. All methods are evaluated on the same identity-disjoint split: accuracy is reported on the held-out *test* split for the in-distribution regimes (Sim-1, Sim-2) and on the *test* split for the Sim-3 OOD probe, which is test-only. All figures are sequence-level, formed by per-sequence majority vote, matching the 2D baselines; clip-level figures for our rows are in the note below. Baselines use a single training run (seed 42), so no standard deviation is reported; our rows ($E(X)$, $E(C)$) report mean ± std over 3 seeds and use `DepthSplat-OC` reconstructions. The confusion matrices for our rows are in Appendix E. Baseline details: §5.7.2; shared evaluation harness: §5.7.1.

					Accuracy ↑			Macro-F1 ↑	
#	Method	Temp.	MV	3D	Sim-1 (ID)	Sim-2 (ID)	Sim-3 (OOD)	(Sim-3)	Capability
1	WRN-YOLO [136]	No	No	No	75.62	64.58	**45.83**	**28.35**	2D det. floor
2	YOLO26-cls [139]	No	No	No	**85.62**	**85.42**	33.33[†]	12.50[†]	2D appearance cls.
3	YOLOv7 [137] + CSRT	Trk	No	No	70.62	25.00	37.50	18.46	+ 2D tracking
4	FBOD-SV [138]	Fus	No	No	33.33[‡]	–[‡]	–[‡]	–[‡]	Best 2D temporal
5	**Ours:** $E(X)$	Yes	3D	Yes	78.8 ± 4.1	72.2 ± 2.4	24.6 ± 17.6	16.8 ± 14.0	**Full pipeline**
6	**Ours:** $E(C)$	Yes	3D	Yes	66.8 ± 7.8	72.2 ± 16.2	20.3 ± 6.6	13.6 ± 7.6	**Centroid-only**

[†]Single-class collapse: the model assigns one class to every Sim-3 input, so accuracy reduces to that class's prevalence and macro-F1 is 0.125 (one class at F1 0.5, the other three at 0); this is the four-class degenerate floor. [‡]FBOD-SV is reported as a uniformly failing baseline. On Sim-1, it collapses to a single predicted class (every input classified as drone; validation AP@50 $\approx 6 \times 10^{-4}$), yielding 33.33% accuracy and 12.50% macro-F1. The Sim-2 and Sim-3 cells are left empty by design: FBOD-SV training deadlocked at epoch 10 of 30 with no recoverable checkpoint and was not retrained.
Clip-level figures for our rows (vs the sequence-level numbers tabulated): $E(X)$ accuracy 78.8 / 73.6 / 27.0 across Sim-1 / Sim-2 / Sim-3 with Sim-3 clip macro-F1 18.9; $E(C)$ accuracy 66.8 / 68.8 / 26.5 with Sim-3 clip macro-F1 21.5.

The baseline rows expose a clear split between in-distribution accuracy and out-of-distribution generalization. Within distribution, the single-frame appearance classifier is strongest: YOLO26-cls reaches 85.62% on Sim-1 and 85.42% on Sim-2, above the WRN-YOLO detection floor of 75.62% and 64.58%. The detect-and-track baseline does not transfer even between the two in-distribution rigs, as YOLOv7 plus CSRT falls from 70.62% on Sim-1 to 25.00% on Sim-2 once the rig geometry changes. On the Sim-3 OOD probe, this ranking inverts and the appearance classifiers collapse: YOLO26-cls drops to 33.33% accuracy with a macro-F1 of 12.50%, the signature of single-class collapse, in which the model assigns one class to every input so that accuracy reduces to that class's prevalence. By contrast, WRN-YOLO degrades the least, retaining 45.83% accuracy and the highest Sim-3 macro-F1 at 28.35%, so the single-frame detector generalizes least badly to the line-formation geometry. FBOD-SV does not yield a usable temporal-fusion baseline on `AeroSplat-4D`: its only trainable regime (Sim-1) is degenerate, and its Sim-2 and Sim-3 cells are unavailable, as the table footnote details. Against this backdrop, the 3D-and-temporal pipeline is competitive in distribution: Ours $E(X)$ reaches 78.8% on Sim-1 and 72.2% on Sim-2, holding across the rig change that collapses the detect-and-track baseline, though it stays below YOLO26-cls. Under the Sim-3 OOD shift, however, neither feature mode surpasses the 2D detectors on accuracy: $E(X)$ retains 24.6% (16.8% macro-F1) and $E(C)$ 20.3% (13.6%), against WRN-YOLO's 45.83% (28.35%), so on these numbers the pipeline does not close the OOD gap on this probe. Retraining the strongest 2D classifier, YOLO26-cls, from random initialization instead of ImageNet-pretrained weights did not yield a usable baseline on `AeroSplat-4D`: under a 50-epoch budget, more than twice the 20 epochs the pretrained

variant needed to converge, validation top-1 accuracy stayed pinned at the majority-class floor of roughly 33% with a flat, near-uniform training loss, which is consistent with the appearance baselines' reliance on large-scale pretraining but does not by itself isolate the out-of-distribution axis.

6.6.2 Efficiency

This experiment reports per-frame efficiency on the RTX 5090 reference workstation across Stage-2 reconstruction (`DepthSplat-OC` and LGM), Stage-2.5 pre-pool decimation, and Stage-3 classification (`MambaSplat-4D` $E(C)$ and $E(X)$, Mamba4D, and P4Transformer). The measurement protocol is formalized in §5.7.3.

Reconstruction dominates the per-frame cost: `DepthSplat-OC` takes 52.34 ms/frame and LGM 21.24 ms/frame, against at most 3.55 ms/frame for any Stage-3 classifier. The two reconstructors trade off in opposite directions: `DepthSplat-OC` is roughly 10× smaller than LGM in parameters (38.12 M versus 415.04 M) but about 2.5× slower per frame and uses roughly twice the peak VRAM (5.67 GB versus 2.92 GB). The LGM figure is consistent with the 20.79 ms/frame U-Net latency reported for LGM [69]. Stage-2 cells time the model forward only, per the protocol in §5.7.3. Stage-2.5 pre-pool decimation is cheap: the canonical `rand(8192)`→`FPS(1200)`→`rand(1024)` path costs 5.50 ms/frame, the `rand(1024)` baseline 0.76 ms/frame at a source of 100,000 Gaussians, with negligible peak VRAM for both.

At Stage 3, `MambaSplat-4D` is 23× smaller than P4Transformer and 58× smaller than Mamba4D in parameters (1.85 M for $E(C)$ to 1.88 M for $E(X)$, versus 44.10 M and 109.77 M); its $E(C)$ GFLOPs (44.42) is about 1.2× that of P4Transformer (35.54), and its $E(X)$ GFLOPs (103.54) about 2.9×. Composing the stages, the end-to-end per-frame cost of `DepthSplat-OC` plus pre-pool plus `MambaSplat-4D` $E(X)$ is roughly 58.59 ms/frame, which at $T = 24$ amounts to 1.41 s/clip; the LGM front-end gives roughly 27.49 ms/frame, or 0.66 s/clip. With the reconstruction cached, the live cost reduces to pre-pool plus Stage 3, roughly 6.2 ms/frame. Power draw is not instrumented, and the corresponding column is dropped from the merged table. The envelope is approximately 300 W on the RTX 5090, and an on-device Jetson AGX Orin benchmark remains pending and is only qualitative.

Table 6.19: Per-frame efficiency on the RTX 5090 reference workstation across Stage-2 reconstruction, Stage-2.5 pre-pool decimation, and Stage-3 classification. Stage-2 latency is the reconstruction-model forward only; the custom CUDA rasterizer is excluded. Stage-2.5 vendor imports the canonical training-time decimation. Stage-3 latency is per-sequence ($T = 24$) divided by T, since the temporal Mamba SSM is linear in T. Median of 30 timed runs after 5 warm-ups, CUDA-synchronized. Cells marked with a dash do not apply: preprocessing has no learnable parameters, and FPS is not traced by the FLOP counter. Measurement protocol: §6.1.2; tooling: §5.7.3.

Stage	Model	#Params (M)	GFLOPs	Latency (ms/frame)	Peak VRAM (GB)
S2	**DepthSplat-OC (ours)**	**38.12**	**969.02**	**52.34**	**5.67**
S2	LGM [69]	415.04	1585.41	21.24	2.92
S2.5	**rand(8192)→FPS(1200)→rand(1024)**	–	–	**5.50**	**0.00**
S2.5	rand(1024) (baseline)	–	–	0.76	0.00
S3	**MambaSplat-4D** $E(C)$	**1.85**	**44.42**	**0.63**	**1.34**
S3	**MambaSplat-4D** $E(X)$	**1.88**	**103.54**	**0.75**	**1.41**
S3	Mamba4D [104]	109.77	304.74	3.55	0.53
S3	P4Transformer [135]	44.10	35.54	0.36	0.21

Edge deployment note: the Jetson AGX Orin is a target platform, but no on-device benchmark has been run yet. The peak GPU memory (roughly 1.4 GB for Stage 3, 2.92 GB for LGM, and 5.67 GB for `DepthSplat-OC` per Table 6.19) suggests feasibility; an on-device benchmark remains pending and qualitative only.

6.6.3 2D-Baseline Latency

This experiment completes the latency comparison with a per-frame analysis of the four 2D baselines against `MambaSplat-4D`, holding apart two regimes the comparison must not conflate. End-to-end, the 2D baselines are the faster pipelines per frame: each is a single forward over one input image (0.93 to 11.26 ms per frame; Table 6.20), whereas the full 3DGS pipeline must first reconstruct the scene, which raises its per-frame cost to

roughly 58.59 ms and leaves reconstruction as the dominant term (Table 6.19). At the classifier-forward level the table reports, where each method's preprocessing is excluded just as the 2D figures exclude image loading and non-maximum suppression, the ordering reverses: `MambaSplat-4D`'s Stage-3 classifier costs only 0.63 to 0.75 ms per frame, below every 2D baseline, though at a higher peak GPU memory (1.34 to 1.41 GB versus 0.13 to 0.93 GB). Among the baselines, the wide WRN-YOLO backbone is the lone per-frame outlier at 11.26 ms on its 960^2 input, while the other three run between 0.93 and 1.39 ms at their own deployed resolutions; FBOD-SV is nominally the cheapest at 0.93 ms, but this divides its five-frame fused forward by five, reporting steady-state amortization while hiding the five-frame buffering window a streaming deployment would incur. Against the roughly 30 FPS edge target, the 2D baselines and the reconstruction-cached live path (pre-pool plus Stage 3, about 6.2 ms per frame) both sit within budget, whereas the full pipeline at 58.59 ms per frame does not, confirming reconstruction as the frame-rate bottleneck.

Table 6.20: Forward-only inference latency on the RTX 5090 reference workstation for the four 2D baselines of Table 6.18 (rows 1–4) and the Stage-3 `MambaSplat-4D` classifier. The model is only forward-propagated at each method's deployed input resolution; image loading, non-maximum suppression, and the CSRT tracker are excluded. FBOD-SV fuses five frames per forward, so its per-frame figure is the five-frame forward divided by five. Median of 30 timed runs after 5 warm-ups, CUDA-synchronized. The `MambaSplat-4D` rows are reproduced from Table 6.19. Measurement protocol: §6.1.2; tooling: §5.7.3. Baseline details: §5.7.2.

Method	Latency / frm (ms)	#Params (M)	GPU Mem (GB)
WRN-YOLO [136]	11.26	100.68	0.93
YOLO26-cls [139]	1.18	1.53	0.17
YOLOv7 [137] + CSRT	1.39	6.02	0.13
FBOD-SV [138]	0.93	46.56	0.55
MambaSplat-4D $E(C)$	**0.63**	**1.85**	**1.34**
MambaSplat-4D $E(X)$	**0.75**	**1.88**	**1.41**

7 Discussion

7.1 Overview

This chapter interprets the experimental results one research question at a time, situating each against the relevant prior work and the limitations that bound it. Because Stages 0–2 are engineering contributions and Stage 3, the rotation-invariant temporal classifier, is the core academic contribution of this thesis (Chapter 1), the interpretation is weighted accordingly: RQ1 and RQ2 are read as evidence that the pipeline furnishes the classifier with well-annotated, adequately reconstructed inputs, while RQ3 receives the most extended treatment. Each section opens with a direct verdict on its research question and then turns to why the numbers come out as they do; the overall synthesis across the four questions, and the directions they open, are drawn together in the Conclusion (Chapter 8).

7.2 RQ1: Dataset Generation and Annotation

How can we generate a synchronized multi-camera dataset of flying objects with full ground-truth annotations (RGB, instance masks, depth, 3D trajectories, and camera intrinsics/extrinsics) suitable for training and evaluating feed-forward 3DGS reconstruction and 4D temporal classification? Yes, and the contribution is best stated as a comparative one: across the benchmarks surveyed in Table 6.2, `AeroSplat-4D` is, to our knowledge, the only one that simultaneously provides wide-baseline multi-view capture, four near-balanced flying-object classes ($H/H_{\max} > 0.95$), and reconstructed 3DGS, together with the complete per-frame annotation stack (RGB, instance mask, depth, 3D position, 6-DoF trajectory, and camera intrinsics and extrinsics) that feed-forward 3DGS reconstruction and 4D temporal classification both require.

The distinguishing property is completeness of supervision rather than raw scale, and the head-to-head positioning (Table 6.2, §6.3.1) makes the gap concrete. We do not claim the largest corpus: a single real benchmark, such as AOT, reaches roughly 5.9 M frames, compared with Sim-1's 54 600. What no prior dataset offers is the *pairing* this pipeline depends on. The real airborne benchmarks in the comparison capture with a single camera (two for Anti-UAV) and annotate only 2D bounding boxes, occasionally with a track identifier or range; the synthetic airborne datasets SynDroneVision [141] and UEMM-Air [142], and the ShapeSplat [96] 3DGS anchor, each supply only part of the required stack. `AeroSplat-4D` delivers all of it at once, at 2560×1440 across a 2–380 m target-distance envelope and four to five synchronized cameras [5], [143]. That intersection, rather than any one of its components, is what makes the dataset enabling.

This is also why the dataset is best understood as an engineering enabler rather than a result in itself. Because the foreground masks are taken from the simulator's ground-truth channel rather than a learned segmenter, the reconstruction and classification stages can be evaluated on their own merits, isolated from segmentation error, and every downstream ablation in this thesis draws on the same annotations. Following established dataset-documentation practice [144], the dataset-composition analysis (§6.3.1) confirms that this supervision is well-conditioned for learning: normalized class entropy stays between 0.959 and 0.978 across all seven (Sim, split) cells, so no class dominates, and the residual imbalance is a deliberate ∼2:1 primary-to-secondary asset ratio rather than a sampling artifact, on identity-disjoint splits that keep any single physical asset out of both the training and test partitions.

The dataset analysis further provides empirical support for temporal modeling rather than assuming it. The temporal characterization (§6.3.5) measures a characteristic per-class motion periodicity directly from the reconstructed geometry: the bird gives the strongest and most coherent signal at a period of roughly twelve frames, drones oscillate at roughly eight frames and helicopters at roughly sixteen, while the airplane's signal is weakest and least regular. This periodicity is, by construction, invisible to a per-frame 2D detector or a static 3D classifier, which motivates aggregating evidence across frames rather than committing to any single one. Because per-frame signals degrade with range, extended temporal context precisely offsets weaker single-frame evidence at distance, which is the design rationale that RQ3 then tests directly.

7.3 RQ2: Feed-Forward Reconstruction

How can feed-forward Gaussian Splatting methods be adapted to reconstruct small, distant flying objects from sparse multi-camera views against textureless sky backgrounds? Yes, with a clear qualifier on what kind of success this is. Feed-forward Gaussian Splatting can be adapted to the aerial, textureless-sky regime, and the adaptation that makes it work is mask-gating: conditioning the cost volume and encoder on the foreground mask is the single dominant design factor, worth +2.50 dB PSNR on GSO and +2.90 dB on ABO over the unmasked model (§6.4.1). On absolute reconstruction quality `DepthSplat-OC` is not state-of-the-art, but it is the strongest feed-forward reconstructor in its compute tier, and that is the more useful claim to make.

The comparative standing is best read in terms of the quality-versus-compute trade-off rather than on PSNR in isolation, and on that axis the result is stronger than a bare quality number suggests. At its training-matched five-view operating point `DepthSplat-OC` reaches 24.65 PSNR on GSO, beating the closest-protocol LGM baseline by 3.2 dB (21.44 PSNR) while training on roughly 27 GPU-hours against LGM's roughly 3,072, more than a hundredfold less compute (§6.4.2) [69]. In other words, it dominates the cheap baseline across both quality and cost. The only matched-protocol baseline that scores higher, GS-LRM at 29.59 PSNR, reached that quality with roughly 114 times more training compute: approximately 3,072 GPU-hours on 64 A100 GPUs and a 300 M-parameter transformer, against a single RTX 5090 [70]. We therefore view the residual gap to GS-LRM as a compute gap rather than an architectural deficit, and `DepthSplat-OC` as a proof-of-concept that a lightweight, single-GPU reconstructor can be the strongest option within the thesis's compute budget and adequate for serving the downstream classification pipeline [63]. The comparison still carries the caveat that baseline numbers are quoted under each source's own protocol without variance estimates.

Two findings bound how far the reconstructor can be relied upon. First, the monocular depth branch initialized from Depth Anything v2 is tightly coupled to the cost-volume module it is trained against: reverting to the stock backbone costs a further +2.09 dB PSNR on GSO (§6.4.1) [63], [66], so the depth prior cannot be hot-swapped and exploiting a stronger one would require a full retrain rather than an inference-time substitution. Second, reconstruction quality degrades with target distance: the distance sweep (§6.4.3) shows LPIPS rising beyond 32 m, and beyond 140 m the target subtends only a few pixels, so the upscaled 256×256 crop becomes nearly uniform sky, and the reconstruction collapses to a near-white field. The apparent recovery of PSNR and SSIM at the longest ranges is a degenerate-crop artifact rather than a genuine gain, flagged as such and treated with caution; reaching usable quality past roughly 64 m would require higher-resolution or physically zoomed cameras.

7.4 RQ3: Temporal 4D Gaussian Classification

Can temporal changes in 4D Gaussian parameters (position, scale, rotation, opacity) provide discriminative features for classification that improve upon static 3D or 2D appearance-based methods? Yes, decisively, and this is the question on which the pipeline most clearly outperforms prior work. `MambaSplat-4D` is, to the best of our knowledge, the only 4D temporal classifier whose rotation invariance is structural rather than learned from augmentation. Under per-frame SO(3) rotation, the strongest temporal-3D baselines collapse to near chance (P4Transformer to 5.2 % and Mamba4D to 10.3 %), whereas our model stays stable across all four rotation protocols (§6.5.2). That invariance is verified rather than merely asserted: across 9600 rotated forward passes, the decision never changes (zero argmax flips, worst-case drift $\Delta_{\max} = 8.09 \times 10^{-5}$), clearing the decision-invariance inequality by a safety margin of roughly 1067× (§6.5.9). On clean, un-rotated data the strongest baseline is in fact modestly ahead, Mamba4D reaching 74.9 % against our 70 to 71 %, but that ordering inverts the moment rotation enters, and augmentation cannot substitute for the structural guarantee: an SO(3)-augmented Mamba4D recovers only to 58.2 %, still about 13 points below the 70.2 % that $E(X)$ holds

under arbitrary rotation (§6.5.3). The model delivers this while being 23 to 58 times smaller than those baselines, at roughly 1.9 M parameters against P4Transformer's 44.10 M and Mamba4D's 109.77 M (§6.6.2).

This research question is the heart of the thesis, and the headline accuracy behind the verdict above merits unpacking. The full-attribute temporal mode $E(X)$ reaches 70.2 % four-class accuracy on Sim-1, surpassing the positions-only baseline $E(C)$ by approximately 8 percentage points (§6.5.3). We report 70.2 % as the deployment-facing figure because it holds under the full rotation-invariance protocol that matches operational airspace monitoring; the approximately 80 % seen in the bridge and component ablations (§6.5.4, §6.5.5) comes from the less demanding rotation-augmentation-free subset. To the best of our knowledge, `MambaSplat-4D` is the first system to combine 4D Gaussian temporal features with a flying-object classification objective, a gap already identified in the related-work review (Chapter 4): the primary community benchmark, WOSDETC Drone vs Bird [6], is 2D-detection only, while Mamba4D [104] and P4Transformer [135] apply non-Gaussian temporal-3D representations to human-action recognition. The remainder of this section interprets, in turn, the rotation invariance that makes the approach operationally credible, the attribute set that makes it accurate, the temporal aggregation that makes it discriminative, and the capacity at which it saturates.

Structural rotation invariance, not augmented coverage. Because `DepthSplat-OC` reconstructs each frame independently, the Gaussian point cloud at time t need not share the coordinate frame of the one at $t-1$. Racing quadcopters have been reported to execute sharp 90-degree direction changes on timescales of tens of milliseconds, comparable to the per-frame reconstruction latency, so arbitrary inter-frame rotations follow directly, and we therefore enforce SO(3) invariance at the classifier rather than relying on data augmentation. The distinction matters because augmentation improves rotation robustness only distributionally, broadening training coverage while leaving untrained rotations exposed, whereas the bias-free VN linear layers in `MambaSplat-4D` make the invariance unconditional [29], [106]. The VN-In bridge realizes this structurally: $(\mathbf{V}\mathbf{R}^\top)(\mathbf{T}\mathbf{R}^\top)^\top = \mathbf{V}\mathbf{T}^\top$ holds for any rotation $\mathbf{R}$ with no assumption on the learned frame $\mathbf{T}$, so the guarantee is built in rather than enforced or checked. The empirical consequence is stark. On MSR-Action3D under per-frame SO(3) rotation, P4Transformer collapses to 5.2 % accuracy and Mamba4D to 10.3 %, near chance, whereas `MambaSplat-4D` stays stable across all four rotation protocols (§6.5.2) [104], [135]. On the four-class Sim-1 table, augmentation recovers only part of this gap: the SO(3)/SO(3)-augmented Mamba4D variant reaches 58.2 %, still trailing $E(X)$ by approximately 13 percentage points (§6.5.3). Because that rotation study (§6.5.2) uses a single training seed and a human-action proxy dataset, its numbers are directional evidence for the importance of structural invariance rather than definitive magnitude comparisons in the flying-object domain. That the guarantee is not merely nominal is confirmed by a controlled check of 9600 forward passes on Sim-1, which found zero argmax flips and $\Delta_{\max} \leq 8.09 \times 10^{-5}$, matching the ε-bounded invariance proof of Chapter 5 [106].

The learned bridge frame earns its dimensionality. The bridge frame $\mathbf{T}$ is learned and data-adapted rather than fixed, identifying which directions in the equivariant feature space are most discriminative for classification. The bridge-method ablation (§6.5.4) quantifies what this buys: the learned VN-In bridge is the strongest variant, reaching 80.0 ± 2.4 % val and 79.8 ± 3.7 % on the held-out test split of the rotation-augmentation-free evaluation, while the fixed and lower-dimensional alternatives trail in the mid-sixties, the fixed PCA frame among them at 64.6 % despite sharing the bridge's dimensionality. The near-identical validation and test figures indicate the result is not an artifact of the evaluation split, and the consistent advantage of the learned frame over the fixed alternatives confirms that both the full invariant dimensionality and the learned, data-adapted frame contribute. The takeaway against the Vector-Neuron operators is correspondingly nuanced: evaluated at the centroid-only mode $E(C)$ on ModelSplat-10, VN-DGCNN reaches 92.3 % against 42.8 % for our VN-In bridge (§6.5.1), but the gap is one of bridge design rather than of the underlying operators, since VN-In realizes its advantage only with the full attribute set, reaching 90.4 % at $E(X)$ where the learned cross-covariance frame makes the additional Gaussian attributes discriminative [29].

Attribute diversity is essential; appearance is a modest marginal. The attribute ablation (§6.5.1) asks whether the five 3DGS attribute families [15] each contribute independently or some are redundant given the others. The centroid-only baseline $E(C)$ reaches only 42.8 % on ModelSplat-10, near-degenerate for a ten-class problem, though this degeneracy is confounded by ModelNet reconstruction quality, the short 200-epoch budget against the 4000 epochs of the original VN-Transformer work [106], and single-seed training, so a strong causal attribution to position-only representation is hard to sustain. Even so, adding any second attribute improves performance: opacity alone restores 73.5 %, confirming that attribute diversity is essential and that centroid embedding is practically insufficient. Color's non-degenerate marginal contribution is by contrast modest: under the VNStdFeature baseline, adding SH-DC color lifts $E(C)$ from 88.8 % to 92.6 %, a gain of 3.8 percentage points,

whereas the much larger +47-point jump from $E(C)$ to $E(C, \mathrm{SH})$ under VN-In primarily reflects escaping the degenerate centroid embedding rather than color's intrinsic informativeness. On `AeroSplat-4D` itself, the more deployment-relevant setting, $E(X)$ recovers its roughly 8-point margin over $E(C)$ with all four added attributes contributing incremental signal (§6.5.3), at a cost of only about 0.01 M parameters and with no structural change to the classifier as attributes are added.

Aggregation carries the temporal gain; sequential modeling adds a provisional margin. The component ablation (§6.5.5) localizes the temporal contribution to aggregation. Classifying from only the last frame of a sequence gives 59.1 % accuracy on the rotation-augmentation-free Sim-1 evaluation, while mean-pooling over all frames raises this to 77.3 %, an 18.2-point gain from aggregation alone, so the motion trajectory rather than single-frame appearance is discriminative. This does not by itself establish that an explicit sequential model is required: mean-pooling already captures most of the temporal gain, with the Mamba state-space model adding approximately 1.5 percentage points (78.8 %). That margin is positive but provisional, since it falls within the seed-to-seed standard deviation, and the temporal-context sweep (§6.5.8) shows accuracy plateauing by $T = 16$. We therefore state the case for the state-space model primarily on computational grounds and only secondarily on accuracy. Choosing Mamba over Transformer self-attention is driven by Mamba's $O(T)$ scaling against attention's $O(T^2)$ in the number of frames [17], [20]; at the current one-second clips this distinction is secondary, but it grows important as the system extends to multi-object scenarios such as a flock of birds sharing the monitored volume with a drone, where linear-time scaling preserves throughput. Rotation invariance is a secondary benefit here: because each per-frame token is already invariant at the VN-In bridge, the temporal model tolerates independently rotated frames, which the correspondence-free $z/\mathrm{SO}(3)_{\mathrm{pf}}$ protocol stresses directly.

Capacity saturates early. The classifier ablations (§6.5.6, §6.5.7) characterize the compute-accuracy frontier of `MambaSplat-4D` on the rotation-augmentation-free Sim-1 evaluation. Width has a flat effect across 64 to 256, holding the 77 to 79 % band, then degrades at 512 (74.3 %), so capacity saturates at or below 256 dimensions, and over-widening is mildly harmful. Depth peaks more sharply, four Mamba layers reaching 80.3 % at approximately 1.9 M parameters, whereas the shallower one- and two-layer configurations and the deeper eight-layer configuration (76.8 %) all sit several points lower. Temporal modeling beyond four layers therefore provides no benefit, and may slightly hurt generalization, at the current dataset size. This early saturation is consistent with the modest dataset size and the fixed 50-epoch training budget for `MambaSplat-4D`: deeper or wider temporal models add capacity to fit but not necessarily to generalize at this scale.

The helicopter is the hardest class. Per-class accuracy under $E(X)$ reaches 0.63 for the helicopter on Sim-1, recovering substantially from the 0.13 to 0.30 range of the augmentation-only baselines, though helicopter identities show wide confidence intervals across the four test instances (§6.5.3). This is expected, since the helicopter combines an airplane-like forward trajectory with a distinctive rotor signature, making it the most ambiguous class. A subtler failure appears in the centroid-only mode: $E(C)$ leaks invariance, its rank-degenerate bridge frame yielding an argmax flip rate of 8×10^{-4}, small but nonzero, against zero flips under full $E(X)$ (§6.5.9). The learned VN-In bridge therefore underpins invariance stability and accuracy, whereas position-only lifting is structurally fragile.

7.5 RQ4: End-to-End System Performance

What classification accuracy can the integrated pipeline achieve on held-out synthetic test data, both in-distribution and out-of-distribution, and how does it compare to 2D detection baselines? This is the one question the thesis can only partly answer at the time of writing, and the answer splits cleanly by axis. On efficiency the verdict is a clear yes: the integrated pipeline runs at roughly 58.6 ms per frame, inside the sub-100 ms per-frame latency budget of Chapter 2, with the `MambaSplat-4D` classifier contributing only 0.63 to 0.75 ms of that at roughly 1.9 M parameters, 23 to 58 times smaller than the 4D-temporal baselines (§6.6.2). On accuracy, the verdict is mixed: in distribution the pipeline is competitive, with $E(X)$ reaching 78.8% on Sim-1 and 72.2% on Sim-2, but on the Sim-3 OOD probe it does not surpass the 2D detectors (24.6% against WRN-YOLO's 45.83%), a gap that likely reflects their ImageNet-pretrained backbones rather than a limit of the 3D representation (§6.6.1). A direct probe of this attribution, retraining the strongest 2D classifier (YOLO26-cls) from random initialization rather than ImageNet-pretrained weights, failed to converge on `AeroSplat-4D` and never reached usable in-distribution accuracy, which is consistent with the baselines' competitiveness resting on

large-scale pretraining even though a non-converging run cannot by itself separate the out-of-distribution gap from the in-distribution one.

Within that accuracy picture, the clearest result is a negative one: the FBOD-SV baseline collapses to 33.33% accuracy on the four-class Sim-1 problem [138], consistent with near-chance single-class prediction. The collapse may reflect under-tuned retraining at lower resolution rather than a fundamental limitation.

7.6 Limitations and Threats to Validity

The primary threat to external validity is that all evaluation is synthetic. Sim-1, Sim-2, and Sim-3 are generated by the Isaac Sim renderer with domain-randomized lighting, albedo, and camera parameters [118], [119], and the supervised masks are taken from the simulator's ground-truth foreground channel, a clean segmentation upper bound that real-camera footage is unlikely to match. The sim-to-real gap therefore remains unquantified, with domain randomization as the main mitigation and geometry-level transfer as a separate, unresolved problem; validating the pipeline on real multi-camera footage is the principal next step (§8.2).

Reproducibility is parameter-level only, since the RTX rendering pipeline is not pixel-deterministic across runs, so reproducing exact frames requires fixing the random seeds at each rendering step. A further threat is that the four 2D detection baselines (WRN-YOLO, YOLOv7 with CSRT, FBOD-SV [138], and YOLO26-cls) were retrained on the `AeroSplat-4D` training split with hyperparameters matched to each original publication rather than re-tuned by search; such under-tuning may depress their performance, so we read the comparison as a paradigm-level contrast between 2D detection and 4D temporal classification rather than a claim of per-architecture superiority.

7.7 Practical and Deployment Implications

Latency, not accuracy, is the binding deployment constraint, and it must be read along two distinct axes. On a per-frame basis the system meets its budget: `DepthSplat-OC` reconstruction dominates at 52.34 ms per frame, while the Stage-3 `MambaSplat-4D` classifier contributes only 0.63 to 0.75 ms per frame (§6.6.2), so the live end-to-end pipeline costs 58.59 ms per frame, within the below-100 ms per-frame latency budget of Chapter 2, and drops to roughly 6.2 ms per frame once previously computed Gaussian reconstructions are cached for a static scene. The binding constraint is instead throughput: at 58.59 ms per frame, the live pipeline sustains roughly 17 FPS, below the 30 FPS capture rate, so real-time throughput over a 24-frame clip (1.41 s of wall-clock live, 0.15 s cached) is achieved only with cached reconstructions. The linear-time Mamba encoder was chosen partly for this profile, keeping the classifier's contribution negligible as frame and object counts grow.

Intended use cases for `AeroSplat-4D`-based systems include airport perimeter monitoring, critical-infrastructure protection, prisons, and wildlife monitoring. As declared in the scope of this thesis (Chapter 1), the privacy and legal aspects of such surveillance, together with adversarial evasion, lie beyond this technical contribution. The intended edge target is the NVIDIA Jetson AGX Orin, on which no benchmarking has yet been conducted; the Stage-3 classifier's approximately 1.4 GB peak footprint (§6.6.2) fits the Jetson's memory envelope in principle, but on-device latency characterization is required before deployment feasibility can be claimed, and the approximately 70 % four-class accuracy on Sim-1, against a 25 % chance baseline, does not yet establish real-camera utility. The contributions are nonetheless built for reuse beyond this pipeline: the VN linear layers at the core of `MambaSplat-4D` are general-purpose equivariant building blocks applicable to any 3D point representation [29], and the modular orchestrator exposes typed stage interfaces so that individual stages can be replaced or upgraded independently while reusing cached intermediate artifacts.

8 Conclusions and Future Work

This chapter concludes the thesis by summarizing the main findings and outlining directions for future work.

8.1 Conclusions

This thesis asked whether a ground-based network of synchronized RGB cameras, rather than dedicated counter-UAV hardware, can distinguish flying objects, and answered it by building an end-to-end pipeline that reconstructs each object in 3D from several views and then classifies it as a drone, bird, helicopter, or airplane from how its reconstructed 3D shape moves over time. On held-out synthetic data, the classifier reaches 70.2% four-class accuracy under arbitrary rotation, with that rotation robustness built into the architecture rather than learned from augmented data. On a single GPU, the live pipeline runs at roughly 17 FPS (58.6 ms per frame). All results are obtained in simulation, and real-camera evaluation is deferred to Future Work (§8.2).

The RGB multi-view front-end was a deliberate choice: cameras are passive, inexpensive, and information-dense, and multi-camera reconstruction recovers the range that a single short-baseline camera cannot. Framed this way, the thesis charts a multi-view spatial-temporal paradigm for flying-object classification that, to the best of our knowledge, has not been attempted before: instead of the established 2D appearance-and-detection route, it classifies objects from the spatial geometric features and temporal dynamics of a reconstructed 4D Gaussian representation (§7.4). It establishes the classification half of the problem while utilizing per-camera foreground masks, taken here as simulator ground truth; the detection half, an RGB-only multi-view foreground-segmentation front-end with detection and localization built on top, is a separate computer-vision problem left to future work given the thesis timespan (§8.2).

One methodological pivot shaped the system: optimization-based 3D Gaussian Splatting requires slow per-scene fitting from sparse views, so the pipeline instead uses feed-forward 3D Gaussian Splatting, which brought the entire pipeline within the latency budget.

The four research questions posed in Chapter 1 are answered in full, drawing each verdict only from the results of this thesis, in the Discussion (Chapter 7); the verdicts for RQ1 through RQ4 all rest on completed experiments. Table 8.1 consolidates those verdicts at a glance, and the remainder of this section draws their threads together rather than restating each in turn.

Returning to the central question of Chapter 1 (§1.2), whether flying objects can be reconstructed and classified in 3D from synchronized RGB feeds alone while exploiting the temporal dynamics of a 3D Gaussian representation, this thesis answers it with a qualified yes. The pipeline reconstructs and classifies all four classes from RGB views alone, and the temporal claim is settled decisively in simulation: aggregating a clip improves accuracy over classifying any single frame, and the rotation-invariant 4D representation surpasses static positions-only and augmentation-reliant baselines under arbitrary rotation (§6.5.5, §6.5.2, §6.5.3). Real-camera validation remains the open step, since all evaluation here is synthetic (§8.2).

Taken together, and reading each stage's verdict only against this thesis's own measurements (§7.2–§7.5), the completed results (RQ1 through RQ3) establish the core academic contribution of this thesis: in simulation, a compact ∼1.9 M-parameter rotation-invariant 4D Gaussian classifier separates flying objects with structural, rather than merely learned, robustness to arbitrary inter-frame rotation, a guarantee the augmentation-reliant

baselines cannot match and the property on which the pipeline most clearly surpasses prior work. The recurring pattern across the stages is the same: each adaptation we introduced, from mask-gating the reconstructor to lifting the classifier into a rotation-invariant 4D representation, favored the approach over the unadapted alternative and improved result quality. Therefore, our overall conclusion is that adapting feed-forward 3DGS and a rotation-invariant 4D classifier to multi-view flying-object recognition was the decisive design choice. The end-to-end accuracy result (RQ4) is competitive in distribution but does not yet close the out-of-distribution gap, so validating the pipeline on real multi-camera footage is the principal direction for future work (§8.2).

8.1.1 Key Contributions

The key contributions of this thesis are the following:

- `AeroSplat-4D`. A synthetic multi-camera dataset with animated assets and an Isaac Sim generation pipeline for complete simulations.
- `DepthSplat-OC`. An object-centric feed-forward 3DGS reconstructor for small, distant aerial targets that operates on an arbitrary number of cameras for any kind of scenario.
- `MambaSplat-4D`. A rotation-invariant 4D Gaussian classifier, the core academic contribution of this thesis.
- `End-to-end system`. An integration that composes all stages (Stages 0–4) and will be released as open source.

8.1.2 Per-stage design verdicts

Table 8.1 states, stage by stage, whether each central design decision was a good one, with every verdict and its supporting number drawn solely from the experiments of Chapter 6. Where an absolute claim is not warranted, we record the decision as competitive within the thesis's compute and latency budget rather than overstating it.

Table 8.1: Per-stage verdict on the central design decisions of the `AeroSplat-4D` pipeline. Each verdict is stated as a direct yes or, where an absolute claim is unwarranted, as competitive within the thesis's compute and latency budget. All evidence is drawn from the experiments of Chapter 6.

Stage	Design decision	Verdict	Evidence
0	Isaac Sim multi-camera four-class dataset with full ground truth	Yes	No surveyed dataset pairs wide-baseline multi-view capture, four near-balanced classes ($H/H_{\max} > 0.95$), and 3DGS-ready ground truth; this pairing enables every downstream experiment (§6.3.1).
1	Simulator ground-truth foreground masks	Yes (by design)	A deliberate clean upper bound that isolates Stages 2–3 from segmentation error; a learned RGB-only segmentation front-end is scoped to future work (§8.2).
2	`DepthSplat-OC` feed-forward reconstruction	Competitive (given compute)	+3.2 dB PSNR over the same-tier LGM (24.65 versus 21.44 on GSO) at ∼27 GPU-hours; GS-LRM scores higher only at ∼114× the training compute. Not state-of-the-art in absolute PSNR, strongest in its compute tier (§6.4.2).
3	`MambaSplat-4D` rotation-invariant 4D classifier	Yes	The only structurally SO(3)-invariant method (0 flips / 9600 forward passes); 70.2% under arbitrary rotation versus 58.2% for an augmented Mamba4D, at 23–58× fewer parameters (§6.5.2, §6.5.3, §6.5.9, §6.6.2).
4	End-to-end integration	Competitive (given budget)	58.6 ms per frame, inside the sub-100 ms budget; competitive in-distribution accuracy, below the 2D detectors out-of-distribution (§6.6.1, §6.6.2).

8.2 Future Work

Several directions extend this work. An RGB-only multi-view foreground segmentation front-end using a track-before-detect design [40], [43] (per-camera motion cues, a $\geq k$-camera voxel-vote consensus, and vote back-projection to refined 2D masks) was prototyped and is preserved in Appendix C; open questions remain around handling moving clutter and the sensitivity of the consensus threshold to the camera count. Object detection

and localization can be built on top of this multi-view foreground segmentation front-end. Extending the system to multiple simultaneous objects is the natural follow-on: rather than a single unioned foreground mask, the front-end would emit per-instance masks; each instance would be reconstructed separately; detections would be linked across the camera views and across frames into identity-consistent tracks, with explicit handling of occlusion and identity switches; and the classifier would run per identity in place of the current one-label-per-clip design. Sim-to-real evaluation would validate the full pipeline on a real multi-camera test bench. Online camera-extrinsic refinement relaxes the fixed-calibration assumption when the extrinsics drift over time. Texture and material domain randomization would extend the simulator's domain-randomization coverage [119]. Edge and on-device validation on the NVIDIA Jetson AGX Orin would report GFLOPs and memory (RAM) usage as the proxy metric for edge-deployment feasibility. Explainability diagnostics for the 4D classifier would combine an empirical equivariance-error metric over Haar-uniform rotations, per-point and per-frame gradient-saliency attribution [157], and RISE deletion/insertion fidelity curves [158]. Retraining `DepthSplat-OC` from a fine-tuned monodepth initialization is another direction: a standalone monodepth fine-tune already exists as an unpublished preliminary result (validation $d_1 = 97.1\%$, AbsRel $= 5.7\%$, not otherwise reported in this thesis), and re-running training with this checkpoint as the monodepth initialization would require re-running the Stage-3 and Stage-4 tables (§6.5–§6.6); the cost volume and Gaussian head are tightly coupled to the depth backbone (§6.4.1), so a full retrain, not an inference-time hot-swap, is required. Pre-training the VN-Transformer on ShapeSplat [96], [159] would teach general 3D shape priors before fine-tuning on the flying-object classes. Feeding a 3D trajectory as a feature to the VN-Transformer would feed each object's accumulating 3D position and speed to estimate its future direction, a capability the VN-Transformer already supports. Point-cloud segmentation of the reconstructed object would separate an attached external payload from its carrier, flagging whether a drone is transporting a package, a security-relevant capability bounded by the reconstruction resolution available at closer range. Environmental variation would broaden the range of simulated conditions to include lighting (sun, clouds), weather (haze, rain), and horizon clutter. Temporary occlusions, such as clouds, other aircraft, and fixed obstacles, would test robustness under transient visibility loss. Thermal and night-vision capture would add a sensing modality beyond daylight RGB for both training and inference [160]. More physically realistic asset animations, such as bird flight patterns, drone maneuvers, and smooth helicopter curves at varied animation speeds, would improve the fidelity of the synthetic motion. Rolling-shutter modeling remains acknowledged but unmodeled in the current simulation.

Bibliography

[1] Reuters. "Drone sightings disrupt flights at Copenhagen, Oslo airports," Accessed: May 29, 2026. [Online]. Available: https://www.reuters.com/business/aerospace-defense/copenhagen-airport-halts-traffic-due-drone-sightings-police-says-2025-09-22/.

[2] The New York Times. "In UK prisons, drones fly in contraband 'As if by Uber Eats'," Accessed: May 29, 2026. [Online]. Available: https://www.nytimes.com/2026/04/01/world/europe/uk-prisons-contraband-drones.html.

[3] A. Coluccia et al., "Drone vs. Bird Detection: Deep learning algorithms and results from a grand challenge," en, *Sensors*, vol. 21, no. 8, p. 2824, Apr. 2021, ISSN: 1424-8220. DOI: 10.3390/s21082824.

[4] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting flying objects using a single moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 879–892, 2017, Source paper for the FL-Drones (EPFL) dataset. DOI: 10.1109/TPAMI.2016.2564408.

[5] NVIDIA. "Isaac Sim." original-date: 2025-05-28T18:38:18Z, Accessed: Dec. 15, 2025. [Online]. Available: https://github.com/isaac-sim/IsaacSim.

[6] A. Coluccia et al., "The Drone-vs-Bird Detection Grand Challenge at IJCNN 2025," in *2025 International Joint Conference on Neural Networks (IJCNN)*, ISSN: 2161-4407, Jun. 2025, pp. 1–8. DOI: 10.1109/IJCNN64981.2025.11228314.

[7] IEC. "IEC 62676 – Video Surveillance," Accessed: Dec. 3, 2025. [Online]. Available: https://www.imatest.com/imaging/iec-62676/.

[8] C. Shannon, "Communication in the Presence of Noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949, ISSN: 2162-6634. DOI: 10.1109/JRPROC.1949.232969.

[9] T. J. Ma and R. J. Anderson, "Remote Sensing Low Signal-to-Noise-Ratio Target Detection Enhancement," en, *Sensors*, vol. 23, no. 6, p. 3314, Jan. 2023, Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s23063314.

[10] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991. DOI: 10.1109/26.103043.

[11] IEEE, *IEEE standard for a precision clock synchronization protocol for networked measurement and control systems*, IEEE Std 1588-2019, 2020. DOI: 10.1109/IEEESTD.2020.9120376.

[12] NVIDIA Corporation. "Embedded systems developer kits & modules from NVIDIA Jetson," Accessed: Jun. 4, 2026. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/.

[13] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2004, ISBN: 0521540518. DOI: 10.1017/CBO9780511811685.

[14] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA, USA: MIT Press, 1993, ISBN: 9780262061582.

[15] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," in *ACM Transactions on Graphics*, vol. 42, ACM, 2023. DOI: 10.1145/3592433.

[16] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "Ewa splatting," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, IEEE, 2002, pp. 223–238. DOI: 10.1109/TVCG.2002.1021576.

[17] A. Vaswani et al., "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. Guide Proceedings, Dec. 2017, pp. 6000–6010, ISBN: 978-1-5108-6096-4. DOI: 10.5555/3295222.3295349.

[18] N. Thomas et al., *Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds*, arXiv:1802.08219 [cs], May 2018. DOI: 10.48550/arXiv.1802.08219.

[19] M. Zaheer et al., "Deep Sets," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. Guide Proceedings, Dec. 2017, pp. 3394–3404, ISBN: 978-1-5108-6096-4. DOI: 10.5555/3294996.3295098.

[20] A. Gu and T. Dao, *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*, arXiv:2312.00752 [cs.LG], May 2024. DOI: 10.48550/arXiv.2312.00752.

[21] J. L. Schonberger and J.-M. Frahm, "Structure-from-Motion Revisited," en, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 4104–4113, ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.445.

[22] J. L. SchÃnberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise View Selection for Unstructured Multi-View Stereo," en, in *Computer Vision " ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 501–518, ISBN: 978-3-319-46487-9. DOI: 10.1007/978-3-319-46487-9_31.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. DOI: 10.1023/b:visi.0000029664.99615.94.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain: IEEE, 2011, pp. 2564–2571. DOI: 10.1109/iccv.2011.6126544.

[25] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. DOI: 10.1109/34.888718.

[26] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987. DOI: 10.1109/JRA.1987.1087109.

[27] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 150–162, 1994. DOI: 10.1109/34.273735.

[28] K. N. Kutulakos and S. M. Seitz, "A Theory of Shape by Space Carving," en, *International Journal of Computer Vision*, vol. 38, no. 3, pp. 199–218, Jul. 2000, ISSN: 1573-1405. DOI: 10.1023/A:1008191222954.

[29] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. Guibas, "Vector Neurons: A General Framework for SO(3)-Equivariant Networks," en, in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 12 180–12 189, ISBN: 978-1-6654-2812-5. DOI: 10.1109/ICCV48922.2021.01198.

[30] T. Fei, L. Bi, J. Gao, S. Chen, and G. Zhang, "MVSGS: Gaussian splatting radiation field enhancement using multi-view stereo," en, *Complex & Intelligent Systems*, vol. 11, no. 1, p. 80, Dec. 2024, ISSN: 2198-6053. DOI: 10.1007/s40747-024-01691-x.

[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[32] A. Dosovitskiy et al., *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv:2010.11929 [cs.CV], Jun. 2021. DOI: 10.48550/arXiv.2010.11929.

[33] F. B. Fuchs et al., "SE(3)-transformers," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. Guide Proceedings, Dec. 2020, pp. 1970–1981, ISBN: 978-1-7138-2954-6. DOI: 10.5555/3495724.3495890.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[35] M. Chaman, A. E. Maliki, H. Dahou, and A. Hadjoudja, "Benchmarking YOLO-based deep learning models for real-time object detection in hybrid ADAS and intelligent transportation systems," en, *Results in Engineering*, vol. 29, p. 108 942, Mar. 2026, ISSN: 2590-1230. DOI: 10.1016/j.rineng.2025.108942.

[36] R. Sapkota et al., "YOLO advances to its genesis: A decadal and comprehensive review of the You Only Look Once (YOLO) series," en, *Artificial Intelligence Review*, vol. 58, no. 9, Jun. 2025, ISSN: 1573-7462. DOI: 10.1007/s10462-025-11253-3.

[37] Y. Zhao et al., "DETRs beat YOLOs on real-time object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 16 965–16 974. DOI: 10.1109/cvpr52733.2024.01605. arXiv: 2304.08069.

[38] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review," en, *Image and Vision Computing*, vol. 97, p. 103 910, May 2020, ISSN: 0262-8856. DOI: 10.1016/j.imavis.2020.103910.

[39] J. Wang, W. Yang, H. Guo, R. Zhang, and G.-S. Xia, "Tiny object detection in aerial images," en, in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, Jan. 2021, pp. 3791–3798. DOI: 10.1109/ICPR48806.2021.9413340.

[40] Y. Barniv, "Dynamic Programming Solution for Detecting Dim Moving Targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-21, no. 1, pp. 144–156, Jan. 1985, ISSN: 1557-9603. DOI: 10.1109/TAES.1985.310548.

[41] S. J. Davey, M. G. Rutten, and B. Cheung, "A comparison of detection performance for several Track-before-Detect algorithms," en, *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 1, p. 428 036, 2008, ISSN: 1687-6180. DOI: 10.1155/2008/428036.

[42] E. Grossi, M. Lops, and L. Venturino, "A novel dynamic programming algorithm for Track-Before-Detect in radar systems," en, *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2608–2619, May 2013, ISSN: 1941-0476. DOI: 10.1109/TSP.2013.2251338.

[43] S. M. Tonissen and R. J. Evans, "Performance of dynamic programming techniques for Track-Before-Detect," en, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 4, pp. 1440–1451, 1996, Title corrected from publisher record (original IEEE deposit reads "Peformance")., ISSN: 0018-9251. DOI: 10.1109/7.543865.

[44] V. Magoulianitis, D. Ataloglou, A. Dimou, D. Zarpalas, and P. Daras, "Does Deep Super-Resolution Enhance UAV Detection?" en, in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Taipei, Taiwan: IEEE, Sep. 2019, pp. 1–6, ISBN: 978-1-7281-0990-9. DOI: 10.1109/AVSS.2019.8909865.

[45] Y. Sun et al., "Enhancing UAV Detection in Surveillance Camera Videos through Spatiotemporal Information and Optical Flow," en, *Sensors*, vol. 23, no. 13, p. 6037, Jun. 2023, ISSN: 1424-8220. DOI: 10.3390/s23136037.

[46] Y. Hou, L. Zheng, and S. Gould, "Multiview detection with feature perspective transformation," en, in *Computer Vision – ECCV 2020*, ser. Lecture Notes in Computer Science, Springer, 2020, pp. 1–18, ISBN: 9783030585716. DOI: 10.1007/978-3-030-58571-6_1.

[47] C. Qiu, Z. Zhang, H. Lu, and H. Luo, "A survey of motion-based multitarget tracking methods," en, *Progress In Electromagnetics Research B*, vol. 62, pp. 195–223, 2015, ISSN: 1937-6472. DOI: 10.2528/pierb15010503.

[48] A. Rozantsev, S. N. Sinha, D. Dey, and P. Fua, "Flight Dynamics-Based Recovery of a UAV Trajectory Using Ground Cameras," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jul. 2017, pp. 2482–2491. DOI: 10.1109/CVPR.2017.266.

[49] J. Li, J. Murray, D. Ismaili, K. Schindler, and C. Albl, *Reconstruction of 3D flight trajectories from ad-hoc camera networks*, en, arXiv:2003.04784 [cs], Jul. 2020. DOI: 10.48550/arXiv.2003.04784.

[50] J. Rosner et al., "Multimodal dataset for indoor 3D drone tracking," en, *Scientific Data*, vol. 12, no. 1, p. 257, Feb. 2025, ISSN: 2052-4463. DOI: 10.1038/s41597-025-04521-y.

[51] W. Lindenheim-Locher et al., "YOLOv5 Drone Detection Using Multimodal Data Registered by the Vicon System," en, *Sensors*, vol. 23, no. 14, p. 6396, Jul. 2023, ISSN: 1424-8220. DOI: 10.3390/s23146396.

[52] S. Yuan et al., "MMAUD: A Comprehensive Multi-Modal Anti-UAV Dataset for Modern Miniature Drone Threats," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 2745–2751. DOI: 10.1109/ICRA57147.2024.10610957.

[53] T. Deng et al., *Multi-Modal UAV Detection, Classification and Tracking Algorithm – Technical Report for CVPR 2024 UG2 Challenge*, en, arXiv:2405.16464 [cs], May 2024. DOI: 10.48550/arXiv.2405.16464.

[54] J. Philion and S. Fidler, "Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D," en, in *Computer Vision " ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., vol. 12359, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 194–210, ISBN: 978-3-030-58568-6. DOI: 10.1007/978-3-030-58568-6_12.

[55] Z. Li et al., "BEVFormer: Learning Birds-Eye-View Representation From LiDAR-Camera via Spatiotemporal Transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 3, pp. 2020–2036, Mar. 2025, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2024.3515454.

[56] Z. Liu et al., *BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation*, arXiv:2205.13542 [cs], Sep. 2024. DOI: 10.48550/arXiv.2205.13542.

[57] Z. Lin et al., "RCBEVDet: Radar-Camera Fusion in Bird's Eye View for 3D Object Detection," en, in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2024, pp. 14 928–14 937, ISBN: 979-8-3503-5300-6. DOI: 10.1109/CVPR52733.2024.01414.

[58] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "LoFTR: Detector-free local feature matching with transformers," en, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, arXiv: 2104.00680, IEEE, Jun. 2021, pp. 8918–8927. DOI: 10.1109/CVPR46437.2021.00881.

[59] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan, "Image matching from handcrafted to deep features: A survey," en, *International Journal of Computer Vision*, vol. 129, no. 1, pp. 23–79, 2021, ISSN: 1573-1405. DOI: 10.1007/s11263-020-01359-2.

[60] K. Cheng et al., *GaussianPro: 3D Gaussian Splatting with Progressive Propagation*, arXiv:2402.14650 [cs], Feb. 2024. DOI: 10.48550/arXiv.2402.14650.

[61] S. Szymanowicz, C. Rupprecht, and A. Vedaldi, "Splatter Image: Ultra-Fast Single-View 3D Reconstruction," en, in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2024, pp. 10 208–10 217, ISBN: 979-8-3503-5300-6. DOI: 10.1109/CVPR52733.2024.00972.

[62] D. Charatan, S. L. Li, A. Tagliasacchi, and V. Sitzmann, "PixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction," en, in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2024, pp. 19 457–19 467, ISBN: 979-8-3503-5300-6. DOI: 10.1109/CVPR52733.2024.01840.

[63] H. Xu et al., *DepthSplat: Connecting Gaussian Splatting and Depth*, arXiv:2410.13862 [cs], Mar. 2025. DOI: 10.48550/arXiv.2410.13862.

[64] Y. Chen et al., "MVSplat: Efficient 3D Gaussian Splatting from Sparse Multi-view Images," en, in *Computer Vision " ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., Cham: Springer Nature Switzerland, 2025, pp. 370–386, ISBN: 978-3-031-72664-4. DOI: 10.1007/978-3-031-72664-4_21.

[65] S. Zheng et al., "GPS-Gaussian: Generalizable Pixel-Wise 3D Gaussian Splatting for Real-Time Human Novel View Synthesis," en, in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2024, pp. 19 680–19 690, ISBN: 979-8-3503-5300-6. DOI: 10.1109/CVPR52733.2024.01861.

[66] L. Yang et al., "Depth Anything V2," in *Advances in Neural Information Processing Systems*, A. Globerson et al., Eds., vol. 37, Curran Associates, Inc., 2024, pp. 21 875–21 911. DOI: 10.52202/079017-0688.

[67] Y. Hong et al., "Lrm: Large reconstruction model for single image to 3d," en, 2024. arXiv: 2311.04400 [cs.CV].

[68] J. Li et al., "Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model," en, 2023. arXiv: 2311.06214 [cs.CV].

[69] J. Tang, Z. Chen, X. Chen, T. Wang, G. Zeng, and Z. Liu, "LGM: Large Multi-view Gaussian Model for High-Resolution 3D Content Creation," en, in *Computer Vision - ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., vol. 15062, Series Title: Lecture Notes in Computer Science, Cham: Springer Nature Switzerland, 2025, pp. 1–18, ISBN: 978-3-031-73235-5. DOI: 10.1007/978-3-031-73235-5_1.

[70] K. Zhang et al., *GS-LRM: Large Reconstruction Model for 3D Gaussian Splatting*, arXiv:2404.19702 [cs], Apr. 2024. DOI: 10.48550/arXiv.2404.19702.

[71] X. Long et al., "Wonder3D: Single Image to 3D Using Cross-Domain Diffusion," en, in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2024, pp. 9970–9980, ISBN: 979-8-3503-5300-6. DOI: 10.1109/CVPR52733.2024.00951.

[72] R. Shi et al., *Zero123++: A Single Image to Consistent Multi-view Diffusion Base Model*, arXiv:2310.15110 [cs.CV], Oct. 2023. DOI: 10.48550/arXiv.2310.15110.

[73] G. Wu et al., *4D Gaussian Splatting for Real-Time Dynamic Scene Rendering*, arXiv:2310.08528 [cs], Jul. 2024. DOI: 10.48550/arXiv.2310.08528.

[74] Y. Duan, F. Wei, Q. Dai, Y. He, W. Chen, and B. Chen, *4D-Rotor Gaussian Splatting: Towards Efficient Novel View Synthesis for Dynamic Scenes*, en, arXiv:2402.03307 [cs], Jul. 2024. DOI: 10.48550/arXiv.2402.03307.

[75] S. Oh, Y. Lee, H. Jeon, and E. Park, *Hybrid 3D-4D Gaussian Splatting for Fast Dynamic Scene Representation*, arXiv:2505.13215 [cs], May 2025. DOI: 10.48550/arXiv.2505.13215.

[76] Y. Yuan, Q. Shen, X. Yang, and X. Wang, *1000+ FPS 4D Gaussian Splatting for Dynamic Scene Rendering*, arXiv:2503.16422 [cs], Mar. 2025. DOI: 10.48550/arXiv.2503.16422.

[77] J. Ren et al., "L4GM: Large 4D Gaussian Reconstruction Model," in *Advances in Neural Information Processing Systems*, A. Globerson et al., Eds., vol. 37, Curran Associates, Inc., 2024, pp. 56 828–56 858. DOI: 10.52202/079017-1810.

[78] H. Liang et al., *Feed-Forward Bullet-Time Reconstruction of Dynamic Scenes from Monocular Videos*, arXiv:2412.03526 [cs.CV], Sep. 2025. DOI: 10.48550/arXiv.2412.03526.

[79] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, ISSN: 2381-8549, Sep. 2016, pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

[80] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, ISSN: 2381-8549, Sep. 2017, pp. 3645–3649. DOI: 10.1109/ICIP.2017.8296962.

[81] Y. Zhang et al., "ByteTrack: Multi-object Tracking by~ Associating Every Detection Box," en, in *Computer Vision " ECCV 2022*, S. Avidan, G. Brostow, M. Ciss, G. M. Farinella, and T. Hassner, Eds., Cham: Springer Nature Switzerland, 2022, pp. 1–21, ISBN: 978-3-031-20047-2. DOI: 10.1007/978-3-031-20047-2_1.

[82] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking," en, in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada: IEEE, Jun. 2023, pp. 9686–9696, ISBN: 979-8-3503-0129-8. DOI: 10.1109/CVPR52729.2023.00934.

[83] T. Yamane, R. Masumura, S. Suzuki, and S. Orihashi, *MVTrajecter: Multi-View Pedestrian Tracking with Trajectory Motion Cost and Trajectory Appearance Cost*, en, Version Number: 1, 2025. DOI: 10.48550/ARXIV.2509.01157.

[84] T. Chavdarova et al., "WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection," en, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 5030–5039, ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00528.

[85] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," en, in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., vol. 12346, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 213–229, ISBN: 978-3-030-58451-1 978-3-030-58452-8. DOI: 10.1007/978-3-030-58452-8_13.

[86] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "TrackFormer: Multi-Object Tracking with Transformers," en, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 8834–8844, ISBN: 978-1-6654-6946-3. DOI: 10.1109/CVPR52688.2022.00864.

[87] Y. Zhang, T. Wang, and X. Zhang, "MOTRv2: Bootstrapping End-to-End Multi-Object Tracking by Pretrained Object Detectors," en, in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada: IEEE, Jun. 2023, pp. 22 056–22 065, ISBN: 979-8-3503-0129-8. DOI: 10.1109/CVPR52729.2023.02112.

[88] X. Weng, J. Wang, D. Held, and K. Kitani, "Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics," *ECCVW*, 2020. arXiv: 2008.08063 [cs.CV].

[89] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D Object Detection and Tracking," en, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 11 779–11 788, ISBN: 978-1-6654-4509-2. DOI: 10.1109/CVPR46437.2021.01161.

[90] Z. Pang, J. Li, P. Tokmakov, D. Chen, S. Zagoruyko, and Y.-X. Wang, "Standing Between Past and Future: Spatio-Temporal Modeling for Multi-Camera 3D Multi-Object Tracking," en, in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada: IEEE, Jun. 2023, pp. 17 928–17 938, ISBN: 979-8-3503-0129-8. DOI: 10.1109/CVPR52729.2023.01719.

[91] T. Zhang, X. Chen, Y. Wang, Y. Wang, and H. Zhao, *MUTR3D: A Multi-camera Tracking Framework via 3D-to-2D Queries*, arXiv:2205.00613 [cs], May 2022. DOI: 10.48550/arXiv.2205.00613.

[92] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," en, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 77–85, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.16.

[93] C. R. Qi et al., "PointNet++," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. Guide Proceedings, Dec. 2017, pp. 5105–5114, ISBN: 978-1-5108-6096-4. DOI: 10.5555/3295222.3295263.

[94] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, *Point Transformer*, arXiv:2012.09164 [cs], Sep. 2021. DOI: 10.48550/arXiv.2012.09164.

[95] X. Wu et al., *Point Transformer V3: Simpler, Faster, Stronger*, arXiv:2312.10035 [cs], Mar. 2024. DOI: 10.48550/arXiv.2312.10035.

[96] Q. Ma et al., *ShapeSplat: A Large-scale Dataset of Gaussian Splats and Their Self-Supervised Pretraining*, arXiv:2408.10906 [cs], Sep. 2025. DOI: 10.48550/arXiv.2408.10906.

[97] E. Wagstaff et al., "Universal approximation of functions on sets," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 6762–6817, Jan. 2022. DOI: 10.5555/3586589.3586740.

[98] R. L. Murphy, B. Srinivasan, V. A. Rao, and B. Ribeiro, "Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs," *CoRR*, vol. abs/1811.01900, 2018. arXiv: 1811.01900.

[99] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer," *CoRR*, vol. abs/1810.00825, 2018. arXiv: 1810.00825.

[100] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira, "Perceiver: General perception with iterative attention," *CoRR*, vol. abs/2103.03206, 2021. arXiv: 2103.03206.

[101] G. Bertasius, H. Wang, and L. Torresani, *Is Space-Time Attention All You Need for Video Understanding?* arXiv:2102.05095 [cs], Jun. 2021. DOI: 10.48550/arXiv.2102.05095.

[102] D. Liang et al., "PointMamba: A Simple State Space Model for Point Cloud Analysis," in *Advances in Neural Information Processing Systems*, A. Globerson et al., Eds., vol. 37, Curran Associates, Inc., 2024, pp. 32 653–32 677. DOI: 10.52202/079017-1026.

[103] X. Han, Y. Tang, Z. Wang, and X. Li, "Mamba3D: Enhancing Local Features for 3D Point Cloud Analysis via State Space Model," en, in *Proceedings of the 32nd ACM International Conference on Multimedia*, Melbourne VIC Australia: ACM, Oct. 2024, pp. 4995–5004, ISBN: 979-8-4007-0686-8. DOI: 10.1145/3664647.3681173.

[104] J. Liu et al., "Mamba4D: Efficient 4D Point Cloud Video Understanding with Disentangled Spatial-Temporal State Space Models," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 2575-7075, Jun. 2025, pp. 17 626–17 636. DOI: 10.1109/CVPR52734.2025.01642.

[105] Y.-L. Liao and T. Smidt, *Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs*, arXiv:2206.11990 [cs], Feb. 2023. DOI: 10.48550/arXiv.2206.11990.

[106] S. Assaad, C. Downey, R. Al-Rfou, N. Nayakanti, and B. Sapp, *VN-Transformer: Rotation-Equivariant Attention for Vector Neurons*, arXiv:2206.04176 [cs], Jan. 2023. DOI: 10.48550/arXiv.2206.04176.

[107] S. Rahman and D. A. Robertson, "Radar micro-Doppler signatures of drones and birds at K-band and W-band," *Scientific Reports*, vol. 8, no. 1, p. 17 396, 2018, ISSN: 2045-2322. DOI: 10.1038/s41598-018-35880-9.

[108] P. Molchanov, R. I. A. Harmanny, J. J. M. d. Wit, K. Egiazarian, and J. Astola, "Classification of small UAVs and birds by micro-Doppler signatures," en, *International Journal of Microwave and Wireless Technologies*, vol. 6, no. 3-4, pp. 435–444, Jun. 2014, ISSN: 1759-0787, 1759-0795. DOI: 10.1017/S1759078714000282.

[109] S. Rahman and D. A. Robertson, "Classification of drones and birds using convolutional neural networks applied to radar micro-Doppler spectrogram images," en, *IET Radar, Sonar & Navigation*, vol. 14, no. 5, pp. 653–661, 2020, _eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-rsn.2019.0493, ISSN: 1751-8792. DOI: 10.1049/iet-rsn.2019.0493.

[110] F. C. Akyon, E. Akagunduz, S. O. Altinuc, and A. Temizel, *Sequence Models for Drone vs Bird Classification*, arXiv:2207.10409 [cs], Dec. 2022. DOI: 10.48550/arXiv.2207.10409.

[111] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," en, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2018, pp. 6450–6459. DOI: 10.1109/CVPR.2018.00675.

[112] J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," en, in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019, pp. 7082–7092. DOI: 10.1109/ICCV.2019.00718.

[113] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "CARLA: an open urban driving simulator," *CoRR*, vol. abs/1711.03938, 2017. arXiv: 1711.03938.

[114] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *CoRR*, vol. abs/1705.05065, 2017. arXiv: 1705.05065.

[115] M. Savva et al., "Habitat: A platform for embodied AI research," *CoRR*, vol. abs/1904.01201, 2019. arXiv: 1904.01201.

[116] K. Greff et al., "Kubric: A scalable dataset generator," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 2575-7075, Jun. 2022, pp. 3739–3751. DOI: 10.1109/CVPR52688.2022.00373.

[117] M. Denninger et al., *BlenderProc*, 2019. arXiv: 1911.01911.

[118] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Sep. 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133.

[119] J. Tremblay et al., "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," *CoRR*, vol. abs/1804.06516, 2018. arXiv: 1804.06516.

[120] H. Xu et al., "Unifying flow, stereo and depth estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 941–13 958, 2023. DOI: 10.1109/TPAMI.2023.3298645. arXiv: 2211.05783 [cs.CV].

[121] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12 179–12 188. DOI: 10.1109/ICCV48922.2021.01196. arXiv: 2103.13413 [cs.CV].

[122] Y. Xu et al., "GRM: Large Gaussian Reconstruction Model for Efficient 3D Reconstruction and Generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. arXiv: 2403.14621.

[123] L. Downs et al., "Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 2553–2560. arXiv: 2204.11918.

[124] J. Collins et al., "ABO: Dataset and Benchmarks for Real-World 3D Object Understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 21 126–21 136. arXiv: 2110.06199.

[125] M. Deitke et al., "Objaverse: A universe of annotated 3D objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 13 142–13 153. DOI: 10.1109/CVPR52729.2023.01263. arXiv: 2212.08051 [cs.CV].

[126] C. Lin et al., *Objaverse++: Curated 3D Object Dataset with Quality Annotations*, arXiv:2504.07334 [cs] version: 1, Apr. 2025. DOI: 10.48550/arXiv.2504.07334.

[127] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 586–595. DOI: 10.1109/CVPR.2018.00068. arXiv: 1801.03924 [cs.CV].

[128] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019. arXiv: 1711.05101 [cs.LG].

[129] M. Deitke et al., "Objaverse-xl: A universe of 10m+ 3d objects," 2023. arXiv: 2307.05663 [cs.CV].

[130] K. Shoemake, "Uniform random rotations," in *Graphics Gems III*, D. Kirk, Ed., San Diego, CA, USA: Academic Press, 1992, pp. 124–132. DOI: 10.1016/B978-0-08-050755-2.50036-1.

[131] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022. arXiv: 2205.14135 [cs.LG].

[132] Z. Wu et al., "3D ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920. DOI: 10.1109/CVPR.2015.7298801.

[133] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3D points," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops (CVPRW)*, 2010, pp. 9–14. DOI: 10.1109/CVPRW.2010.5543273.

[134] X. Liu, M. Yan, and J. Bohg, "MeteorNet: Deep learning on dynamic 3D point cloud sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. arXiv: 1910.09165.

[135] H. Fan, Y. Yang, and M. Kankanhalli, "Point 4D Transformer Networks for Spatio-Temporal Modeling in Point Cloud Videos," en, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 14 199–14 208, ISBN: 978-1-6654-4509-2. DOI: 10.1109/CVPR46437.2021.01398.

[136] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, BMVA Press, 2016, pp. 87.1–87.12. arXiv: 1605.07146.

[137] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464–7475. arXiv: 2207.02696.

[138] Z.-W. Sun, Z.-X. Hua, H.-C. Li, and Y. Li, "A flying bird object detection method for surveillance video," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–14, 2024. DOI: 10.1109/TIM.2024.3435183. arXiv: 2401.03749 [cs.CV].

[139] G. Jocher, J. Qiu, M. Liu, S. Lyu, F. C. Akyon, and M. E. Kalfaoglu, *Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models*, en, Version Number: 1, 2026. DOI: 10.48550/ARXIV.2606.03748.

[140] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition (ICPR)*, Istanbul, Turkey: IEEE, 2010, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[141] T. R. Lenhard, A. Weinmann, K. Franke, and T. Koch, "SynDroneVision: A synthetic dataset for image-based drone detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025. arXiv: 2411.05633 [cs.CV].

[142] L. Yao et al., "UEMM-Air: Make unmanned aerial vehicles perform more multi-modal tasks," *arXiv preprint arXiv:2406.06230*, 2024. arXiv: 2406.06230 [cs.CV].

[143] Y. Dong et al., "Securing the skies: A comprehensive survey on anti-UAV methods, benchmarking, and future directions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2025. arXiv: 2504.11967 [cs.CV].

[144] T. Gebru et al., "Datasheets for datasets," *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021. DOI: 10.1145/3458723. arXiv: 1803.09010 [cs.DB].

[145] R. Joeres, D. B. Blumenthal, and O. V. Kalinina, "Data splitting to avoid information leakage with DataSAIL," *Nature Communications*, vol. 16, p. 3337, 2025. DOI: 10.1038/s41467-025-58606-8.

[146] M. Oquab et al., "Dinov2: Learning robust visual features without supervision," 2024. arXiv: 2304.07193 [cs.CV].

[147] Amazon Prime Air. "Airborne Object Tracking Dataset." Airborne Object Tracking (AOT) Challenge, AICrowd / ICCV 2021 Workshop on Airborne Object Tracking. [Online]. Available: https://registry.opendata.aws/airborne-object-tracking/.

[148] N. Jiang et al., "Anti-UAV: A large-scale benchmark for vision-based UAV tracking," *IEEE Transactions on Multimedia*, vol. 25, pp. 486–500, 2023. DOI: 10.1109/TMM.2021.3128047. arXiv: 2101.08466 [cs.CV].

[149] Z. Ye et al., "An efficient adjacent frame fusion mechanism for airborne visual object detection," *Drones*, vol. 8, no. 4, p. 144, 2024, Source paper for the ARD-100 / ARD100 amateur drone detection dataset. DOI: 10.3390/drones8040144.

[150] J. Li, D. H. Ye, T. Chung, M. Kolsch, J. Wachs, and C. Bouman, "Multi-target detection and tracking from a single camera in unmanned aerial vehicles (UAVs)," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4992–4997. DOI: 10.1109/IROS.2016.7759733.

[151] Y. Chen, P. Aggarwal, J. Choi, and C.-C. J. Kuo, "A deep learning approach to drone monitoring," *arXiv preprint arXiv:1712.00863*, 2017, Source paper for the USC-Drones dataset; also presented at APSIPA ASC 2017. arXiv: 1712.00863 [cs.CV].

[152] G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng, and J. Han, "Towards large-scale small object detection: Survey and benchmarks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. DOI: 10.1109/TPAMI.2023.3290594. arXiv: 2207.14096 [cs.CV].

[153] H. Caesar et al., "nuScenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 621–11 631. DOI: 10.1109/CVPR42600.2020.01164. arXiv: 1903.11027 [cs.LG].

[154] A. Mumuni, F. Mumuni, and N. K. Gerrar, "A survey of synthetic data augmentation methods in computer vision," *Machine Intelligence Research*, 2024. DOI: 10.1007/s11633-022-1411-7. arXiv: 2403.10075 [cs.CV].

[155] M. Li et al., *Multi-View Large Reconstruction Model via Geometry-Aware Positional Encoding and Attention*, arXiv:2406.07648 [cs], 2024. DOI: 10.48550/arXiv.2406.07648. arXiv: 2406.07648 [cs.CV].

[156] A. Lukežic, T. Vojír, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative Correlation Filter with Channel and Spatial Reliability," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jul. 2017, pp. 4847–4856. DOI: 10.1109/CVPR.2017.515.

[157] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*, arXiv:1312.6034 [cs.CV], Apr. 2014. DOI: 10.48550/arXiv.1312.6034.

[158] V. Petsiuk, A. Das, and K. Saenko, "RISE: randomized input sampling for explanation of black-box models," *CoRR*, vol. abs/1806.07421, 2018. arXiv: 1806.07421.

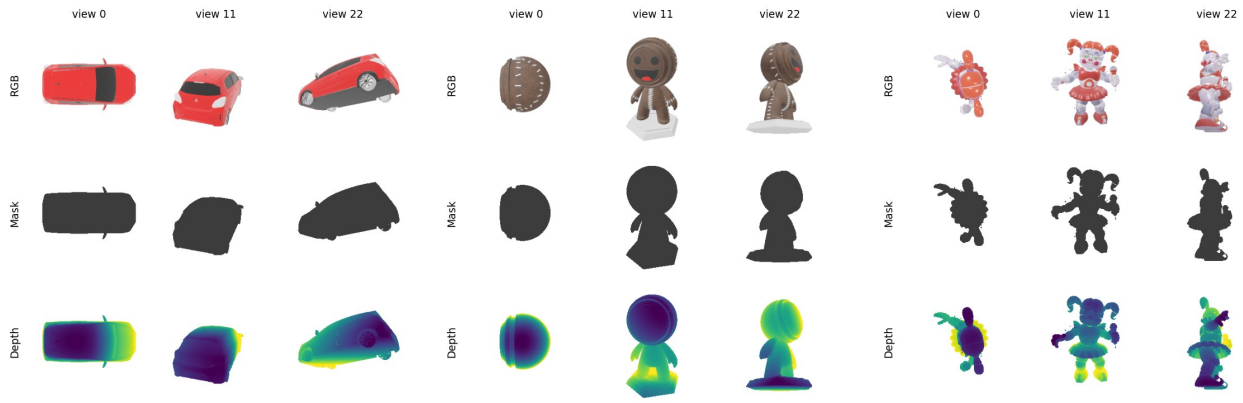
[159] A. X. Chang et al., *ShapeNet: An information-rich 3D model repository*, Technical Report, 2015. arXiv: 1512.03012 [cs.GR].

[160] S. Samaras et al., "Deep Learning on Multi Sensor Data for Counter UAV Applications—A Systematic Review," *Sensors*, vol. 19, no. 22, p. 4837, 2019. DOI: 10.3390/s19224837.

A Datasets

This appendix collects per-dataset render galleries referenced from Chapter 5: §A.1 shows the Stage-2 reconstruction corpora (§5.5.2) and §A.2 shows the Stage-3 classification corpora (§5.6.9). AeroSplat-4D is visualized separately in Appendix B (Asset Library).

A.1 Stage-2 object renders

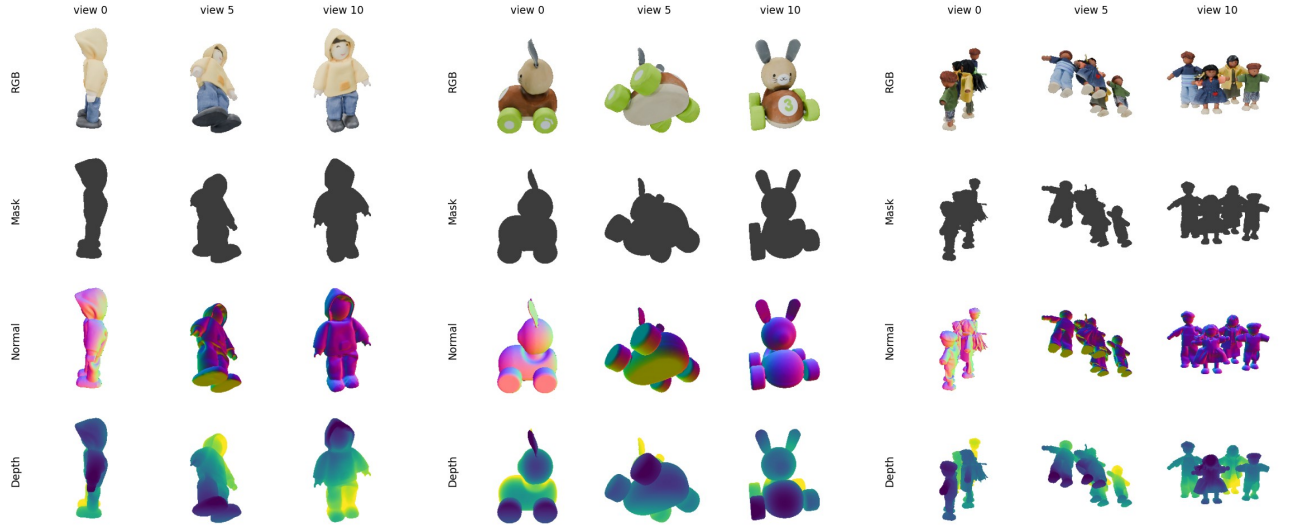


(a) `35965c7e...` (b) `d033ad66...` (c) `a63b628a...`

Figure A.1: Objaverse pre-training corpus (curated Training Set B): three sampled UIDs at three viewpoints, showing the RGB render, binary foreground mask, and per-image normalized EXR depth (viridis colormap, white background). Surface normals are not retained in the processed `.torch` chunks used at training time.

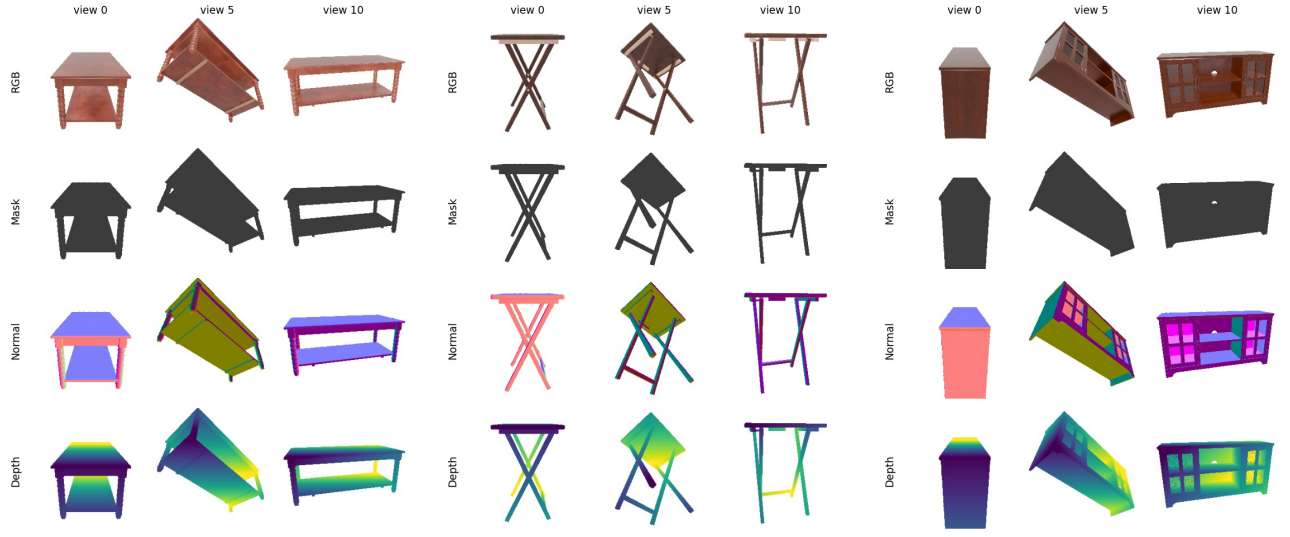
Sketchfab provenance (all CC-BY): `35965c7eb6804ce9884e39ae353b68f2` "2020 Mitsubishi Mirage Facelift" by Ray2007ben; `d033ad663758457c8e3a5d5d30d1459d` "Sackboy Meetmat" by csierra; `a63b628a2d084a6baabbe489940b1018` "Baby" by funtimfoxy9g.

A.2 Stage-3 object renders



(a) `3D_Dollhouse_Happy_Brother` (b) `BUNNY_RACER` (c) `DOLL_FAMILY`

Figure A.2: Google Scanned Objects (GSO) evaluation rig: three example assets at three viewpoints, showing RGB, mask, view-space normals (RGB-encoded), and depth (viridis). All assets are rendered with the Fibonacci-target + cardinal-context protocol shared with ABO (§5.5.2).



(a) `B07MHMTC6T` (b) `B07MDJM5QD` (c) `B07DBFC2N9`

Figure A.3: Amazon Berkeley Objects (ABO) evaluation rig: three example ASINs at three viewpoints, showing RGB, mask, normals, and depth. The render protocol matches GSO exactly; the visual gap is purely the product-imagery vs. photogrammetric domain.

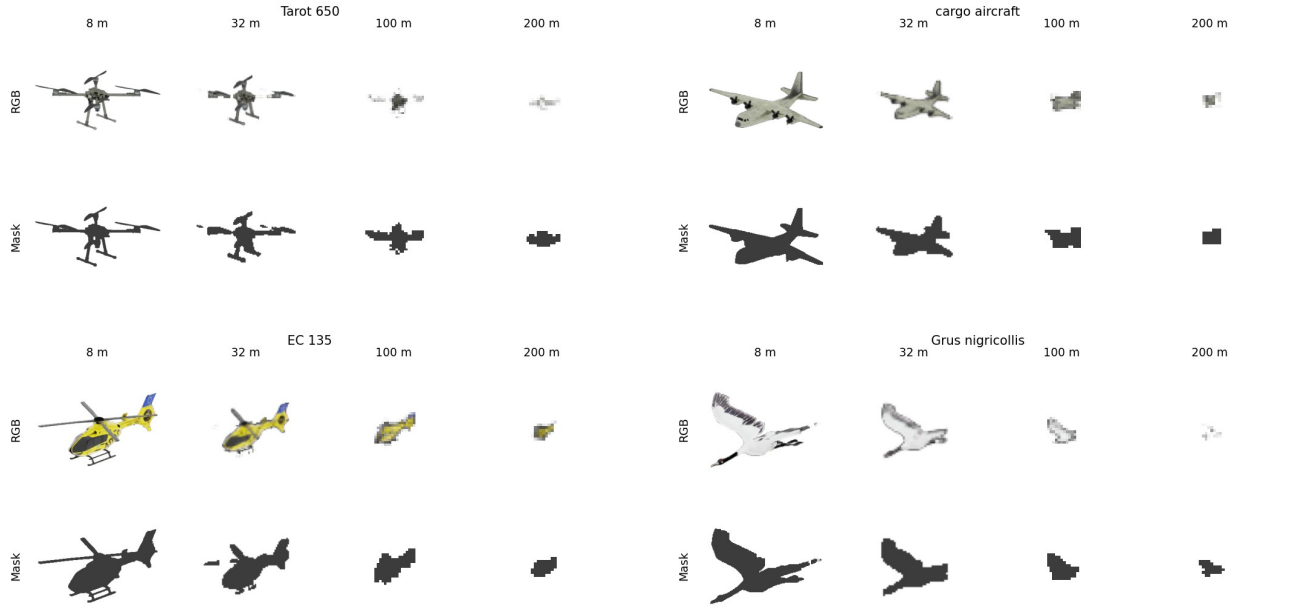
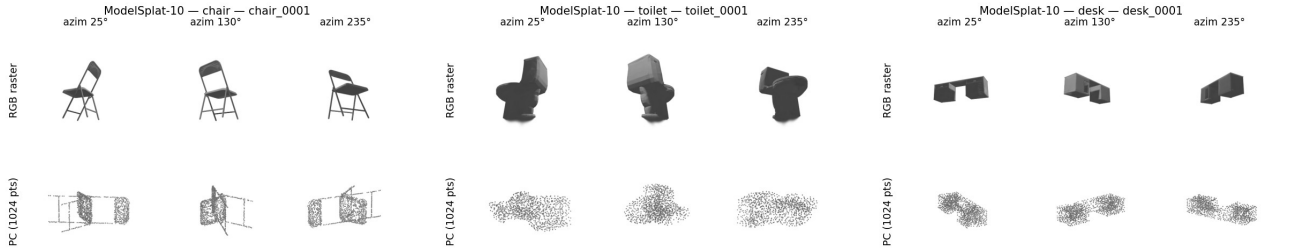
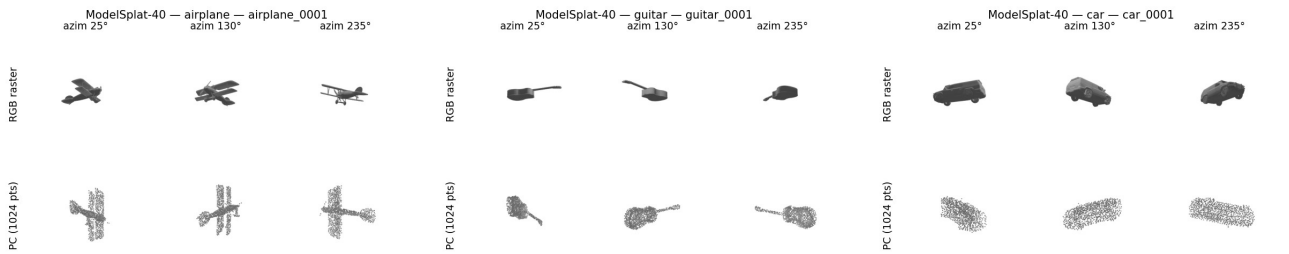


Figure A.4: AeroSplat-4D Sim-1 distance sweep: one asset per category (drone, airplane, helicopter, bird) shown at four radii (8, 32, 100, 200 m) from the camera, cropped to the foreground bounding box and rescaled. RGB and binary mask are the only modalities rendered for Sim-1. The visible degradation arc — "object fills frame" at 8 m to "few-pixel blob" at 200 m — is the regime the distance-degradation study (§6.4.3) probes.



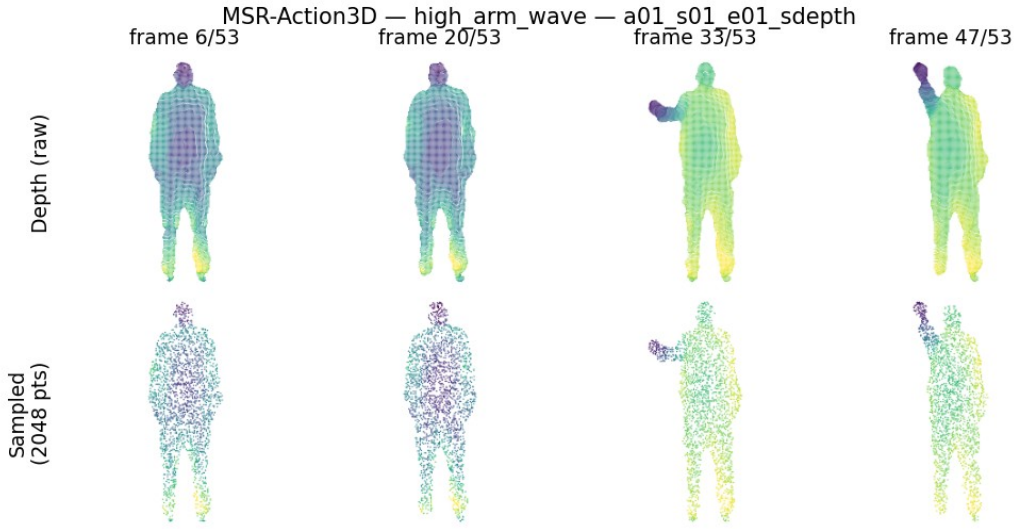
(a) `chair` (b) `toilet` (c) `desk`

Figure A.5: ModelSplat-10 — three example classes, each shown as a rasterized RGB render of the original 3DGS PLY at three viewpoints (top row) and a 3D scatter of the matching preprocessed 1024-point tensor (bottom row). Render protocol described in §5.6.9; preprocessing detailed in Table 5.4.

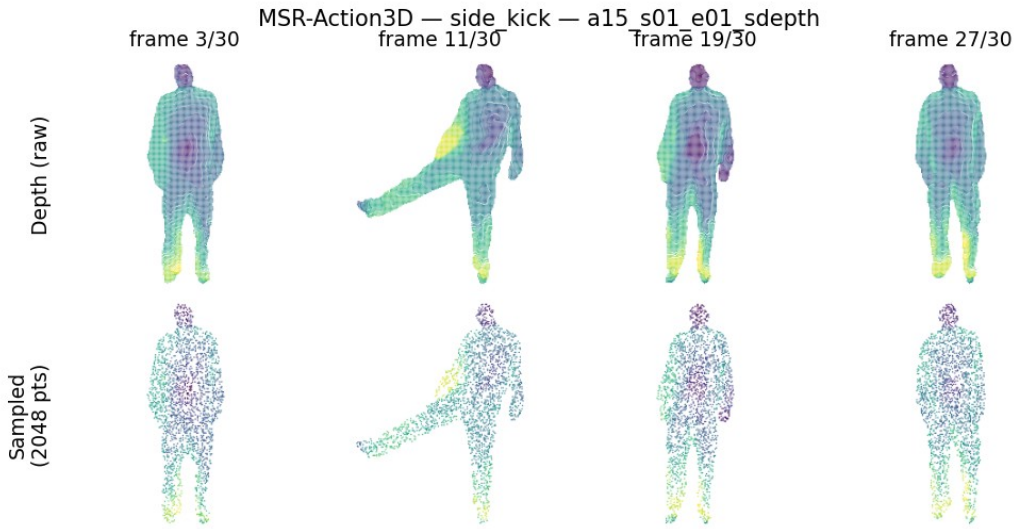


(a) `airplane` (b) `guitar` (c) `car`

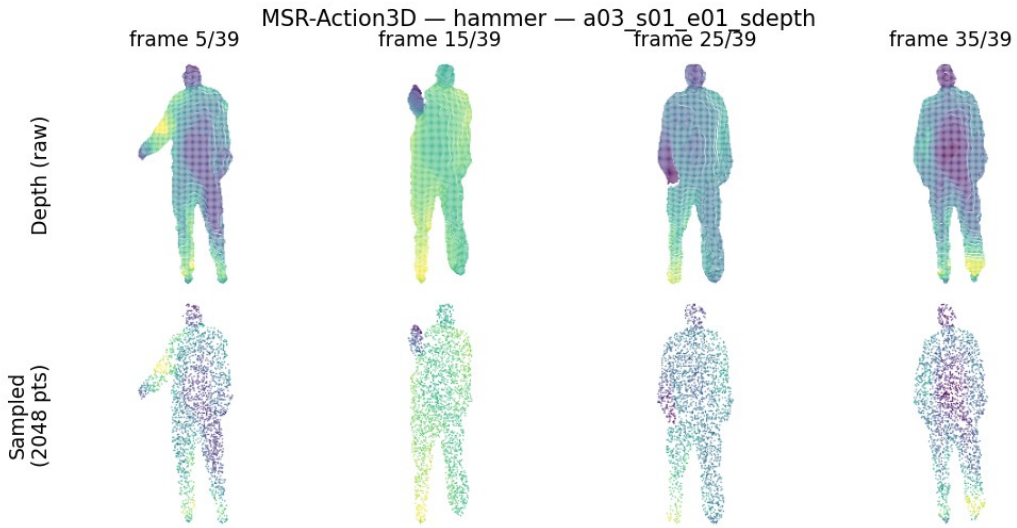
Figure A.6: ModelSplat-40 — three example classes (visually distinct geometries), same layout as ModelSplat-10 (rasterized RGB top, 1024-point preprocessed tensor bottom; three viewpoints). The wider class catalog is the only protocol difference from ModelSplat-10.



(a) `high_arm_wave`



(b) `side_kick`



(c) `hammer`

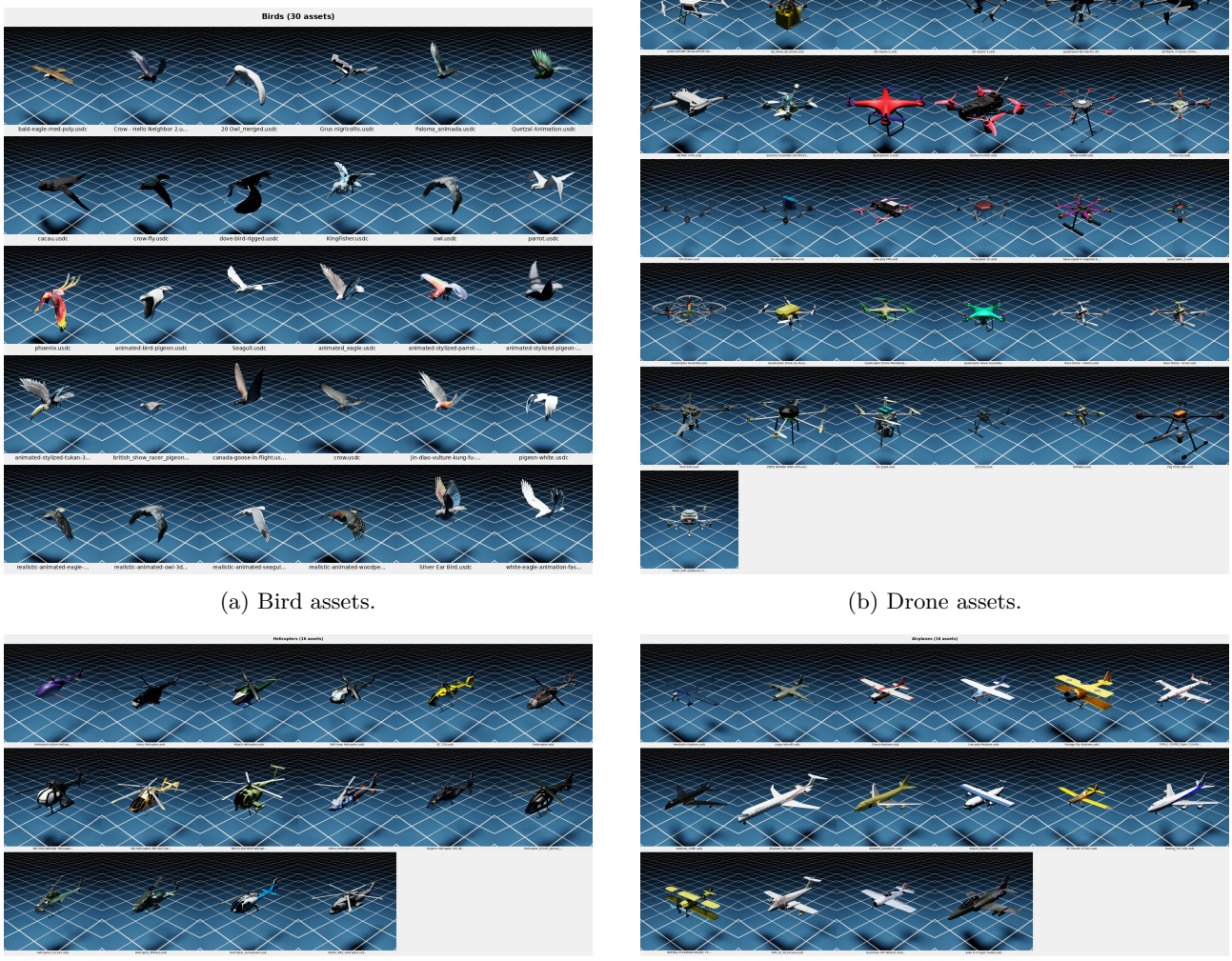
Figure A.7: MSR-Action3D — three example action classes, each shown as four sampled frames per clip with the raw depth-camera point cloud (top row) and the 2048-point loader-side sampling that the trainer consumes (bottom row, matching Mamba4D's `_sample_fixed_points` logic). See §5.6.9.

B

Dataset Characterisation

This appendix collects the per-class asset library, the temporal cycle structure at all camera distances, and the per-camera / per-Gaussian distributions referenced from Chapter 5 (§5.3.3) and Chapter 6 (§6.3.2, §6.3.5).

B.1 Asset Library



(a) Bird assets.

(b) Drone assets.

(c) Helicopter assets.

(d) Airplane assets.

Figure B.1: Asset library across the four flyer classes used in AeroSplat-4D.

B.2 Geometric Coverage

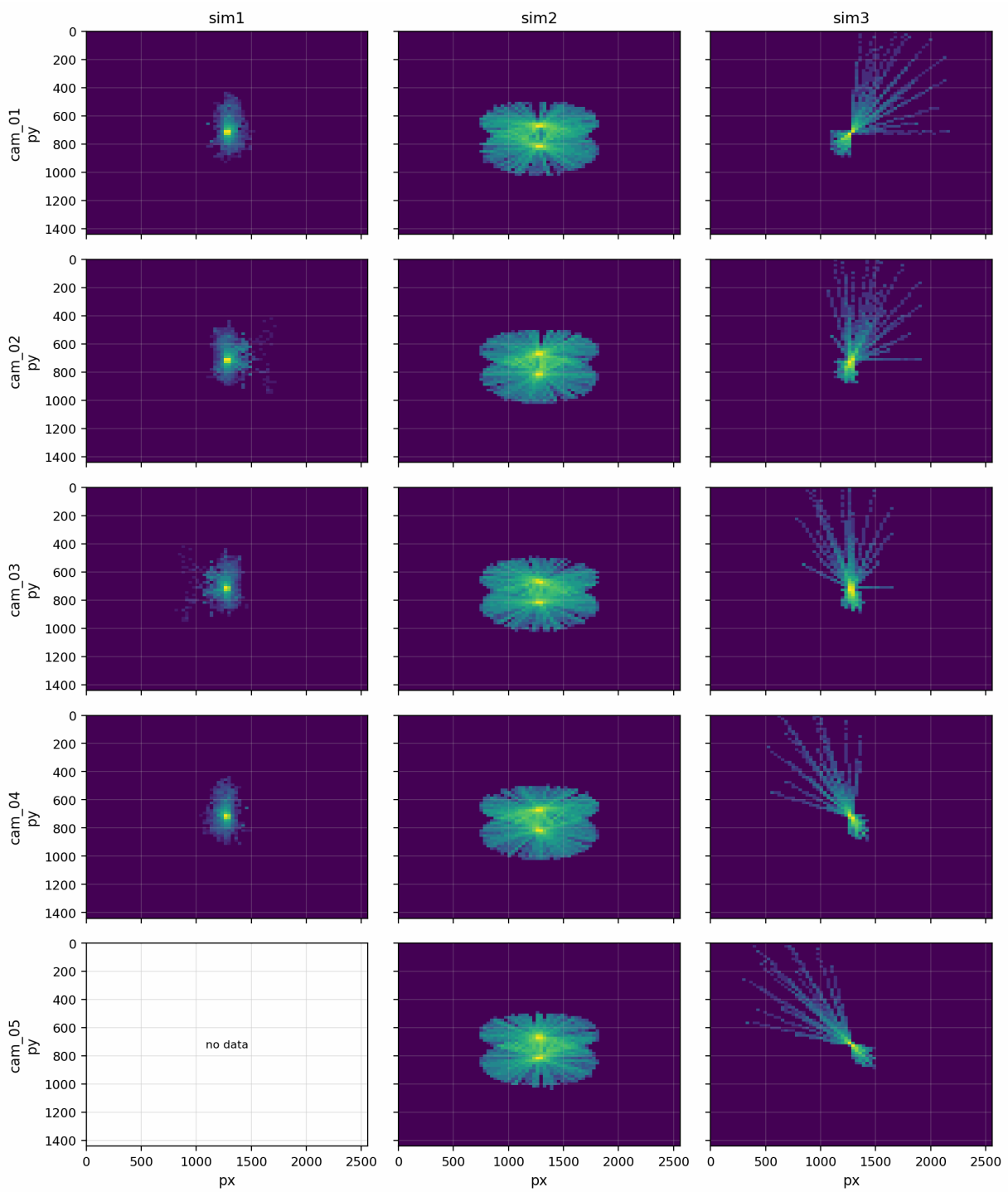


Figure B.2: Per-camera log-density centroid heatmap. Rows: cam_01–cam_05; columns: Sim-1, Sim-2, Sim-3. See Fig. 6.1 for the cam_03 slice used in the main text.

B.3 Temporal Cycle Structure

Figures B.3–B.10 show the per-category mean Chamfer distance-to-frame-1 curves and per-asset FFT overlays for nine camera distances not shown in the main text (Figure 6.7 covers 4r). The temporal signal progressively attenuates with distance; beyond roughly 100r all categories converge to near-constant Chamfer curves.

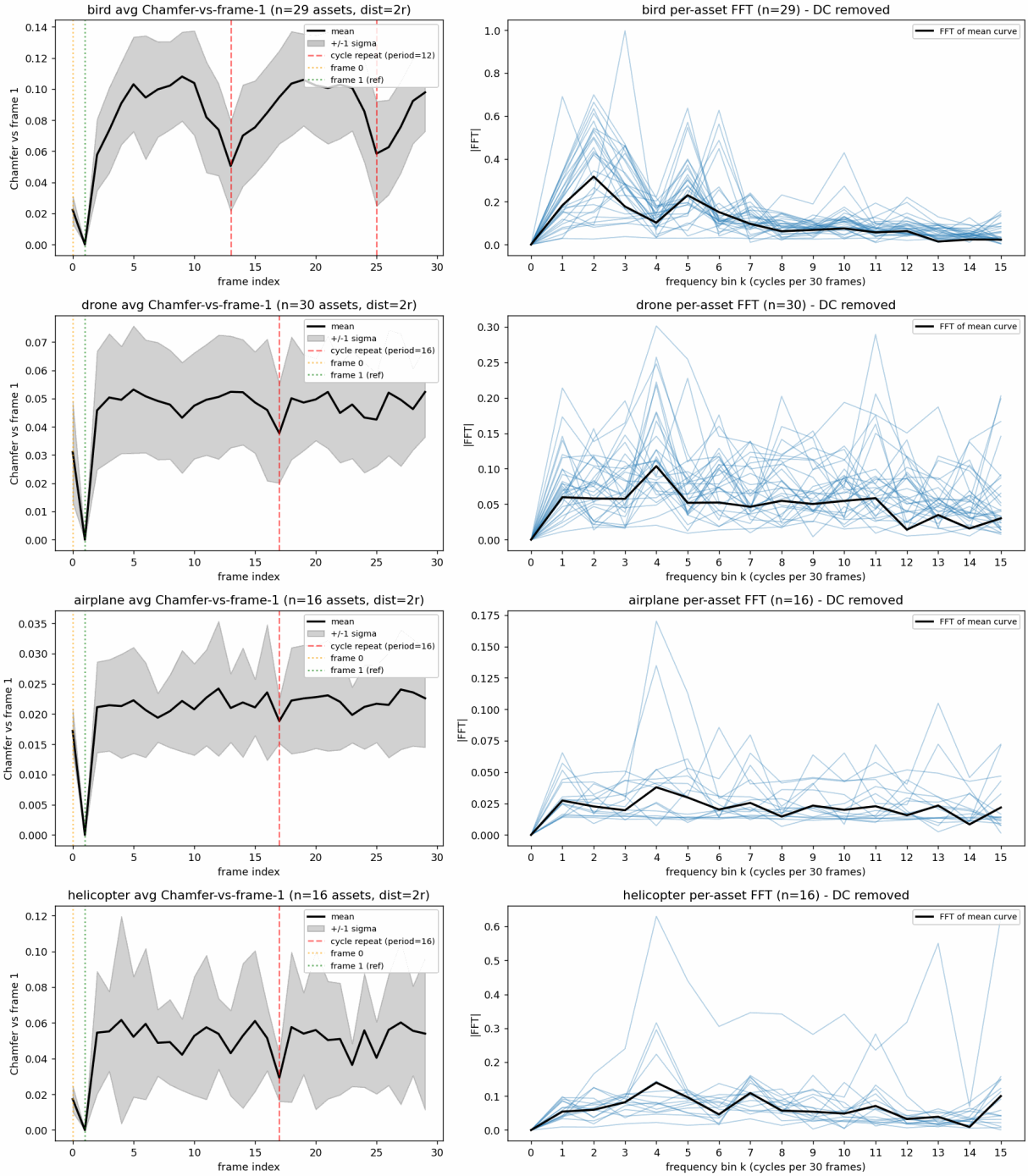


Figure B.3: Per-category Chamfer cycle and FFT at distance 2r. Layout as Figure 6.7.

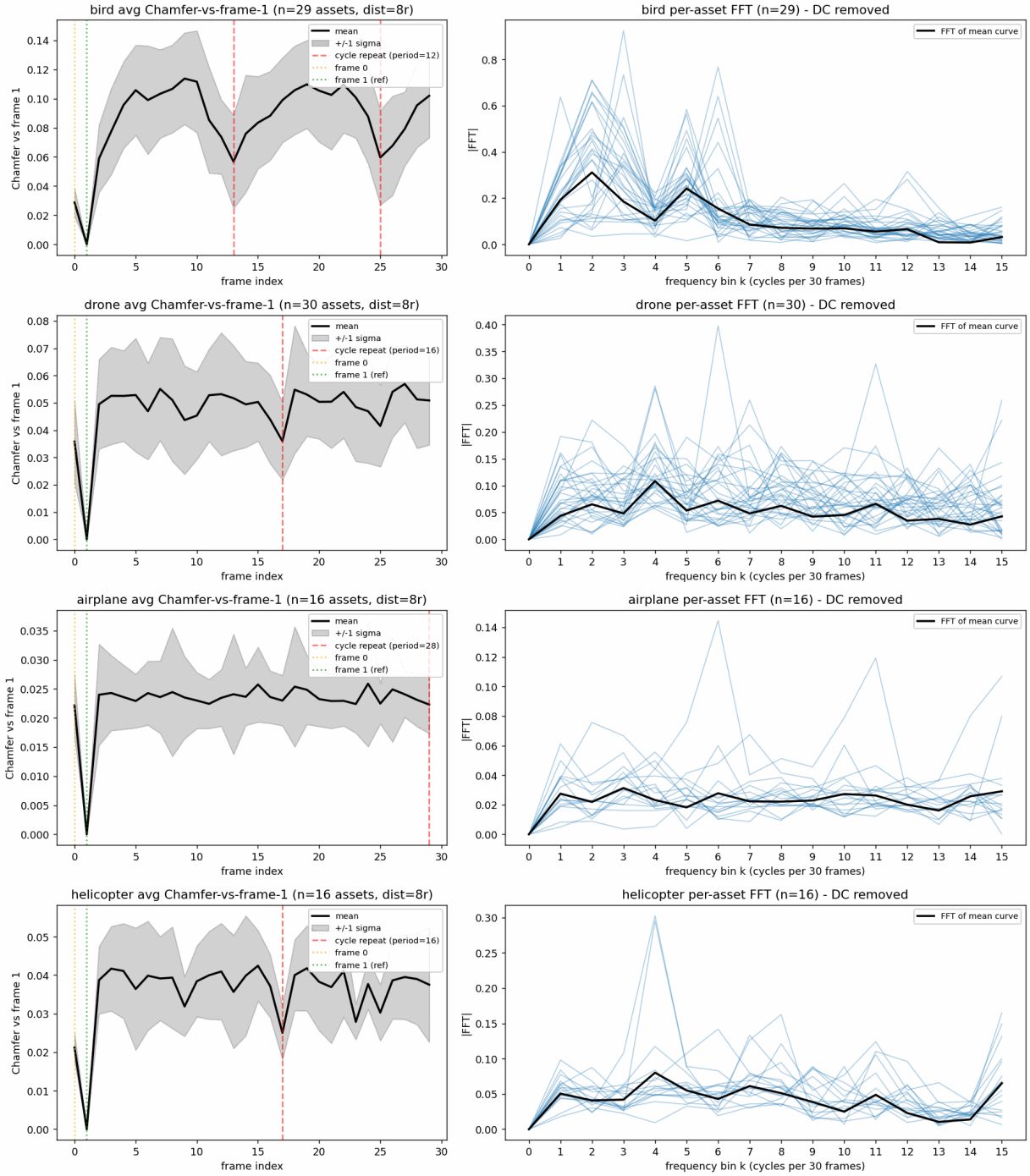


Figure B.4: Per-category Chamfer cycle and FFT at distance 8r. Layout as Figure 6.7.

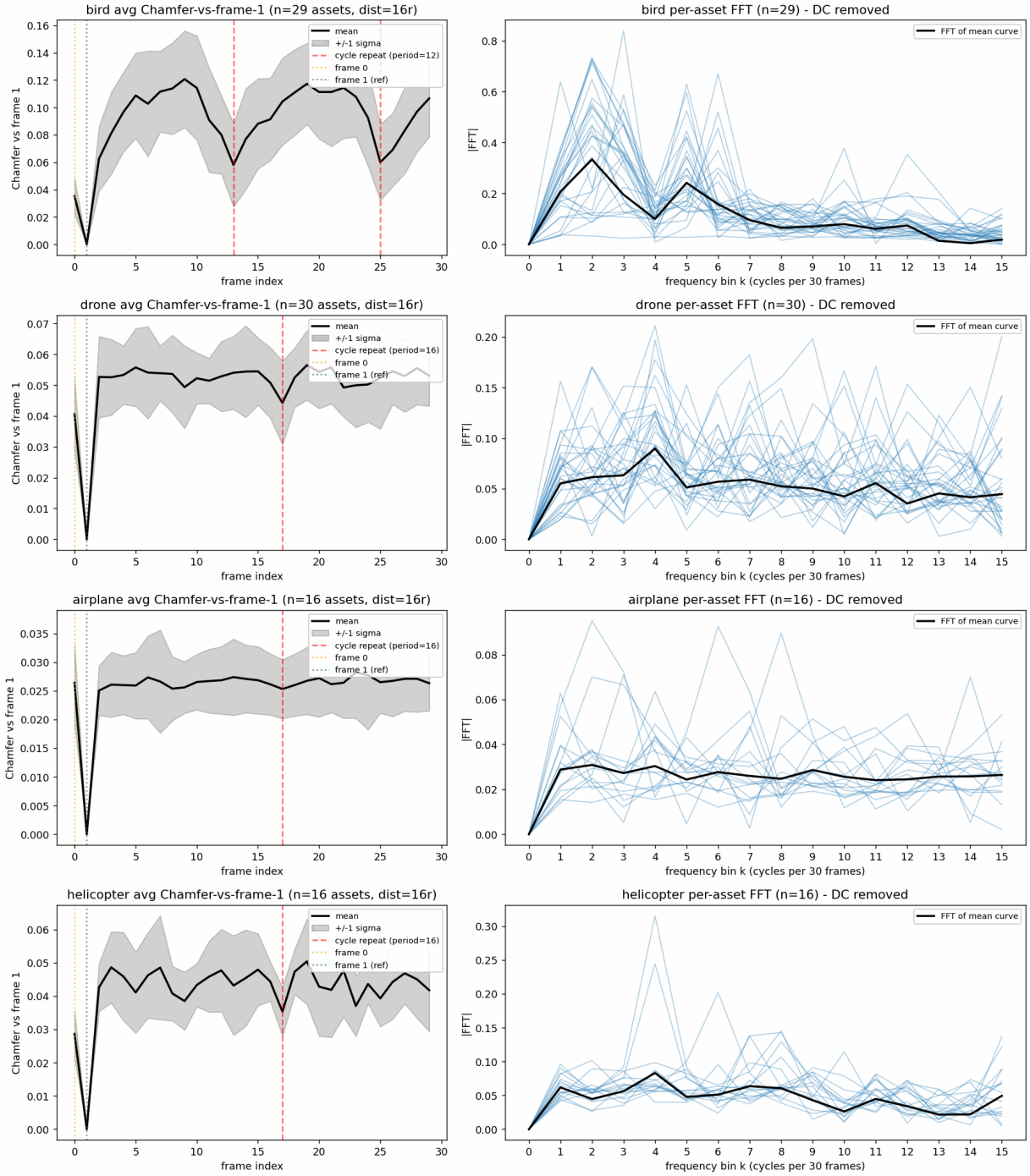


Figure B.5: Per-category Chamfer cycle and FFT at distance 16r. Layout as Figure 6.7.

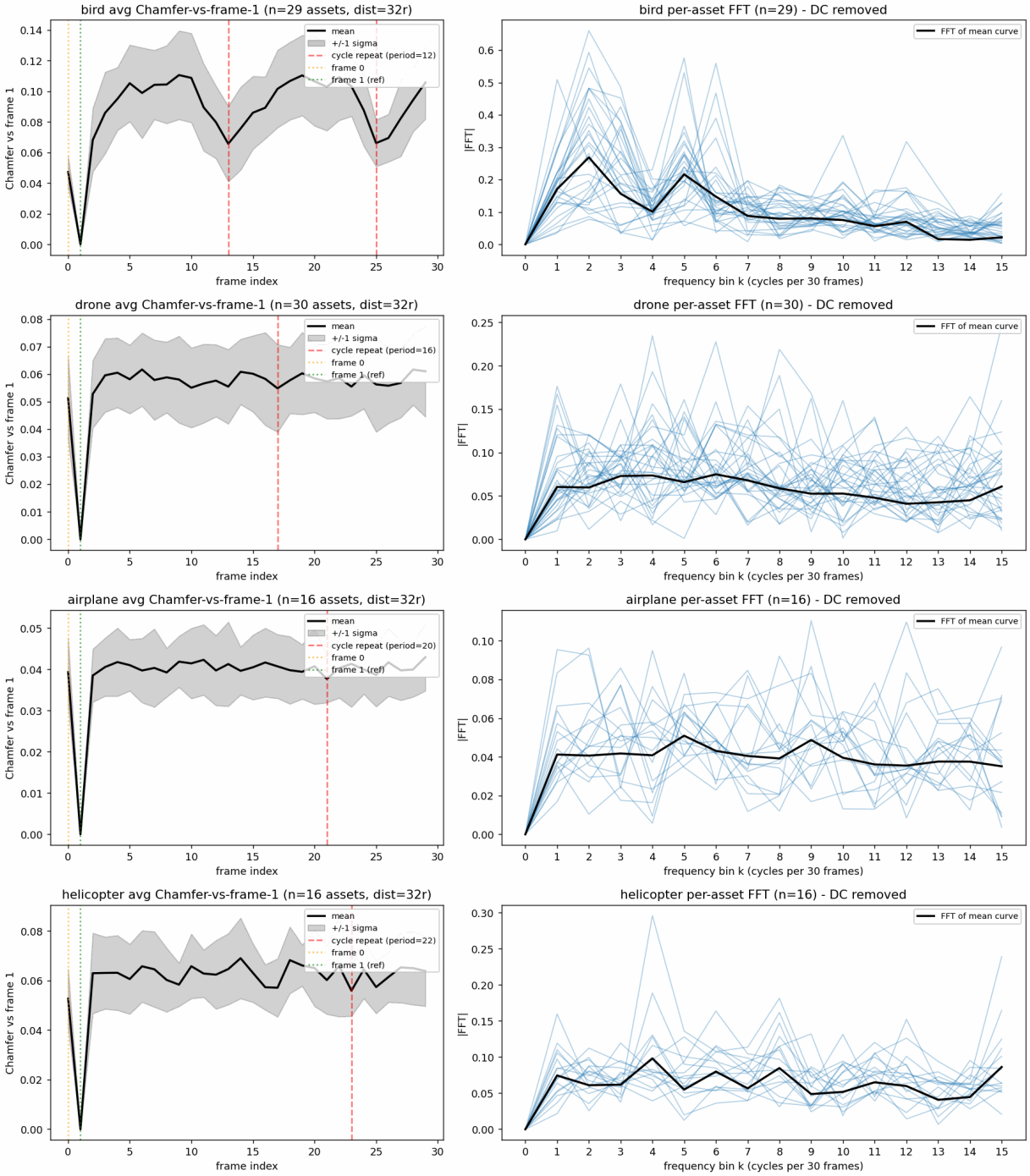


Figure B.6: Per-category Chamfer cycle and FFT at distance 32r. Layout as Figure 6.7.

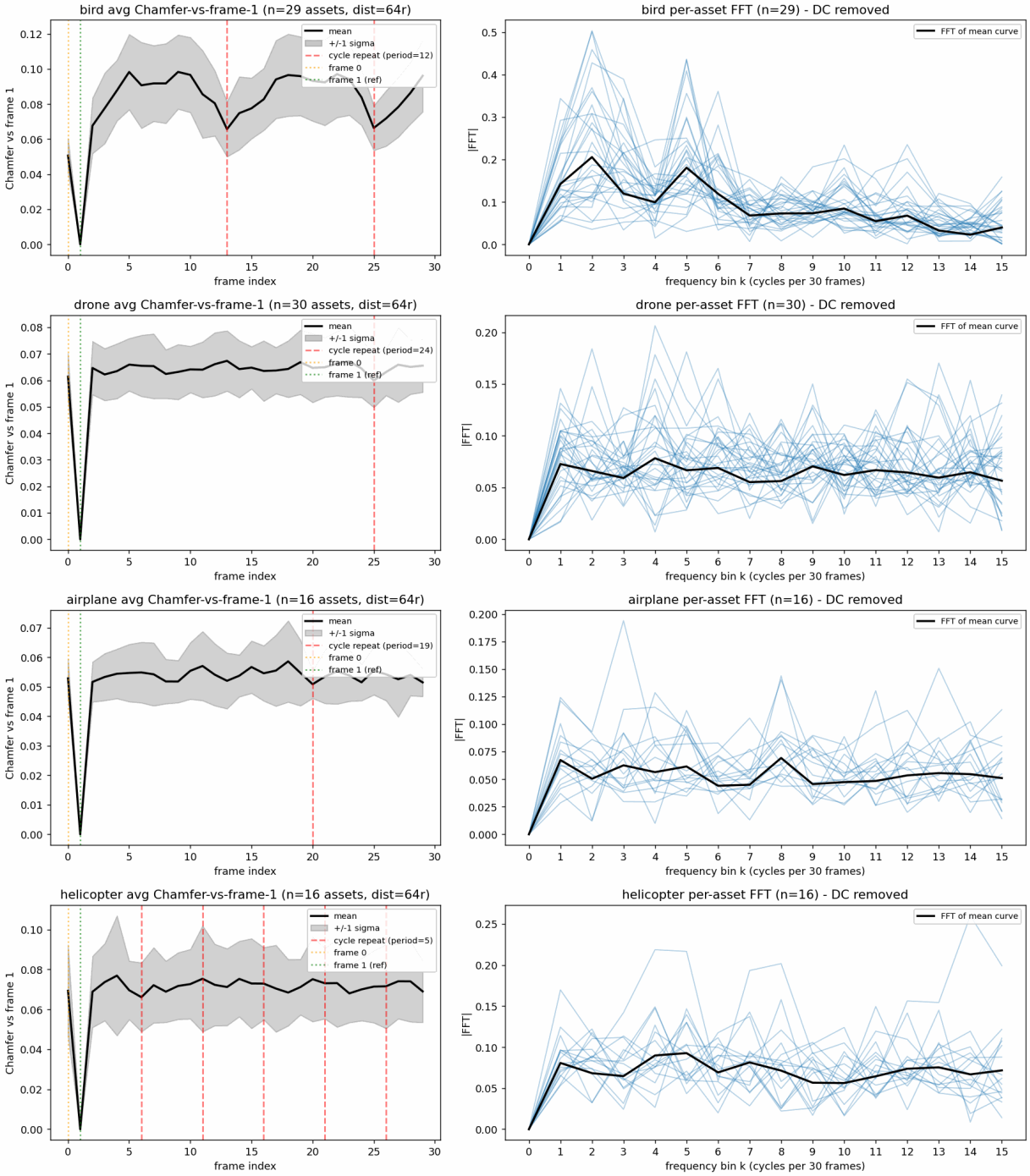


Figure B.7: Per-category Chamfer cycle and FFT at distance 64r. Layout as Figure 6.7.

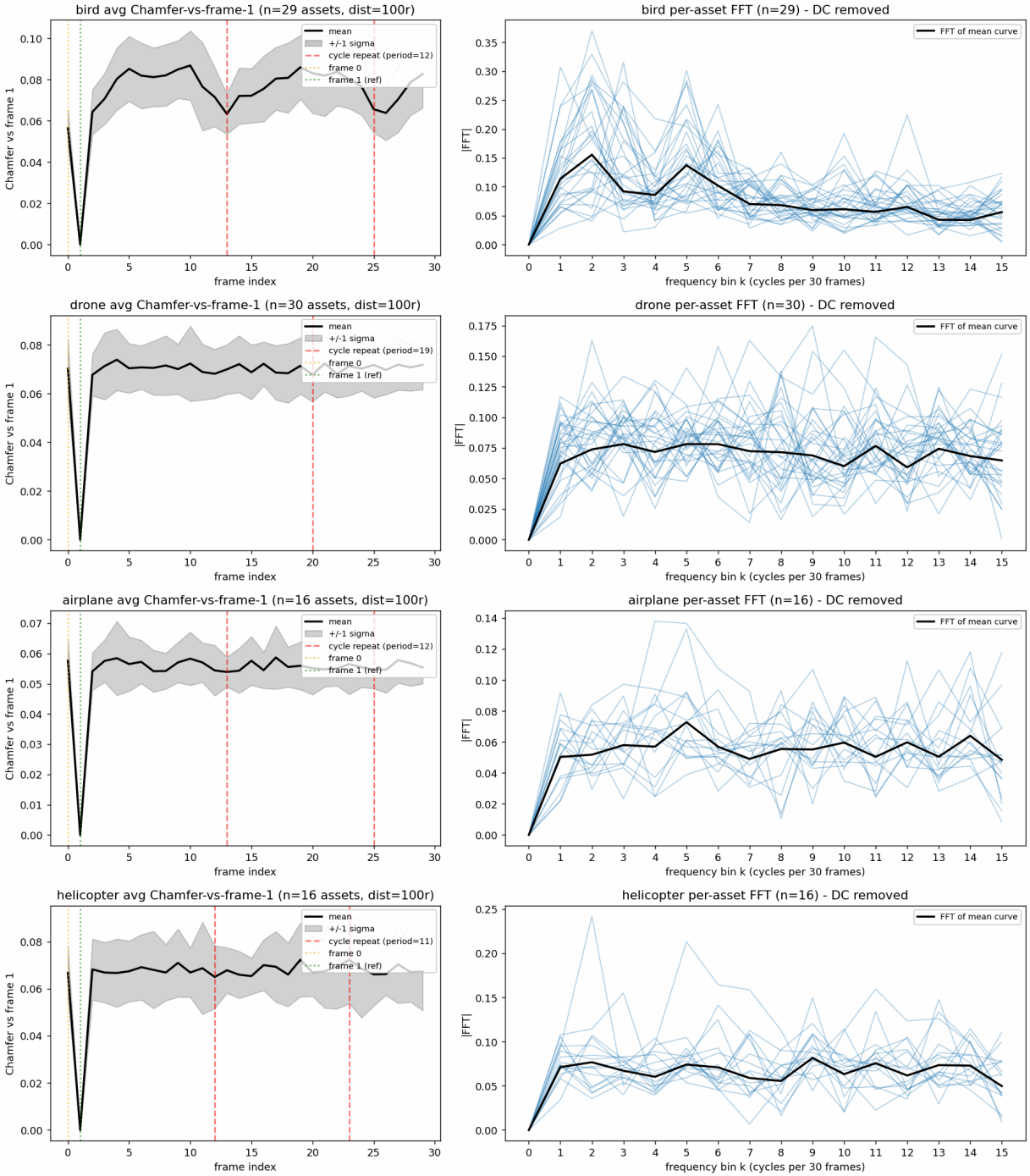


Figure B.8: Per-category Chamfer cycle and FFT at distance 100r. Layout as Figure 6.7.

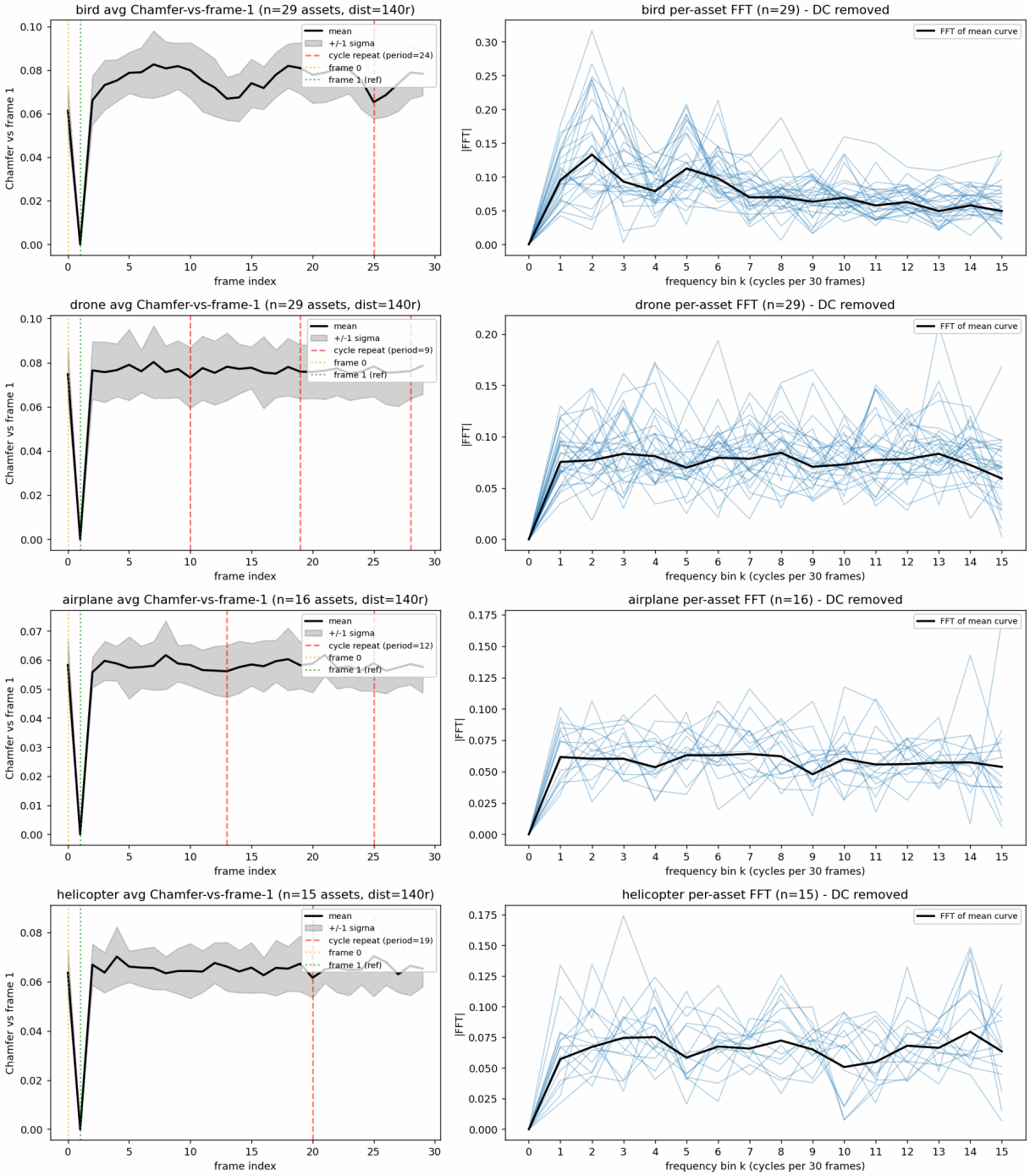


Figure B.9: Per-category Chamfer cycle and FFT at distance 140r. Layout as Figure 6.7.

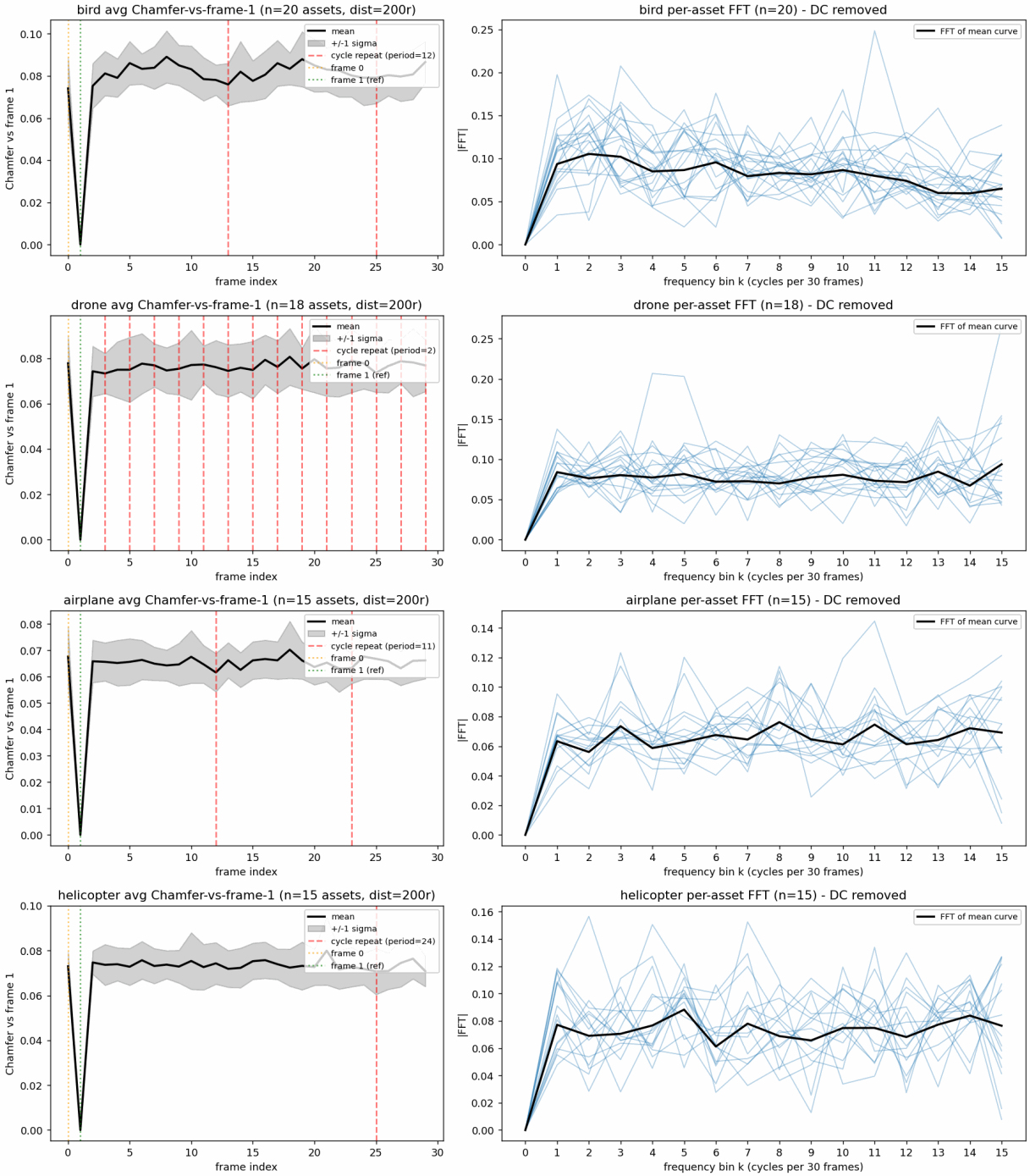


Figure B.10: Per-category Chamfer cycle and FFT at distance 200r. Layout as Figure 6.7.

B.4 Gaussian-Attribute Distributions

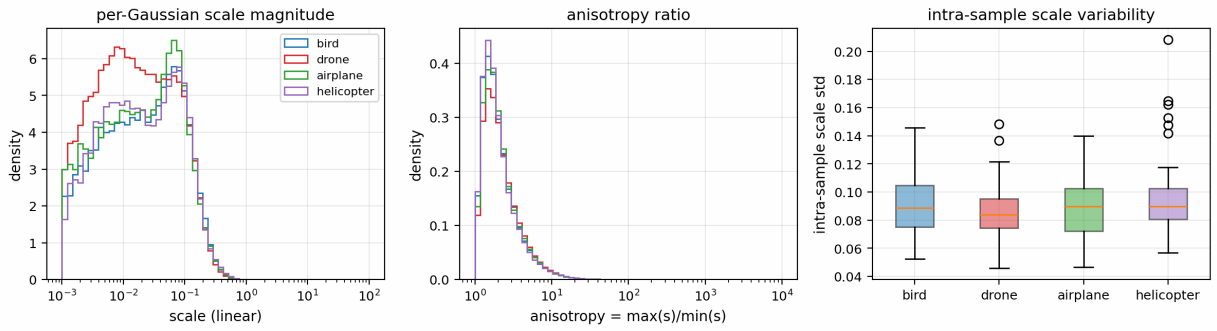


Figure B.11: Per-class scale: magnitude, anisotropy, and box-plot summary.

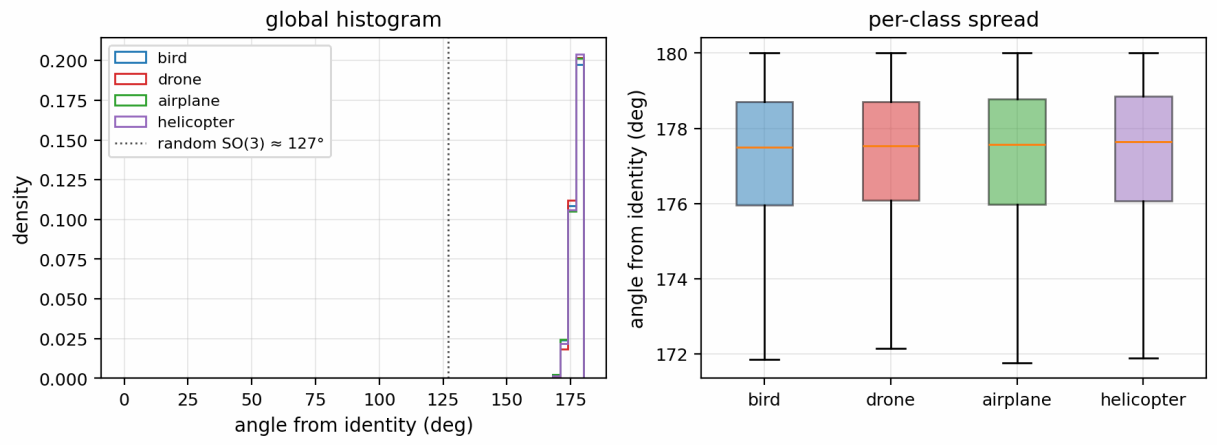


Figure B.12: Per-class quaternion-angle distribution.

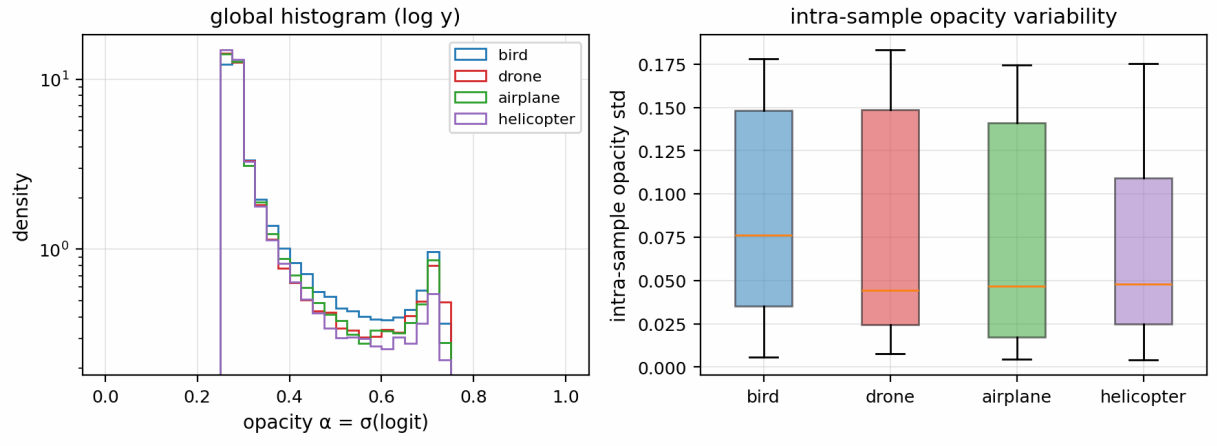


Figure B.13: Per-class opacity variation.

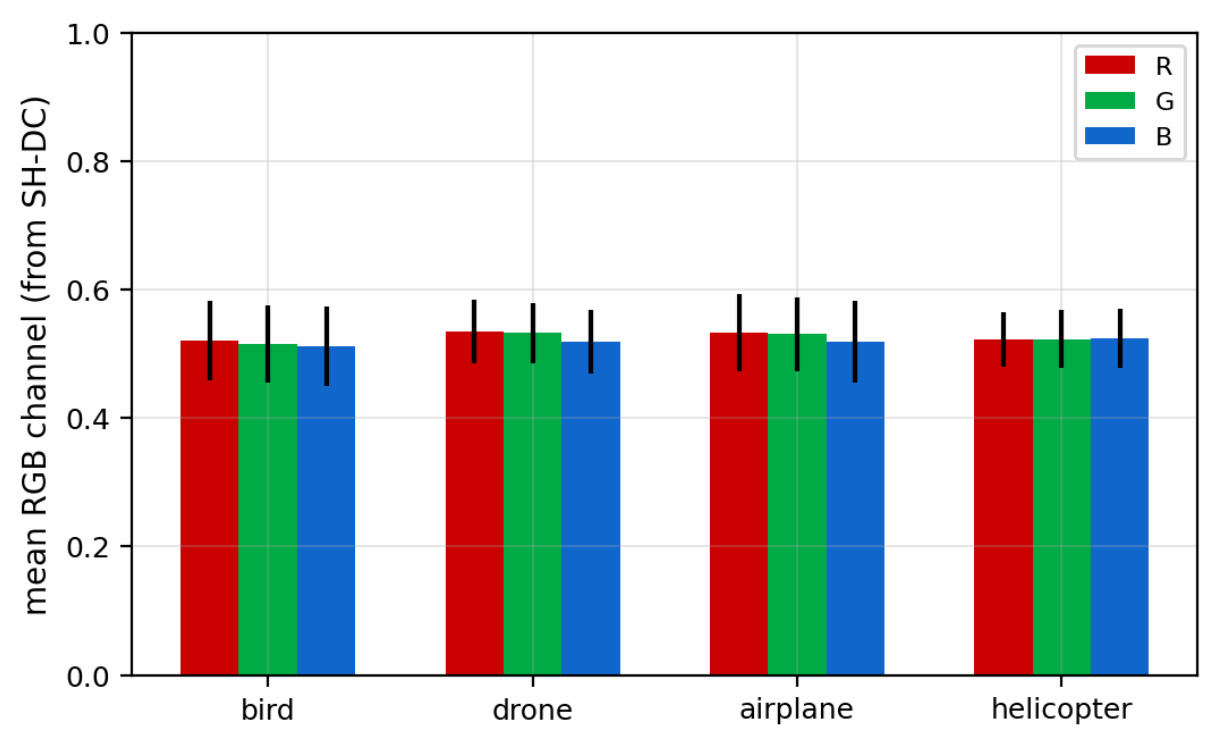


Figure B.14: Per-class mean SH-DC color.

C

Future Work — RGB-only Multi-View Foreground Segmentation

This appendix preserves the design of an RGB-only multi-view foreground segmentation module that was prototyped during the thesis but did not produce final results within the available time budget. The downstream pipeline used Isaac Sim ground-truth masks in its place (§5.4). The design is documented here as the proposed approach for future work (§8.2).

C.1 Problem Formulation

The proposed module isolates small, fast-moving flying objects against a textureless sky and variable clutter — a regime where single-frame appearance is unreliable. The task is framed as a multi-view volumetric accumulation problem: motion cues from each camera unproject into a shared 3D volume, where geometric consensus across cameras yields foreground.

- Segment small flying objects against sky and moving clutter.
- No appearance prior; exploit motion and multi-view geometry jointly.
- Handle textureless sky, dynamic lighting, and shadows.
- Output: per-camera refined 2D foreground masks compatible with the Stage 2 interface.

C.2 Motion-Cue Front-End

Each camera produces a changed-pixel likelihood map from a short temporal window.

- Temporal differencing against neighboring or background frames.
- Gaussian blur, threshold, and morphology suppress single-pixel noise.
- Per camera: changed-pixel set with intensity weights.

C.3 Ray-March Voxel Voting

A shared voxel grid accumulates evidence from rays cast through changed pixels of every camera. Voxels receiving sufficient multi-view consensus are declared foreground.

- Unproject changed pixels to world-space rays per camera.
- Shared voxel grid accumulates per-ray intensity contributions.
- Consensus filter: voxel must receive weight from $\geq k$ cameras.
- Implicit correspondence: no per-pixel matching required.
- Localization via argmax or center-of-mass of the grid.

C.4 Vote Back-Projection to 2D Masks

Foreground voxels are re-projected into each camera to produce refined per-view masks compatible with the Stage 2 mask-aware encoder.

- Cast rays from foreground voxels back into each camera image.
- Pixels above a vote threshold become foreground.
- Delivers per-camera 2D masks to Stage 2's mask-aware encoder.

C.5 Open Questions

- Handling moving clutter: clouds, leaves, cast shadows.
- Consensus-threshold k sensitivity vs. camera count N.
- Real-time feasibility on Jetson AGX Orin under $N \geq 3$ cameras (edge-computing goal retained; no current hardware exists to validate Jetson real-time feasibility under real-world conditions).
- Quantitative comparison against simulator ground-truth masks on Sim-2 (train+val+test) and Sim-3 (test-identity OOD renders).

D

Stage-3 Pipeline Overview

This appendix consolidates the per-dataset configuration of the Stage-3 pipeline (preprocessing, training, augmentation, baselines, and evaluation) in a single reference table. The supervised trainer, cross-entropy objective, protocol-driven rotation augmentation, and rotation-evaluation protocols form a shared core across datasets; Table D.1 states only the per-dataset deviations and is referenced from the Stage-3 Preprocessing (§5.6.10) and Training Pipeline (§5.6.11) sections.

Table D.1: Complete Stage-3 pipeline overview; each column states only the per-dataset deviations from the shared core. "–" denotes not applicable.

	ModelSplat (static, 3DGS)	**MSR-Action3D** (temporal, depth PC)	**AeroSplat-4D** (temporal, 3DGS)
Preprocessing			
Source	ShapeNet PLY, 3DGS-lifted	Raw `.bin` depth maps	DepthSplat-OC 3DGS reconstructions
Decimation	FPS (8192→1200→1024)	Random subsample at load	FPS (8192→1200→1024)
Points/frame N	1024	2048	1024
Stored FPS versions	25 per object	— (resampled per epoch)	1 per frame
Frames/sample T	1	24	24
Centering	Per-sample centroid	Per-clip global centroid	Per-frame centroid
Split	ModelNet official train/test	Subjects 1–5 / 6–10	Identity-disjoint ∼50/25/25
Training			
Experiment(s)	Attribute ablation	Rotation protocol	Four-class & ablations
Optimiser	SGD	AdamW	AdamW
Learning rate	0.1	10^{-2}	10^{-3}
Schedule	step ×0.7/20 ep	multistep ×0.1 @{20, 30}, 10-ep warmup	cosine, 3-ep warmup
Epochs	150	50	50
Batch	64	12	8 × 2 (eff. 16)
Gradient clip	$\|g\|_2$=1.0	off	$\|g\|_2$=1.0
Loss	Cross-entropy	Cross-entropy	Cross-entropy
Label smoothing	0.0	0.0	0.1
Recipe origin	Repository default	Baseline-matched	Repository default
Augmentation			
Scale (per-axis)	[2/3, 3/2], p=1.0	[0.9, 1.1], p=1.0	[0.9, 1.1], p=1.0
Translation	±0.2 per-axis, p=1.0	off	off
Rotation	Protocol-driven, joint (p, q)	Protocol-driven, points only	Protocol-driven, joint (p, q)
Temporal window	—	Contiguous 24-frame slide	Contiguous 24-frame slide, stride 3
Jitter / dropout	off / off	off / off	off / off
Baselines			
Family	Static point-cloud	Temporal point-cloud	Temporal point-cloud
Methods	VN-PointNet, VN-DGCNN	P4Transformer, Mamba4D	P4Transformer, Mamba4D (§6.5.3)
Evaluation			
Rotation protocols	none/none, z/z, z/SO(3), SO(3)/SO(3)	+ z/SO(3) per-frame	+ z/SO(3) per-frame
Aggregation	Clip-level	Clip- and video-level	Clip- and video-level
Special regime	—	Per-class analysis	Sim-3 OOD probe (24 identities)

E

Classifier Confusion Matrices

This appendix shows the per-class confusion matrices behind the two `MambaSplat-4D` rows ($E(X)$ and $E(C)$) of Table 6.18, across the three evaluation regimes (Sim-1 ID, Sim-2 ID, Sim-3 OOD). Each matrix is row-normalized and pooled over the three training seeds (42, 43, 44); the in-distribution regimes use the held-out test split and the Sim-3 OOD probes its test split, matching the table. Figure E.1 reports the sequence-level prediction used for the tabulated accuracy, and Figure E.2 the finer clip-level prediction.

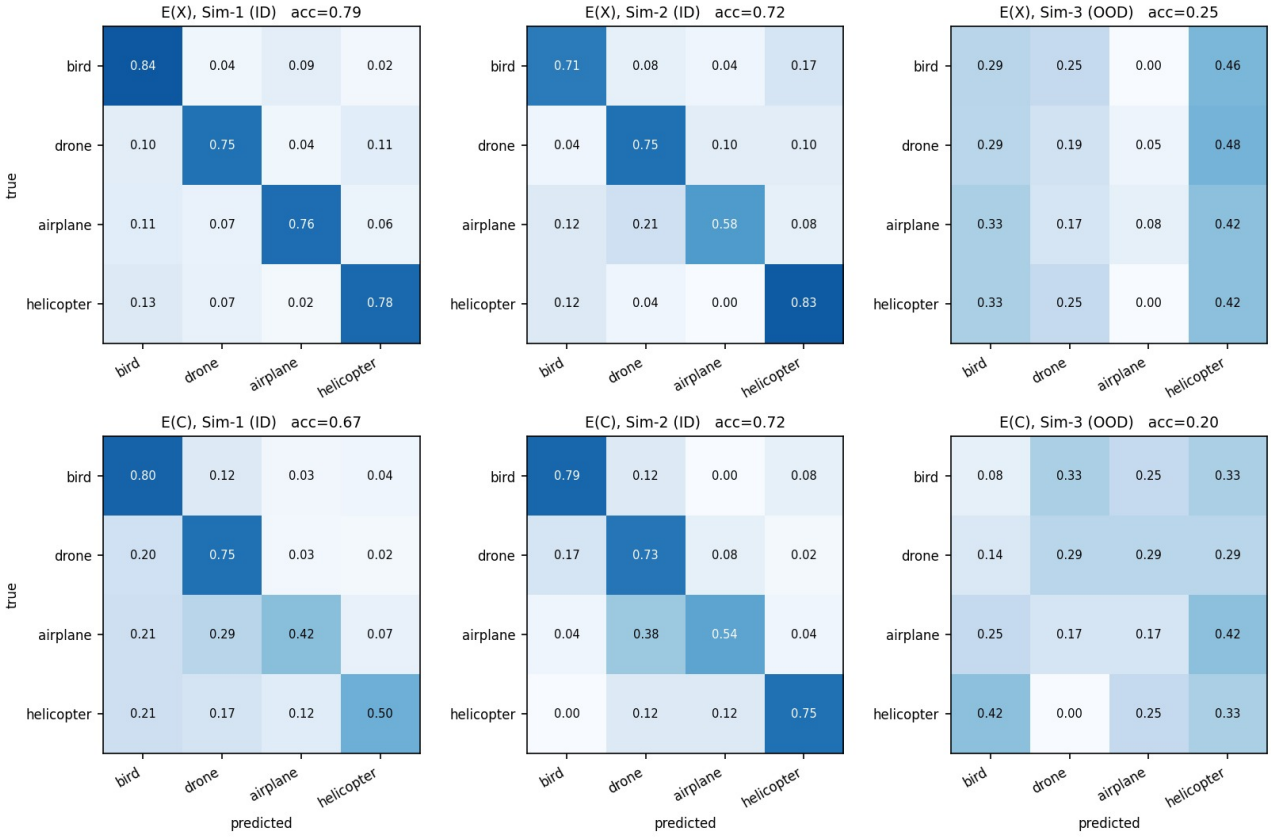


Figure E.1: Sequence(video)-level confusion matrices for the $E(X)$ (top row) and $E(C)$ (bottom row) entries of Table 6.18, row-normalized and pooled over seeds 42/43/44. Columns are Sim-1 (ID, test), Sim-2 (ID, test), and Sim-3 (OOD, test); each panel's accuracy matches the corresponding table cell. Under the Sim-3 OOD shift, both feature modes lose the diagonal structure, consistent with the low out-of-distribution accuracy.

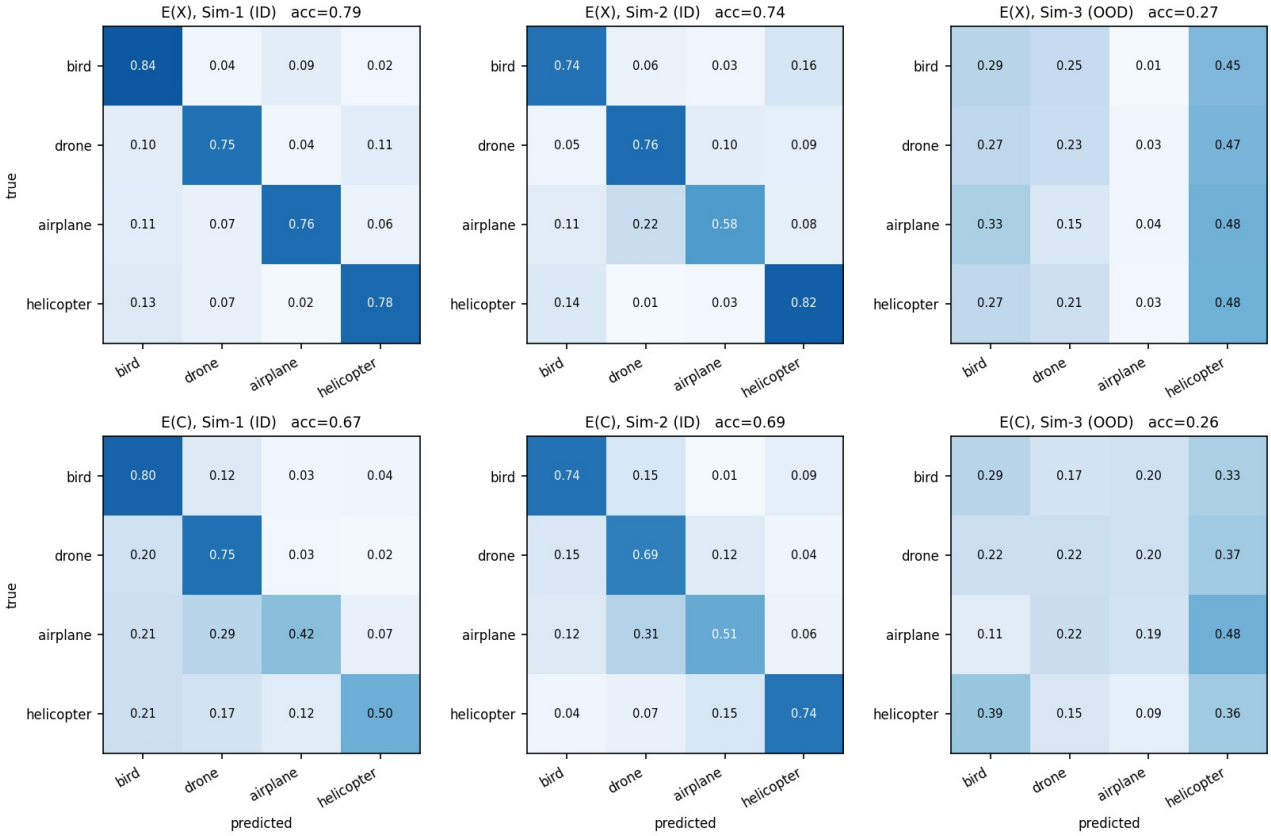


Figure E.2: Clip-level confusion matrices for the $E(X)$ (top row) and $E(C)$ (bottom row) entries of Table 6.18, row-normalized and pooled over seeds 42/43/44, under the same split and regime layout as Figure E.1. Clip-level pooling scores each fixed-length window independently rather than voting over the whole sequence, so these panels match the clip-level figures in the table note.