3D Occupancy Prediction, Classification and Tracking 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 DATE.

Student number:	5885396	
Project duration:	x	
Thesis committee:	Prof. Dr. Peter Hofstee	TU Delft, Supervisor
	Dr. S.T.A.F.F. Member	TU Delft
	Wim Bos	Lumiad BV, External supervisor

An electronic version of this thesis is available at link/.

[**TODO:** This is where you thank people for helping you etc.]

Delft, the Netherlands
January 14, 2026

/todowrite abstract here

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
$\mathbf{P}_c$	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
$\Delta\mathrm{rot}$	Temporal change in rotation

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
BA	Bundle Adjustment
COLMAP	Structure-from-Motion software package
DINOv3	Self-supervised vision transformer (third version)
ECE	Expected Calibration Error
FFT	Fast Fourier Transform
FPS	Farthest Point Sampling (or Frames Per Second, context-dependent)
F1	F1-score (harmonic mean of precision and recall)
HOTA	Higher Order Tracking Accuracy
IDF1	ID F1-score (tracking metric)
InfoNCE	Contrastive loss function
IoU	Intersection over Union
MAE	Mean Absolute Error
MLP	Multi-Layer Perceptron
PnP	Perspective-n-Point (camera pose estimation)
PSNR	Peak Signal-to-Noise Ratio
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)
SE(3)	Special Euclidean group (3D rotations and translations)
SfM	Structure-from-Motion
SH	Spherical Harmonics
SNR	Signal-to-Noise Ratio
SSIM	Structural Similarity Index Measure
TBD	Track Before Detect
DBT	Detection Before Track
ViT	Vision Transformer

1 Introduction

1.1 Motivation and Context

The rapid increase in small unmanned aerial vehicles (UAVs) and the growing need to monitor airspace have created big challenges for security systems. It is critical to distinguish between authorized drones, unauthorized drones, and natural flying objects like birds. This is required for many applications, such as airport security, protecting vital infrastructure, and monitoring wildlife.

Standard methods for detecting and tracking aerial objects rely on special hardware like RADAR systems, RF receivers, or directional microphones. Although these solutions can be effective, they are either expensive or have limitations in range, accuracy, or susceptibility to interference.

We therefore explore a different approach in this thesis. We use multiple RGB cameras located on the ground with overlapping fields of view to predict 3D occupancy, classify objects, and track them. By using the geometric information from multiple views and the changes in 3D reconstructions over time, we aim to create a system that runs on standard hardware while achieving high accuracy.

The combination of 3D Gaussian splatting and classification over time for flying objects is a new area of research. Gaussian splatting has changed real time 3D rendering, and its 4D extensions capture dynamic scenes effectively. However, no existing work applies these features over time to classify objects.

The Drone vs Bird Detection Challenge serves as the primary benchmark for this field. Our research aims to surpass the current best known scores from this challenge by introducing multiple camera configurations and 3D reconstruction techniques to enhance classification accuracy. Through these improvements we hope to steer the challenge towards a multiple view approach.

1.2 Problem Statement

Given a network of $N \geq 5$ fixed RGB cameras with overlapping fields of view monitoring a central airspace volume, we address the following problem:

> *How can we detect, reconstruct, classify, and track flying objects (specifically drones and birds) 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?*

The system must operate under realistic constraints:

- **Input:** Synchronized RGB frames from multiple cameras with known (but potentially imperfect) intrinsics and extrinsics
- **Output:** 3D occupancy predictions, object classifications (drone vs. bird vs. background), and temporally consistent object tracks
- **Constraints:** Real-time or near-real-time processing (potentially on edge computing hardware e.g., NVIDIA Jetson)

- **Challenges:** Distant objects with low pixel resolution, textureless sky backgrounds, varying object scales across cameras, and camera calibration drift

1.3 Research Questions

This thesis investigates the following research questions:

1. **RQ1: Foreground Segmentation**
   How can we effectively segment dynamic flying objects in 3D space without knowledge of their appearance across multiple views and time?
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 textureless 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 and tracking performance can be achieved with the pipeline on both synthetic (Isaac Sim) and real-world datasets, and how does this compare to baseline 2D and 3D approaches?

1.4 Approach

Our approach integrates multiple components into an end-to-end pipeline:

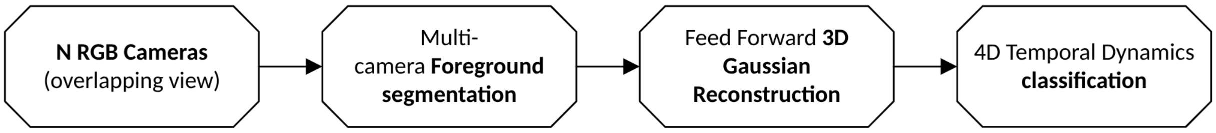
N RGB Cameras (overlapping view)
Multi-camera Foreground segmentation
Feed Forward 3D Gaussian Reconstruction
4D Temporal Dynamics classification

Figure 1.1: Overview of the proposed multi-camera 4D classification pipeline for flying object detection and tracking.

We generate synthetic training data using NVIDIA Isaac Sim to simulate 3D objects like drones and birds in various scenes. This includes realistic camera effects and changing environmental conditions, along with ground truth annotations. For segmentation, we use a multi camera foreground segmentation technique. Instead of detecting objects in each camera separately, we use a track before detect strategy. This method identifies 3D regions visible in at least three cameras using ray marching and accumulates pixel differences over time to capture dynamic objects.

We then apply feed forward Gaussian splatting to quickly reconstruct 3D Gaussian representations from multi view RGB frames. A [**TODO:** classifier architecture] based classifier processes these sequences to analyze 4D temporal dynamics. It extracts features from changes in Gaussian parameters and uses rotation invariant architectures to classify objects as drones, birds, or unknown based on their motion.

1.5 Key Contributions

This thesis makes the following contributions:

1. **Novel 4D Classification Architecture:** A [**TODO:** classifier architecture] classifier that operates directly on temporal sequences of 3D Gaussian representations, exploiting the combination of static and temporal dynamics of Gaussian parameters for improved classification of dynamic flying objects.
2. **Multi-View Foreground Segmentation:** A ray-marching-based volumetric voting approach for dynamic object segmentation that accumulates evidence across views and time, enabling detection of small, low-contrast, fast flying objects.
3. **Real-Time Pipeline:** An end-to-end system integrating synthetic data generation, multi-view track-before-detect, feed-forward 3D reconstruction, 4D classification, optimized for real-time or near-real-time operation.

4. **Synthetic Data Generation Pipeline:** A comprehensive framework for generating realistic synthetic datasets in Isaac Sim, including diverse flying object models, camera effects, and environmental conditions, along with ground truth annotations for training and evaluation.
5. **Open-Source Implementation:** A complete implementation of the pipeline, with synthetic data generation tools for Isaac Sim.

1.6 Thesis Scope and Limitations

1.6.1 In Scope

- Detection, 3D reconstruction, classification, and tracking of flying objects
- Classification: drone vs. bird (with potential extension to unknown class)
- Scenarios with $N \geq 5$ cameras with overlapping fields of view
- Objects flying through a monitored central volume
- Both synthetic (Isaac Sim) and real-world evaluation

1.6.2 Out of Scope

- Fine-grained drone model recognition or bird species classification
- Long-range tracking beyond the monitored volume
- Adversarial scenarios (e.g., drones designed to evade detection)
- Privacy and legal aspects of surveillance systems
- Real-time guarantees with formal verification

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 for the three main pillars to contextualize our approach.
- **Chapter 4** presents a detailed review of related work, organized by technical component.
- **Chapter 5** describes the complete pipeline in detail, explicitly justifying each major design choice based on the comparative analysis from Chapter 3.
- **Chapter 6** presents our 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 our multi-camera flying object detection and tracking system. Design constraints are explicitly stated to provide context for the architectural decisions made in subsequent chapters.

2.1 Scenario: Multi-Camera Airspace Monitoring

2.1.1 Monitored Volume

We consider a ground-based surveillance system monitoring a central airspace volume $\mathcal{V} \subset \mathbb{R}^3$ defined in a world coordinate frame. The volume is specified by:

- **Horizontal extent:** A polygonal or circular region (typically 25-50m radius)
- **Vertical extent:** Ground level to maximum monitoring altitude (typically 25-50m)
- **Volume shape:** Intersection of camera frustums defining the observable region

Objects of interest (drones and birds) enter, traverse, and exit this volume. Our goal is to detect, reconstruct, classify, and track them *while they are within* $\mathcal{V}$.

2.1.2 Multi-Camera Configuration

The system consists of $N \geq 5$ RGB cameras positioned on the ground with upward-looking orientations. See Figure 2.1 for an illustrative setup. The cameras satisfy:

- **Overlapping fields of view:** Each point in $\mathcal{V}$ should ideally be visible to at least 5 cameras to enable reliable 3D reconstruction
- **Inward-looking configuration:** Cameras are arranged to point toward a common central region
- **Fixed mounting:** Cameras are statically mounted (e.g., on poles, buildings, towers) with rigid mounts to minimize pose changes
- **Temporal synchronization:** Cameras are time-synchronized via NTP to within $\pm$10 ms
- **Network connectivity:** All cameras are connected to a central processing unit (edge device) via a local network switch

This configuration is typical of fixed surveillance installations for perimeter security, airport monitoring, or protected area surveillance.

2.1.3 Flying Object Characteristics

We focus on two object categories:

- **Drones (Quadcopters and Multicopters)** Physics-constrained motion with relatively smooth trajectories and hovering capability. Rigid and symmetric structures, with spinning rotors in high-resolution views.
- **Birds** Flapping flight with gliding and more varied motion than drones. Deformable body with wing articulation and biological texture patterns.

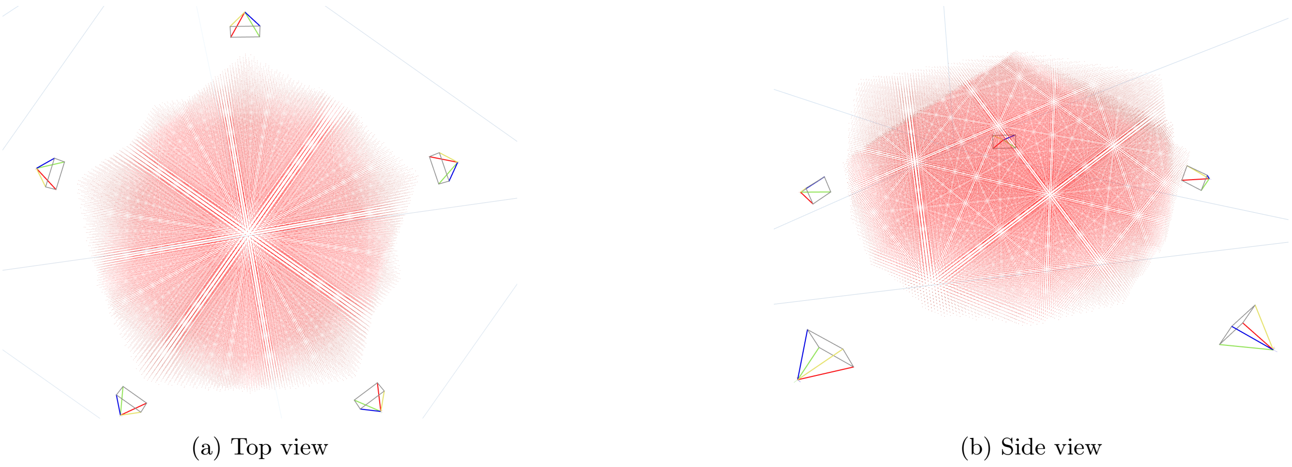
(a) Top view (b) Side view

Figure 2.1: Occupancy grid representation of monitored volume $\mathcal{V}$. Cameras are positioned around the perimeter, looking upward into the volume where flying objects (red) are detected and tracked.

2.1.4 Scene Constraints

The monitoring scenario imposes several challenging constraints:

- **Textureless background:** The primary background is open sky, which provides minimal texture features for traditional feature matching or depth estimation methods
- **Distant small objects:** Flying objects operate within the central volume of overlapping views. With distances ranging from 10–100m (given a camera radius of 100m), targets occupy only a relatively small number of pixels in each camera frame
- **Varying scales across views:** Objects may be significantly closer to some cameras than others, resulting in scale variation across views
- **Environmental variations:** Lighting changes (sun position, clouds), weather conditions (haze, rain), and background clutter (clouds, buildings at horizon) affect appearance
- **Occlusions:** Temporary occlusions may occur due to clouds, other flying objects, or camera-specific obstacles

These constraints guide our design choices, specifically the use of geometric constraints from multiple views and temporal dynamics rather than relying on features based on appearance alone.

2.1.5 Pixel Density and Resolution Limits

A fundamental constraint in our long-range monitoring scenario is the low spatial resolution of distant targets. Even with high-resolution 4K sensors, the large monitored volume results in insufficient pixel density for standard single-frame classification.

Standardized Requirements and System Capabilities The IEC 62676-4 standard [1] defines image quality levels based on pixel density. For reliable "Recognition" (distinguishing known individuals) or "Identification," the standard requires densities of 125 PPM (pixels per meter) and 250 PPM, respectively. "Detection" requires only 25 PPM.

In our setup, a 2K camera (2560×1440) with a 90° horizontal field of view yields a pixel density of approximately $PPM(d) \approx 1280/d$. At $d = 100$m, the density drops to ≈ 12.8 PPM which is barely sufficient for monitoring traffic flow and well below the threshold for object recognition. A 0.5m drone at this distance occupies only ≈ 6.4 pixels, meaning high-frequency spatial details are lost to aliasing [2].

Theoretical Limits and Temporal Integration Information theory formalizes this limitation. The capacity of the imaging channel is bounded by the spatial bandwidth (pixels on target) and Signal-to-Noise Ratio (SNR) [2]. When a target spans fewer than $\approx 10 - 20$ pixels, the spatial bandwidth is too low to encode unique class features, and noise further degrades the effective bit depth per pixel.

However, the total information capacity can be recovered by integrating observations over time. If the target is tracked over T frames, the effective channel capacity scales with T, provided that motion or rotation reveals new information [3]. This motivates our approach: we compensate for the lack of spatial bandwidth ($B_{spatial}$) by exploiting temporal bandwidth ($B_{temporal}$), using motion dynamics and multi-view consistency to classify objects that are spatially irresolvable in any single frame.

2.2 Assumptions

2.2.1 Camera Calibration

Intrinsic Parameters We assume camera intrinsic matrices $\mathbf{K}_i$ (focal lengths, principal points, distortion) are known via standard offline calibration. While environmental factors may cause slight drift, we treat these parameters as fixed for the primary workflow.

Extrinsic Parameters We assume camera extrinsic matrices $[R_i|t_i]$ (rotation and translation from world to camera frame) are initialized via offline calibration methods, such as static scene Structure-from-Motion (SfM) or manual alignment with ground control points. In security contexts (e.g., airports, perimeter monitoring), the camera positions t_i are often known from installation blueprints or GPS, but the orientations R_i are prone to inaccuracy and drift. These parameters are also fixed for the primary workflow.

2.2.2 Temporal Synchronization

Cameras are assumed to be synchronized via NTP or PTP to within ± 10 ms. At typical object speeds (0–15 m/s), this results in an acceptable positional uncertainty of $\approx$ 15 cm. Residual synchronization errors are treated as time offsets.

2.2.3 Data Availability During Training vs. Inference

Training Phase (Synthetic Data from Isaac Sim) During training on synthetic data, we have access to:

- Synchronized RGB frames from all cameras
- **Ground truth depth maps** for each camera
- **Pixel-wise segmentation masks** indicating foreground objects
- **3D object trajectories** (positions, orientations) in world frame
- **Object class labels** (drone model, bird species)
- Perfect camera intrinsics and extrinsics (no calibration error)

Inference Phase (Real-World Deployment) During inference on real-world data, we have access to **only**:

- Synchronized RGB frames from all cameras
- Known camera intrinsics $\mathbf{K}_i$
- Known camera extrinsics $[\mathbf{R}_i|\mathbf{t}_i]$

This sim-to-real transfer is a central challenge addressed in Chapter 6.

2.3 Edge Computing Goal

We target deployment on **NVIDIA Jetson** edge computing platforms:

- Representative platform(s):
  - Jetson Orin Nano Super (8GB, 67 TOPS INT8, $250)
  - Jetson AGX Orin (32GB, 275 TOPS INT8, $2000)
- 5 cameras $\times$ 2560$\times$1440 @ 30 FPS $\approx$ 165 megapixels/sec
- Shared network bandwidth: 1 Gbps Ethernet switch
- Target latency: < **[TODO:** 1] second from frame capture to classification output

2.4 Evaluation Criteria

We evaluate the system along multiple dimensions:

Table 2.1: Evaluation metrics categorized by system aspect.

Category	**Metric**	**Description**
Classification	Accuracy	Overall correct classification rate (drone vs. bird)
	Precision & Recall	Per-class metrics to identify biases
	F1 Score	Harmonic mean of precision and recall
	Confusion Matrix	Detailed error analysis (false positives/negatives)
3D Reconstruction	PSNR / SSIM	Novel view synthesis quality (3DGS evaluation)
	Temporal Consistency	Smoothness of object motion across frames
Tracking	MOTA	Multi-Object Tracking Accuracy
	IDF1	ID F1 score (track consistency)
	HOTA [4]	Higher-Order Tracking Accuracy
	3D Position Error	Euclidean distance to ground truth
Computational	Latency	End-to-end processing time
	Throughput	Frames per second (FPS) on target hardware
	Memory Footprint	Peak GPU/CPU memory usage

2.5 Success Criteria

We defined several success criteria for the proposed system:

Metric	**Baseline**	**Target**
Classification Accuracy (Synthetic)	[**TODO:** 85%]	$> 95\%$
Classification Accuracy (Real-World)	[**TODO:** 70%]	$> 85\%$
3D Position Error (RMSE)	[**TODO:** < 2.0m]	< 0.5m
Tracking IDF1	[**TODO:** 60%]	$> 80\%$
Inference Latency	[**TODO:** < 2s]	< 500ms

Table 2.2: Success criteria for system performance. Baselines represent typical performance of 2D-only or per-frame 3D methods from [**TODO:** literature]; targets represent our goals for the proposed 4D temporal dynamics approach.

2.6 Chapter Summary

This chapter discussed:

- The multi-camera airspace monitoring scenario with central volume $\mathcal{V}$
- Hardware configuration: $N \geq 3$ upward-looking synchronized RGB cameras
- Object characteristics: small distant drones and birds against sky backgrounds
- Sensing assumptions: known intrinsics and extrinsics
- Data availability: rich synthetic training data vs. RGB-only real-world inference
- Edge computing target: NVIDIA Jetson platform, latency requirements
- Evaluation criteria: classification, reconstruction, tracking, computational metrics

3 Background

This chapter reviews basic concepts in multi view geometry, 3D Gaussian Splatting, and deep learning primitives. These concepts describe the methods developed in this thesis. Section 3.1 introduces the geometric principles that allow 3D reconstruction from multiple camera views. Section 3.2 explains the math behind 3D Gaussian Splatting as an explicit scene representation. Finally, Section 3.3 looks at neural network architectures and mechanisms (attention, equivariance, permutation invariance, and state space models). These form the building blocks of our temporal classification pipeline.

3.1 Multi View Geometry Fundamentals

Multi view geometry provides the mathematical framework to understand three dimensional structure from two dimensional images [5], [6]. This section explains the geometric basics of our multi camera reconstruction pipeline. It covers individual camera models and the constraints that appear when multiple cameras look at the same scene. [**TODO:** SfM, MVS]

3.1.1 Camera Models and Projection

We use the pinhole camera model. It simplifies image formation as a central projection through a very small hole [5]. A 3D point $\mathbf{X} = (X, Y, Z)^\top$ in world coordinates projects to a 2D image point $\mathbf{x} = (u, v)^\top$ according to:

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

where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{X}}$ are homogeneous coordinates. The projection splits into intrinsic and extrinsic parts [5].

The intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ contains the optical properties of the camera [5]:

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

where f_x, f_y are focal lengths in pixels, (c_x, c_y) is the principal point, and s captures sensor skew (usually zero for modern cameras). These parameters stay fixed after calibration [7], [8].

The extrinsic parameters $[\mathbf{R} \mid \mathbf{t}]$ define the camera pose in world coordinates: rotation $\mathbf{R} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$ [5]. Together, they transform world points into the local coordinate frame of the camera before projection.

3.1.2 Epipolar Geometry

When two cameras look at the same scene, their geometric relationship limits where matching points can appear [5], [9]. Consider cameras with projection matrices $\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]$ and $\mathbf{P}' = \mathbf{K}'[\mathbf{R}' \mid \mathbf{t}']$. A 3D point $\mathbf{X}$ projects to $\mathbf{x}$ in the first image and $\mathbf{x}'$ in the second. These matches satisfy the epipolar constraint [5], [9]:

$$\mathbf{x}'^\top \mathbf{F}\mathbf{x} = 0 \tag{3.3}$$

where $\mathbf{F} \in \mathbb{R}^{3\times 3}$ is the fundamental matrix. It is a rank 2 matrix that describes the relative geometry between pairs of cameras [5].

Geometrically, this constraint means that $\mathbf{x}'$ must lie on the *epipolar line* $\mathbf{l}' = \mathbf{F}\mathbf{x}$. This line is the projection of the ray from the first camera through $\mathbf{x}$ onto the second image plane [5]. This reduces the search for matches from 2D to 1D, a principle used in stereo matching.

For calibrated cameras, the essential matrix $\mathbf{E} = \mathbf{K}'^{\top}\mathbf{F}\mathbf{K}$ gives a metric version of this constraint. It encodes only rotation and translation (up to scale) [5], [9]. We have $\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}$, where $[\mathbf{t}]_{\times}$ is the skew symmetric cross product matrix.

3.1.3 Triangulation and 3D Point Recovery

Given matching image points $\mathbf{x}, \mathbf{x}'$ and known camera matrices, triangulation recovers the 3D point $\mathbf{X}$ that projects to both observations [5]. Geometrically, we look for the intersection of two back projected rays. Because of noise, these rays rarely intersect exactly. Practical methods minimize reprojection error.

Linear triangulation sets up the problem as a homogeneous system [5]. From $\tilde{\mathbf{x}} \times \mathbf{P}\tilde{\mathbf{X}} = \mathbf{0}$ (the cross product of parallel vectors is zero), we get two independent equations per view. Stacking constraints from multiple views creates an overdetermined system $\mathbf{A}\tilde{\mathbf{X}} = \mathbf{0}$. This is solved via SVD to find the unit singular vector that corresponds to the smallest singular value [5].

The accuracy of triangulation depends heavily on the ratio between the baseline and the depth. Cameras that are far apart give more precise depth estimates than cameras that are close together [5]. For a flying object at distance d with baseline b and pixel noise σ, depth uncertainty scales roughly as $\sigma_d \propto d^2/(b \cdot f)$. This is why we distribute cameras around the monitored volume instead of grouping them together.

3.1.4 Visual Hulls and Silhouette Based Reconstruction

When object silhouettes are available from multiple views, visual hull methods offer another way to reconstruct objects without using features [10]. The visual hull is defined as the intersection of viewing cones. These are the 3D regions that project inside the object silhouette in each image [10].

Formally, let $S_i \subset \mathbb{R}^2$ be the silhouette region in camera i. The viewing cone $\mathcal{C}_i$ consists of all 3D points projecting into S_i. The visual hull is:

$$\mathcal{V} = \bigcap_{i=1}^{N} \mathcal{C}_i \tag{3.4}$$

As described by Laurentini [10], visual hulls provide an *outer bound* on object geometry. They match the true shape only for convex objects seen from enough angles. Hollow parts that cannot be seen in the silhouettes are not removed. However, visual hulls have computational advantages. They only require foreground segmentation (not dense matches) and handle objects without texture well.

Practical implementations divide the volume into a 3D grid (voxel carving). They mark voxels as occupied if they project inside all silhouettes [11]. This volumetric voting approach extends naturally to probabilistic methods where each camera adds evidence for occupancy [11].

3.1.5 Connection: Why $N \geq 3$ Cameras Enable Robust 3D Inference

The setup of multiple cameras in this thesis uses $N \geq 3$ cameras facing upwards with overlapping fields of view. This setup uses the geometric ideas described before.

While two cameras create epipolar constraints, the depth stays unclear along epipolar lines in regions without texture. A third camera adds intersecting constraints that solve this problem. In our scenario with a sky background, where matching based on light intensity fails, this geometric redundancy is necessary.

With three or more views, linear triangulation becomes overdetermined. Errors in one view are fixed by information from the others. This improves resistance to segmentation errors and temporary occlusions.

The foreground segmentation in our pipeline (Section **??**) extends the principles of the visual hull. We classify 3D regions as occupied when they project to the detected foreground in at least 3 cameras. This threshold ensures robustness against false positives in individual views while keeping sensitivity to real objects.

Flying objects at a range of 10 to 100 meters need large baselines to recover depth accurately. Our placement of distributed cameras keeps good ratios between the baseline and depth despite the large distances.

These geometric foundations support the processing stages of our pipeline. They include projection, epipolar constraints, triangulation, and volumetric reconstruction. These elements allow 3D inference from RGB observations alone. [**TODO:** FIGURE: [Suggested diagram showing camera projection geometry]]

3.2 3D Gaussian Splatting Formulation

Neural Radiance Fields (NeRF) revolutionized novel view synthesis by representing scenes as continuous volumetric functions rather than discrete meshes or points, but they require costly sampling for each ray during rendering [12]. 3D Gaussian Splatting (3DGS) offers an alternative. It is an explicit scene representation that uses groups of anisotropic Gaussian primitives. These can be rendered via efficient rasterization instead of ray marching [13]. This section introduces the math behind Gaussian based scene representations.

Gaussian Primitive Parameterization A 3D Gaussian Splatting representation consists of a set $\mathcal{G} = \{G_i\}_{i=1}^N$ of N Gaussian primitives. Each primitive G_i is defined by the following parameters [13]:

The mean $\boldsymbol{\mu} \in \mathbb{R}^3$ specifies the center of the Gaussian in world coordinates.

Instead of storing the full covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ directly, 3DGS splits it into a scale vector $\mathbf{s} \in \mathbb{R}^3$ and a rotation quaternion $\mathbf{q} \in \mathbb{R}^4$. The covariance is reconstructed as:

$$\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T \tag{3.5}$$

where $\mathbf{S} = \text{diag}(\mathbf{s})$ is a diagonal scaling matrix and $\mathbf{R}$ is the rotation matrix derived from the unit quaternion $\mathbf{q}$. This split guarantees positive semi definiteness. It also reduces parameters from 6 (symmetric matrix) to 7 (scale + quaternion), and the quaternion constraint ensures valid rotations.

A scalar $\alpha \in [0, 1]$ controls the contribution of the Gaussian to the rendered image. Low opacity creates semi-transparent effects. High opacity produces solid surfaces.

Appearance is encoded using spherical harmonic (SH) coefficients that model view dependent color variation. For degree ℓ, each Gaussian stores $(\ell+1)^2$ coefficients per color channel. Degree 0 captures diffuse color. Higher degrees encode specular reflections. At render time, SH coefficients are evaluated for the viewing direction to produce RGB values $\mathbf{c} \in \mathbb{R}^3$.

The complete parameter vector for a single Gaussian is thus:

$$\theta_i = \{\boldsymbol{\mu}_i, \mathbf{s}_i, \mathbf{q}_i, \alpha_i, \mathbf{c}_i^{\text{SH}}\} \tag{3.6}$$

where $\mathbf{c}_i^{\text{SH}}$ denotes the SH coefficient tensor.

[**TODO:** FIGURE: [Gaussian primitive visualization]]

Differentiable Rasterization Rendering a Gaussian scene requires projecting all primitives to screen space and combining their contributions. Unlike volumetric ray marching, 3DGS uses *splatting*: each Gaussian is projected as a 2D ellipse and rendered directly [13].

The projection of a 3D Gaussian to 2D follows the Elliptical Weighted Average (EWA) splatting framework [14]. Given camera extrinsics $[\mathbf{R}_c|\mathbf{t}_c]$ and the Jacobian $\mathbf{J}$ of the projective transformation at the Gaussian center, the 2D covariance becomes:

$$\boldsymbol{\Sigma}' = \mathbf{J}\mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^T\mathbf{J}^T \tag{3.7}$$

where $\mathbf{W}$ is the viewing transformation matrix. The resulting $\boldsymbol{\Sigma}' \in \mathbb{R}^{2\times2}$ defines an elliptical footprint on the image plane. For a pixel at position $\mathbf{p}$, the contribution of the Gaussian is weighted by:

$$G(\mathbf{p}) = \exp\left(-\frac{1}{2}(\mathbf{p} - \boldsymbol{\mu}')^T{\boldsymbol{\Sigma}'}^{-1}(\mathbf{p} - \boldsymbol{\mu}')\right) \tag{3.8}$$

where $\boldsymbol{\mu}'$ is the projected 2D mean.

To achieve real time performance, the image is divided into tiles (typically 16×16 pixels). Gaussians are sorted by depth and assigned to tiles they overlap. Each tile then processes only its relevant Gaussians, which allows parallel GPU execution [13]. This tile based approach avoids the per pixel ray sampling of NeRF style methods.

Alpha Compositing The final pixel color results from front-to-back alpha compositing of depth-sorted Gaussians. Let $\{G_1, \ldots, G_M\}$ be the Gaussians overlapping a pixel, sorted by increasing depth. The rendered color is:

$$C = \sum_{i=1}^{M} c_i \alpha_i G_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - \alpha_j G_j(\mathbf{p})) \tag{3.9}$$

The product term $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j G_j(\mathbf{p}))$ represents *transmittance*. This is the fraction of light reaching Gaussian i after absorption by previous Gaussians. Rendering stops early when accumulated opacity approaches 1, which provides additional speedup.

[**TODO:** FIGURE: [EWA projection and compositing pipeline]]

Optimization Objective The standard 3DGS training objective combines photometric losses [13]:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{D-SSIM}} \tag{3.10}$$

where $\mathcal{L}_1 = \|C_{\text{pred}} - C_{\text{gt}}\|_1$ measures pixel-wise absolute error, $\mathcal{L}_{\text{D-SSIM}} = 1 - \text{SSIM}(C_{\text{pred}}, C_{\text{gt}})$ captures structural similarity [15], and $\lambda = 0.2$ balances the two terms. The D-SSIM component encourages perceptually coherent reconstructions beyond pixel accuracy.

Optimization proceeds via gradient descent on all Gaussian parameters. Crucially, the entire pipeline is differentiable. This allows end-to-end training from multi-view images.

3.2.1 Connection to Flying Object Reconstruction

The Gaussian splatting formulation offers several properties that are suitable for our surveillance setting:

- Unlike implicit neural representations that require dense sampling, Gaussians provide explicit 3D positions and extents. This allows direct geometric reasoning about the location and size of the object, which is critical for tracking.
- Rasterization based on tiles achieves rendering speeds of over 100 FPS. This is orders of magnitude faster than volume rendering. This permits rapid testing of hypotheses and verification of consistency across multiple views.
- The decomposed parameterization (position, scale, rotation, opacity, color) provides a structured representation. Temporal changes in these parameters (such as the rigid motion of a drone versus the wing articulation of a bird) form discriminative features for classification.
- Gaussians can represent scenes from fewer views than methods based on dense stereo require, as each primitive contributes to the reconstruction independently.

These properties make 3D Gaussians natural intermediate representations for our pipeline. Reconstructed Gaussians from frames with multiple views become input tokens to the temporal classifier introduced in Section 3.3.

3.3 Deep Learning Primitives

This section introduces the architectural components that support the 4D temporal classifier developed in this thesis. We start with attention mechanisms and Transformers. Then we look at symmetry constraints that are essential for processing 3D data. Finally, we discuss state space models that offer computational benefits for real-time deployment.

3.3.1 Self Attention and Transformer Architecture

The attention mechanism allows neural networks to weight input elements dynamically based on how relevant they are to a specific query. Consider a sequence of N input tokens. Each token is a d dimensional vector. We organize these tokens into three matrices: queries $\mathbf{Q} \in \mathbb{R}^{N \times d_k}$, keys $\mathbf{K} \in \mathbb{R}^{N \times d_k}$, and values $\mathbf{V} \in \mathbb{R}^{N \times d_v}$. These are obtained through learned linear projections of the input.

Scaled dot product attention calculates a weighted combination of values:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{3.11}$$

The scaling factor $\sqrt{d_k}$ stops the dot products from becoming too large. Large values would push the softmax into regions with vanishing gradients. Intuitively, each query calculates similarity scores against all keys. Then it collects values based on these scores.

Multi head attention extends this mechanism. It runs H parallel attention operations, each with independent projections:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_H)\mathbf{W}^O \tag{3.12}$$

where $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$ and $\mathbf{W}^O$ projects the concatenated outputs. Multiple heads allow the model to attend to information from different representation subspaces at the same time.

The Transformer architecture [16] stacks layers of multi-head self-attention with position-wise feed-forward networks. The Vision Transformer (ViT) [17] adapts this design for images. It splits an image into fixed size patches, linearly embeds each patch, and adds learnable position embeddings to encode spatial structure.

A critical factor for our application is computational complexity. Self-attention calculates pairwise interactions between all tokens. This results in $O(N^2)$ time and memory complexity with respect to sequence length N. For long sequences of Gaussian primitives or extended temporal windows, this quadratic scaling can be too much. This motivates the state space alternatives discussed in Section 3.3.4.

[**TODO:** FIGURE: [diagram showing the self-attention computation flow]]

3.3.2 SE(3) Equivariance

Flying objects seen from ground cameras may appear in any orientation. A drone flying at an angle from the north looks different than one coming from the east, but both should be classified the same way. This observation motivates adding geometric symmetries to neural architectures.

The special Euclidean group SE(3) includes all rigid transformations in three dimensional space: rotations combined with translations. Formally, an element $g \in \text{SE}(3)$ acts on a point $\mathbf{x} \in \mathbb{R}^3$ as $g \cdot \mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{t}$. Here $\mathbf{R} \in \text{SO}(3)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector.

We distinguish two related but different properties. A function f is invariant to SE(3) transformations if:

$$f(g \cdot \mathbf{x}) = f(\mathbf{x}) \quad \forall g \in \text{SE}(3) \tag{3.13}$$

The output stays the same regardless of how the input is transformed. Classification tasks often require invariance. A drone or bird should be recognized as such regardless of its orientation.

A function f is equivariant to SE(3) transformations if:

$$f(g \cdot \mathbf{x}) = g \cdot f(\mathbf{x}) \quad \forall g \in \text{SE}(3) \tag{3.14}$$

The output transforms in a predictable way with the input. Equivariance in intermediate layers keeps geometric information. This can be combined into invariant predictions at the final layer.

Since our application will be deployed on edge devices, the trade-off between guaranteed equivariance and computational cost becomes critical. Some approaches provide practical equivariance suitable for real-time inference, whereas full spherical harmonic methods may be too expensive.

3.3.3 Permutation Invariance for Set Processing

A set of 3D Gaussian primitives that reconstruct a flying object has no inherent order. Whether we list Gaussians from front to back, largest to smallest, or randomly, the underlying object stays the same. Neural networks working on such data must respect this permutation symmetry.

A function f acting on a set $\mathcal{X} = \{x_1, \dots, x_n\}$ is permutation invariant if:

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

for any permutation π. The basic result from DeepSets [18] states that any permutation-invariant function on finite sets can be decomposed as:

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

where ϕ encodes individual elements and ρ processes the aggregated representation. The summation (or equivalently, mean pooling) results in permutation invariance.

These principles translate directly to Gaussian representations. Each Gaussian primitive is a set element. It is parameterized by position $\boldsymbol{\mu}$, scale $\mathbf{s}$, rotation $\mathbf{q}$, opacity α, and color coefficients. A permutation invariant encoder can process this set to produce a fixed dimensional representation suitable for classification.

3.3.4 State Space Models

Transformers achieve remarkable performance across sequence modeling tasks. However, they suffer from quadratic complexity that limits scalability to long sequences. State space models (SSMs) offer an alternative with linear complexity. This allows efficient processing of long time windows.

Classical linear time invariant SSMs are defined in continuous time as:

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

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

where $\mathbf{h}(t) \in \mathbb{R}^D$ is the hidden state, $x(t)$ is the input, and $y(t)$ is the output. The matrices $\mathbf{A} \in \mathbb{R}^{D\times D}$, $\mathbf{B} \in \mathbb{R}^{D\times 1}$, and $\mathbf{C} \in \mathbb{R}^{1\times D}$ control the state dynamics. For discrete sequence processing, these equations are discretized using a step size Δ to obtain:

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

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

where $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are discretized versions of the continuous parameters.

The Mamba architecture [19] extends this foundation with selective state spaces. Unlike classical SSMs with fixed dynamics, Mamba makes the parameters $\mathbf{B}$, $\mathbf{C}$, and Δ functions of the input:

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

This input dependence allows the model to selectively propagate or forget information based on content. It combines the efficiency of SSMs with the context awareness of attention mechanisms.

The computational advantage is substantial. While Transformer self-attention requires $O(N^2)$ operations for a sequence of length N, Mamba achieves $O(N)$ complexity through its recurrent formulation. For temporal classification of Gaussian sequences, we may process dozens of frames each containing hundreds of primitives. In this case, linear scaling could enable real-time operation that would be challenging with quadratic methods.

3.3.5 Connection to Thesis Architecture.

The primitives introduced in this section form the computational foundation of the 4D classifier developed in Chapter **[TODO:** ref chapter]. Reconstructed Gaussians serve as input tokens processed by a permutation-invariant encoder. SE(3) equivariant layers ensure classification remains consistent regardless of object orientation. Temporal modeling, whether through Transformer attention or Mamba's selective SSM, captures the dynamic signatures that distinguish drone flight patterns from bird locomotion. The efficiency of state space models proves particularly valuable for deployment on edge computing hardware. Here, the target latency of under 500 milliseconds limits architectural choices.

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

As mentioned in Chapter 1, this thesis integrates multiple components into an end-to-end pipeline for flying object detection and tracking. In this chapter, we review related work organized by technical component, highlighting key advances and identifying gaps that motivate our approach.

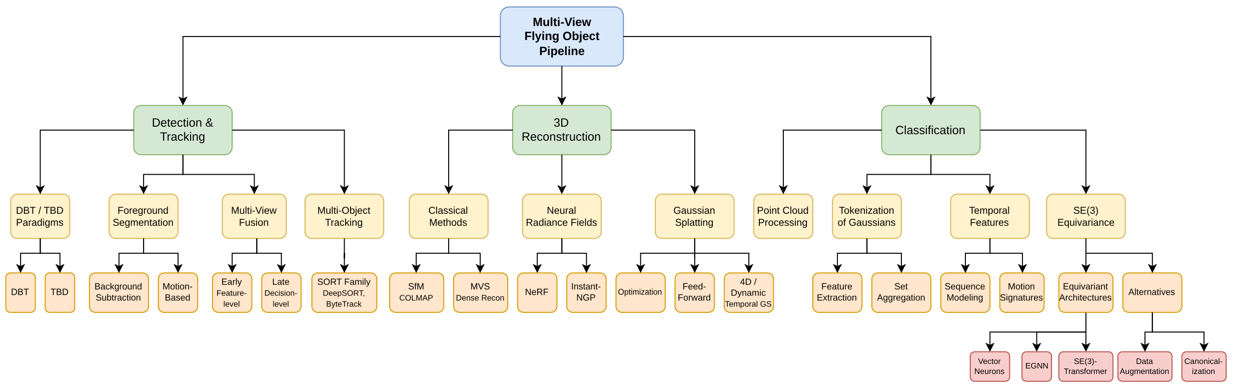

Figure 4.1: Overview of the different components of the literature review to accommodate the proposed approach.

4.1 Detection & Tracking of Flying Objects

4.1.1 Detection Paradigms: DBT vs. TBD

The choice between Detect-Before-Track (DBT) and Track-Before-Detect (TBD) paradigms depends on target resolution and signal-to-clutter ratios (SCR). DBT approaches dominate real-time applications; YOLO variants [20] and transformer-based detectors achieve inference speeds of 50–130 FPS on resolved targets. These architectures fail systematically when objects fall below 20 pixels or have backgrounds with high complexity (e.g. vegetation or buildings). Deep feature downsampling reduces tiny targets to negligible information, while standard anchor designs miss objects smaller than 16×16 pixels entirely. While TBD methods excel at weak signals (SCR ≈ 1.5) by integrating multi-frame evidence, neither paradigm adequately leverages multi-view 3D constraints for robust aerial detection.

Background complexity dominates detection performance

The Drone-vs-Bird Detection Challenge, now in its 8th edition at IJCNN 2025, provides the most comprehensive benchmark for aerial object detection [21]. The 2025 winner achieved 73.7% mAP using YOLOv11m with multi-scale processing, where input images are processed both as a whole and in overlapping segments to improve small drone detection. Background type dramatically affects detection accuracy: sky backgrounds yield median AP of 51% with relatively consistent performance, vegetation backgrounds produce high variance (median 26%, range 0–89%), while buildings and mixed backgrounds prove most challenging with median AP below 15%

[21]. This performance degradation in cluttered environments underscores the limitation of appearance-based methods when targets blend with textured backgrounds.

Approaches to improve small target detection

Super-resolution preprocessing offers one solution to address resolution limitations. Deep SR models applied prior to detection can enlarge images by a factor of 2, improving recall by up to 32.4% when trained end-to-end with the detector [22]. This approach effectively extends detection range. However, it requires additional processing steps and may introduce computational overhead that may conflict with real-time requirements.

Spatiotemporal fusion provides an alternative strategy. Extending detector input to continuous image sequences with inter-frame optical flow enables extraction of motion features from small, weak targets [23]. This approach achieves 86.87% average precision, representing an 11.49% improvement over single-frame baselines while maintaining speeds above 30 FPS. The key insight is that temporal integration compensates for spatial bandwidth limitations, a principle our method extends to 3D Gaussian representations.

Track-Before-Detect for weak signal conditions

TBD approaches process unthresholded measurements over multiple frames before detection decisions, enabling tracking of targets below conventional SNR thresholds. Dynamic Programming TBD (DP-TBD) achieves detection probability exceeding 90% at SCR = 1.5 with false alarm rates below 0.01% [24]. Multi-frame integration over 5–20 frames enables reliable detection when single-frame SCR falls below 3 dB. The computational complexity of $O(N \times V \times K)$ for N state cells, V velocity hypotheses, and K frames limits real-time feasibility without parallel implementations.

Particle Filter TBD handles nonlinear motion through Sequential Monte Carlo methods [25]. Generalized Labeled Multi-Bernoulli (GLMB) filters provide a theoretically optimal framework for multi-target tracking under weak signal conditions, with recent Belief Propagation variants achieving linear complexity [26]. Our volumetric voting approach draws from TBD principles but constrains the search space using multi-view geometry, reducing computational burden while maintaining sensitivity to weak signals.

The multi-view 3D constraint gap

All mainstream DBT architectures operate purely on single-view 2D images without native support for multi-view consistency, epipolar constraints, or 3D geometric priors. YOLO variants, Faster R-CNN, and DETR were designed and benchmarked exclusively on monocular imagery.

Flight dynamics-based methods demonstrate the value of 3D constraints for aerial targets. Bundle adjustment with flight dynamics priors enables robust 3D trajectory reconstruction from multiple ground cameras without requiring perfect single-view tracking or appearance matching across views [27]. Ad-hoc camera networks with unsynchronized, uncalibrated consumer cameras can achieve centimeter-accurate trajectory reconstruction when combined with rolling shutter correction [28].

Multi-view datasets with synchronized cameras and motion capture ground truth have recently emerged to support 3D drone tracking research. The DPJAIT dataset provides synchronized multi-camera video with Vicon-based 3D position ground truth in both real and simulated variants [29]. Related work demonstrates YOLOv5 detection on synchronized multi-camera systems with motion capture reference, proposing centroid distance metrics specifically for 3D tracking evaluation [30]. MMAUD provides multi-modal data including stereo cameras, LiDAR, radar, and audio arrays for anti-UAV research [31].

The CVPR 2024 UG2+ Challenge winner demonstrated feasibility of multi-modal 3D tracking using stereo vision, LiDAR, radar, and audio, achieving first place through dynamic points analysis and trajectory completion [32].

Conclusion: paradigm selection depends on operational requirements

DBT approaches deliver optimal performance for real-time detection of resolved targets (> 32 pixels) at reasonable SNR (> 5 dB), but performance degrades significantly with background complexity. Super-resolution and spatiotemporal fusion can extend detection range but add computational cost. TBD approaches become essential when targets fall below DBT thresholds, typically SCR < 3 dB (motion-blurred) or size < 20 pixels.

Current paradigms largely neglect the geometric constraints offered by multi-view configurations, operating exclusively on monocular imagery. Our approach bridges this gap by combining the temporal integration principles of TBD with explicit 3D reconstruction, using multi-camera geometry to resolve weak signals that defeat single-view methods.

4.1.2 Foreground Segmentation

Detecting flying objects against sky backgrounds requires robust foreground extraction. Statistical background subtraction methods, including GMM [33], [34] and ViBe [35], model pixel intensity distributions to identify anomalies. While computationally efficient for static scenes, these approaches fail when cloud motion or illumination changes violate stationarity assumptions, generating false positives during weather shifts. Motion estimation offers an alternative, with classical optical flow [36], [37] and deep learning variants like RAFT [38] and SEA-RAFT [39] tracking pixel displacement. However, these methods struggle in textureless sky regions where correspondence is ambiguous, and deep methods incur high computational costs unsuitable for low-latency surveillance.

Foundation models for segmentation, such as SAM [40] and SAM 2 [41], achieve strong generalization but lack autonomy. These promptable (user interaction for selecting foreground) architectures require prior localization to generate masks, creating a circular dependency for detection tasks. Moreover, they prioritize salient object boundaries over the fine-grained details of small aerial targets, such as propellers or landing gear, which are essential for classification.

Gap. Current 2D segmentation methods operate independently per view, propagating errors that downstream 3D reasoning cannot correct. Missed detections or false positives in individual frames corrupt the final output. We address this by formulating foreground detection directly in 3D space, using a ray-marching voting scheme that accumulates multi-view evidence to resolve ambiguities before making detection decisions.

4.1.3 Multi-View Fusion Strategies

The architectural decision of *when* to fuse information (before or after detection) defines the fundamental trade-off between geometric consistency and computational efficiency.

Early Fusion: 3D-First Representations. Early fusion methods project 2D features into shared 3D representations, typically Bird's Eye View (BEV), to ensure geometric consistency by construction. Lift-Splat-Shoot (LSS) [42] pioneered explicit depth-based lifting, while BEVFormer [43] and BEVFusion [44] employ attention mechanisms to implicitly reason about 3D geometry from multi-view queries. Although recent optimizations like RCBEVDet [45] achieve real-time performance, the memory overhead of dense voxel grids remains prohibitive for large surveillance volumes (> 500m), and their reliance on ground-plane assumptions proves ill-suited for the aerial domain.

Late Fusion: 2D-First Detection. Late fusion approaches leverage mature 2D detectors to independently identify objects before aggregating them via triangulation. While this paradigm scales linearly with camera count and exploits abundant 2D training data, it suffers from irreversible error accumulation. Missed detections in one view cannot be recovered, and false positives triangulate to spurious ghosts. This brittleness is particularly fatal for small, distant flying objects that frequently fall below single-view detection thresholds against textureless skies.

Volumetric Approaches. Volumetric methods like space carving [46] and ray-marching voting avoid explicit feature matching by accumulating occupancy evidence directly in 3D. These approaches naturally handle textureless regions where photometric correspondence fails, but they lack the learned discriminative power of deep features and scale poorly with resolution.

Gap. A critical gap exists for flying object surveillance: BEV methods depend on ground planes and texture; late fusion fails on weak signals; and volumetric methods lack semantic representations.

4.1.4 Multi-Object Tracking (MOT)

Multi-object tracking (MOT) extends single-frame detection to maintain consistent identities across time, evolving from purely geometric models to appearance-enhanced association. The dominant tracking-by-detection paradigm, epitomized by SORT [47], combines Kalman filtering with bipartite matching but fails under occlusion. DeepSORT [48] addresses this by fusing motion with learned appearance embeddings, while ByteTrack [49] improves association by recovering low-confidence detections. Recent refinements like OC-SORT [50] mitigate error accumulation during occlusion through observation-centric momentum, demonstrating that algorithmic improvements to classical components remain competitive.

Transitioning to 3D, MVTrajecter [51] lifts tracking to world coordinates by integrating multi-view evidence on datasets like Wildtrack [52]. End-to-end transformer architectures like TrackFormer [53] (extending DETR [54]) and MOTRv2 [55] treat tracking as set prediction, though at higher computational cost. In autonomous driving, methods like AB3DMOT [56] and CenterPoint [57] adapt these paradigms to LiDAR data, while PF-Track [58] and MUTR3D [59] explore transformer-based 3D tracking from dense sensor inputs.

Gap. existing 3D MOT methods rely on dense sensor coverage or outward facing cameras. Tracking small, distant targets against different backgrounds from sparse, upward-facing cameras precludes reliable appearance matching and standard 3D association.

4.2 3D Reconstruction Methods

The following section reviews existing methods for 3D reconstruction from multiple views with a focus on feed-forward architectures that can be used in real-time.

4.2.1 Classical Multi-View Reconstruction

Classical methods establish the geometric foundations of 3D reconstruction but struggle with the specific constraints of flying object surveillance. Pipelines like Structure from Motion (SfM) and Multi-View Stereo (MVS) (see Section 3.1) rely on the extraction and matching of local features (e.g., SIFT, ORB) to estimate poses and dense geometry. In textureless sky regions, these descriptors degenerate due to the lack of unique corners or edges, causing SfM pipelines to fail initialization and resulting in empty reconstructions. Volumetric approaches such as space carving [46] extend visual hulls (Section 3.1.4) to reason about occupancy without explicit correspondence, though they struggle with concavities and require precise segmentation. Despite their rigor, these approaches face critical limitations in our context: the iterative optimization required for bundle adjustment exceeds our 500ms latency target. Moreover, classical methods degrade significantly under sparse-view conditions ($N = 5$) and cannot amortize computation across scenes, motivating the development of the data-driven neural priors explored in this thesis.

4.2.2 Neural Radiance Fields (NeRF)

While Neural Radiance Fields (see Section **??**) revolutionized novel view synthesis, their application to flying object surveillance is hindered by prohibitive optimization times and sensitivity to sparse views. Generalizable variants like PixelNeRF [60] and MVSNeRF [61] attempted to eliminate per-scene optimization by conditioning on image features, while Instant-NGP [62] achieved real-time training via hash encoding. However, these methods remain fundamentally limited in our scenario:

- **Sparse-view degradation:** NeRFs typically require dense camera views (often > 50) to resolve geometry. In our sparse setup ($N \approx 5$), they suffer from overfitting [63].
- **Texture dependence:** Photometric losses fail in textureless sky regions, leading to "floater" artifacts [64] where masses of density are hallucinated to satisfy training views without representing real geometry.
- **Latency constraints:** The inference speed of volumetric ray-marching (seconds per frame) violates our sub-500ms latency requirement.
- **Small object sampling:** Stratified sampling misses fine details of distant objects (< 20 pixels).

These limitations necessitate the explicit, point-based representations offered by 3D Gaussian Splatting.

4.2.3 3D Gaussian Splatting Revolution

The emergence of 3D Gaussian Splatting (3DGS) [65] represents a paradigm shift in neural scene representation, optimizing anisotropic primitives for real-time rendering. While achieving speedups of 20× over NeRF, the requirement for per-scene optimization precludes direct application to novel scenes without retraining.

Feed-Forward Generalizable Architectures

To address this bottleneck, feed-forward architectures predict Gaussian parameters directly from input images. We categorize these methods based on their geometric priors and alignment with surveillance constraints (see Appendix A for a detailed taxonomy).

Pixel-Aligned and Probabilistic Methods. Pixel-aligned approaches like pixelSplat [66] and Splatter Image [67] predict primitives via image-to-image translation or probabilistic depth estimation. While pixelSplat handles depth ambiguity through discrete probability distributions, it produces floating artifacts in textureless regions where all depth hypotheses appear equally valid. Single-view methods like SAM 3D [68] rely heavily on learned priors, risking geometric hallucination that corrupts downstream classification.

Cost-Volume and Dual-Branch Architectures. Cost-volume methods such as MVSplat [69] and GPS-Gaussian [70] enforce geometric consistency through multi-view plane sweeping. MVSplat achieves efficient reconstruction (22 FPS) but degrades against uniform sky backgrounds where photometric variance vanishes. DepthSplat [71] resolves this by fusing multi-view cost volumes with monocular depth priors (Depth Anything V2). This dual-branch design provides a critical fallback: the monocular branch maintains structure when stereo matching fails in textureless regions, while the multi-view branch refines geometry where correspondence exists. Recent recurrent approaches like ReSplat achieve state-of-the-art fidelity through iterative error feedback, though they currently lack open implementations.

Transformer-Based Large Models. Large Reconstruction Models (LRMs) like GS-LRM [72] process tokenized images through massive transformer blocks. While powerful, these models are trained on object-centric datasets (Objaverse) with close-range targets filling the frame. This distribution mismatch renders them ineffective for aerial surveillance, where distant flying objects appear with limited spatial resolution, often falling below the effective capacity of standard patch tokenization.

Textureless and Sparse-View Limitations. Modern architectures exhibit fundamental limitations when applied to isolated objects against featureless backgrounds. Methods relying on explicit geometric initialization, such as GaussianPro [73] and ProSplat [74], fail catastrophically in sky regions where SfM cannot produce initialization points. Similarly, pure cost-volume methods suffer from depth ambiguity without texture gradients. Diffusion-based reconstruction methods are explicitly rejected for our pipeline; while visually plausible, they hallucinate unobserved geometry, introducing features that do not exist in the input. For classification tasks, this fabrication corrupts the feature space; we require architectures like DepthSplat that represent only observable geometry.

Conclusion. The surveyed methods systematically fail under the specific conditions of flying object classification: textureless backgrounds, wide baselines, and small distant targets. We address this by adopting DepthSplat's dual-branch architecture, which combines the robustness of monocular priors with the geometric fidelity of multi-view constraints. As detailed in Appendix A, we further augment this with silhouette consistency loss to anchor geometry in textureless regions.

4.2.4 Dynamic and 4D Gaussian Splatting

Dynamic Gaussian splatting extends the 3DGS paradigm to model temporal evolution through two fundamental approaches. Deformation based methods such as 4D Gaussian Splatting [75] decompose spacetime into HexPlane feature planes, where a lightweight MLP predicts per Gaussian deformations $(\Delta\mu, \Delta q, \Delta s)$ conditioned on time, achieving 82 FPS at 800×800 resolution. Native 4D methods take a different approach: 4D Rotor Gaussian Splatting [76] represents primitives as anisotropic XYZT Gaussians using geometric algebra rotors, obtaining 3D Gaussians through temporal slicing at each timestamp and reaching 277 FPS. Subsequent work addresses efficiency: Hybrid 3D-4D Gaussian Splatting [77] converts temporally invariant Gaussians from 4D to 3D during training based on temporal scale thresholds, while 4DGS-1K [78] eliminates short lifespan Gaussians and skips inactive primitives during rasterization to achieve 1000+ FPS with 41× storage reduction. Despite these advances, all methods share two fundamental characteristics: they require iterative per scene optimization rather than feed forward inference, and they target novel view synthesis quality exclusively.

[**TODO:** Discuss Feed-Forward 4DGS, but that all methods use feed forward per X frames and then interpolate in some way]

Gap. No existing 4D Gaussian method extracts temporal motion patterns as discriminative features for downstream tasks. The deformation fields and rotor parameters encoding Gaussian translation and rotation remain untapped signal sources. For flying object discrimination, where drones exhibit rigid body motion and birds display periodic wing deformation, these temporal dynamics provide powerful classification cues beyond static appearance. This thesis addresses this gap by treating 4D Gaussian dynamics not as a rendering mechanism but as a feature space for classification.

4.2.5 Section Summary and Positioning

The works of 3D reconstruction going from classical Structure from Motion to feed-forward Gaussian Splatting demonstrates a clear trend toward improved fidelity and efficiency. Despite these advances, a significant division persists. Traditional methods provide precise geometry but struggle in scenes that lack sufficient texture or scale. Conversely, neural approaches remain robust in textureless areas through learned priors yet often generate artificial geometry when cameras are widely separated. Effective aerial surveillance demands a hybrid strategy. This approach must leverage the geometric constraints of multiple cameras to locate small objects while utilizing Gaussian Splatting to model their shape and appearance efficiently. Our thesis builds upon DepthSplat and adapts it to be retrained for reconstruction centered on objects from multiple views.

4.3 Classification from 3D Representations

4.3.1 Point Cloud Processing Architectures

Processing unordered 3D point sets presents a fundamental challenge for neural networks due to the lack of regular grid structure. Since 3D Gaussian representations share this unstructured nature, point cloud methods provide relevant architectural foundations. PointNet [79] pioneered direct processing of raw point clouds by combining per point MLPs with symmetric aggregation functions. Building on this, PointNet++ [80] introduced hierarchical processing to capture local geometric patterns through recursive abstraction. Point Transformer [81] and its serialized successor V3 [82] further advanced the field by applying self attention mechanisms to model complex dependencies.

Classification on Gaussian Representations. ShapeSplat [83] adapts these principles to 3D Gaussian splatting by jointly processing geometric and appearance properties. Their findings confirm that effective classification requires the full Gaussian parameters rather than reduced centroid proxies. However, a critical gap remains as these architectures operate on static snapshots and ignore temporal dynamics. For flying objects where motion signatures like wing flapping are discriminative, purely static processing yields suboptimal results. This limitation necessitates architectures capable of jointly reasoning over Gaussian geometry, appearance, and temporal evolution.

4.3.2 Set Processing and Gaussian Tokenization

Gaussian representations inherently constitute unordered sets of variable cardinality, requiring permutation-invariant architectures for classification. DeepSets [18] established that permutation invariance can be achieved via sum-pooling individual element encodings, though this aggregation discards critical pairwise relationships. To capture these interactions, the Set Transformer [84] introduced self-attention mechanisms, utilizing inducing points to reduce quadratic complexity to $O(nm)$ for tractability. The Perceiver [85] further refined this by mapping variable-size inputs to fixed-size latent arrays via iterative cross-attention, enabling modality-agnostic processing.

Gap. However, no established paradigm exists for Gaussian-based classification. Existing set architectures, developed for point clouds or multimodal inputs, fail to exploit the unique geometric structure and temporal evolution of Gaussian primitives.

4.3.3 Temporal Sequence Modeling

State Space Models (SSMs) have emerged as the leading architecture for 3D and 4D point cloud sequence modeling, achieving up to 92.6% accuracy on challenging benchmarks while offering linear complexity $O(N)$. This contrasts sharply with the quadratic complexity $O(N^2)$ of transformers, which limits their applicability in edge computing scenarios. The evolution of sequence modeling reveals a clear trajectory toward architectures that balance global context with computational efficiency. Classical Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks suffer from sequential bottlenecks and gradient instability, limiting their effective context window. While Temporal Convolutional Networks (TCNs) introduced parallelization, they remain constrained by fixed receptive fields. The transformer revolution, exemplified by TimeSformer [86], achieved high accuracy through factorized space-time attention. However, its quadratic complexity creates prohibitive memory pressure.

The Mamba architecture [19] fundamentally reconceptualizes this landscape by introducing selective state space models with linear complexity. By making parameters input-dependent, Mamba enables content-based reasoning while maintaining the efficiency of global convolutions. Its hardware-aware selective scan algorithm fuses operations in GPU SRAM, reducing memory I/O by $O(N)$ and achieving inference throughput 5 times higher than equivalent transformers. This architecture extrapolates perfectly to sequences exceeding 1 million tokens, a capability unmatched by attention-based models.

Adapting this to 3D data, PointMamba [87] utilizes Hilbert space-filling curves to serialize unordered point clouds, outperforming transformers with significantly fewer parameters. Addressing the lack of local features, Mamba3D [88] introduces Bidirectional SSMs and Local Norm Pooling, achieving 92.6% accuracy on ScanObjectNN. For 4D video understanding, Mamba4D [89] disentangles spatial and temporal modeling, demonstrating a 10.4% accuracy improvement over P4Transformer [90] on long sequences while reducing GPU memory usage by 87.5%. This linear scaling and constant memory requirement $O(D \cdot N)$ make SSMs the optimal choice for real-time edge deployment in aerial surveillance where longer frame sequences can be used to capture temporal dynamics.

4.3.4 SE(3) Equivariance for 3D Data

SE(3) equivariant neural networks ensure that outputs transform predictably under 3D rigid body transformations, eliminating the need for extensive data augmentation. While standard augmentation attempts to cover the continuous SO(3) manifold through random sampling, it provides no theoretical guarantees and often fails to generalize to unseen orientations. Equivariant architectures solve this by baking geometric symmetry directly into the network structure, sharing weights across all possible rotations.

The Trade-off: Expressivity vs. Efficiency

The landscape of equivariant methods is defined by a fundamental trade-off between computational cost and representational power. Spherical harmonic methods, including Tensor Field Networks (TFN) [91] and SE(3) Transformers [92], achieve high expressivity by constraining filters to products of radial functions and spherical harmonics. Equiformer [93] advances this further with nonlinear message passing, achieving state of the art accuracy on molecular dynamics tasks. However, these gains come at a prohibitive cost; the required tensor products scale as $O(L^6)$ or $O(L^3)$, making them 10 to 100 times slower than scalar networks and unsuitable for edge deployment.

Conversely, lightweight architectures prioritize inference speed by avoiding complex basis functions. Vector Neurons [94] extend standard operations to 3D vector spaces, enabling SO(3) equivariance with only a small constant overhead over PointNet. Building on this foundation, VN-Transformer [95] introduces rotation equivariant attention through Frobenius inner products between vector representations, achieving 90.8% accuracy on ModelNet40 **[TODO:** modelnet40]. For efficient processing of large point clouds, VN-Transformer employs VN-MeanProject, a permutation invariant downsampling operation that reduces self-attention complexity from $O(N^2C)$ to $O(M^2C')$ where $M \ll N$. Crucially, VN-Transformer supports early fusion of non-spatial attributes (such as opacity and color) by extending features from $C \times 3$ to $C \times (3 + d_A)$ matrices, directly applicable to the heterogeneous attributes of 3D Gaussians. This combination of parameter efficiency, scalable attention, and attribute fusion makes VN-Transformer the most viable candidate for downstream tasks such as real-time Gaussian classification.

Gap in Aerial Object Detection

Despite the inherent 3D nature of flying objects, SE(3) equivariant methods remain virtually unexplored in the domain of drone and bird detection. Current systems rely on standard CNNs or Transformers that struggle to generalize across arbitrary viewing angles or backgrounds without massive datasets. This represents a critical gap; applying equivariant architectures could enable significant robustness for classifying aerial targets that are randomly oriented relative to ground-based sensors.

4.3.5 Motion Signatures for Drone-Bird Discrimination

Radar micro-Doppler research has established that motion signatures provide highly discriminative features for drone-bird classification. The fundamental discriminator is the frequency difference between drone rotor rotation (>50 Hz) and bird wing flapping (4–6 Hz) [96]. Molchanov et al. [97] achieved 92% accuracy using eigenpair features extracted from micro-Doppler signatures, while Rahman and Robertson [98] reported 99% accuracy using CNNs on spectrogram images.

Despite radar's success, vision-based methods remain dominated by appearance-based deep learning with limited exploitation of temporal motion cues. Akyon et al. [99] demonstrated that adding LSTM/Transformer temporal fusion to video features improves bird classification F1-score by 73%, confirming that motion information provides substantial discriminative value even from RGB imagery. Similarly, Sun et al. [23] utilized spatiotemporal information and optical flow to enhance UAV detection in surveillance videos.

Gap and Opportunity. While radar can directly measure Doppler frequencies, standard video frame rates (30–60 fps) cannot capture high-frequency rotor motion but *can* resolve wing flapping frequencies (4–10 Hz). No existing work extracts discriminative motion signatures from 3D representations. We aim to extract discriminative motion signatures (frequency, periodicity) from dynamic gaussians to address the limitations of appearance-only methods, exploiting the theoretical feasibility of capturing wing flapping frequencies (4-10 Hz) at standard frame rates.

4.4 Chapter Summary and Research Gaps

This chapter has reviewed the three technical pillars underpinning our research: flying object detection, 3D reconstruction, and temporal classification. Across these domains, we observe a recurring pattern: methods are optimized for either close-range, texture-rich environments (automotive, indoor) or single-view 2D tasks. The specific intersection of constraints in our problem setting (sparse wide-baseline views, textureless sky backgrounds, and small dynamic targets) exposes fundamental gaps in the state of the art.

Table 4.1 summarizes these identified gaps and maps them to the specific methodological contributions of this thesis.

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

Identified Research Gap	Thesis Contribution
Detection: TBD paradigms accumulate evidence but lack multi-view 3D awareness; DBT methods fail on low-SNR targets.	A multi-view ray-marching voting mechanism that accumulates foreground evidence in 3D space before detection.
Reconstruction: Feed-forward Gaussian Splatting assumes textured scenes and bounded objects; fails on sky backgrounds and wide baselines.	An adapted feed-forward architecture that leverages cost-volume geometry to reconstruct isolated targets against featureless backgrounds.
Classification: No existing paradigm for classifying 4D Gaussian sequences; point cloud methods ignore appearance/opacity.	A novel 4D Gaussian tokenization scheme combined with SE(3)-equivariant processing.
Dynamics: Temporal evolution of Gaussian parameters (deformation, rotation) is unexploited for discrimination.	A temporal State Space Model (Mamba) that extracts motion signatures from Gaussian trajectories.
Data: Lack of synchronized multi-view datasets with 3D ground truth for aerial targets.	A synthetic dataset generation pipeline using NVIDIA Isaac Sim to bridge the data gap.

The following chapter presents our methodology, detailing how we synthesize these contributions into an end-to-end pipeline that addresses each identified limitation.

5 Methodology

5.1 Introduction: The Optical Imperative for Aerial Surveillance

The widespread availability of small Unmanned Aerial Vehicles creates challenges for airspace security. Inexpensive commercial drones, often smaller than 50 centimeters, can disrupt infrastructure and threaten privacy. Conventional radar systems are designed to detect metallic aircraft and have challenges distinguishing plastic airframes from birds. Directed microphones suffer from environmental noise and rely on the acoustic signature of active propulsion. Radio frequency analysis fails when signals are jammed or when drones operate via optical fiber cables. Furthermore, high frequency systems are too expensive for widespread perimeter monitoring.

These factors guided the development of a passive optical sensing system. However, the optical domain presents it's own challenges. Distant drones and birds appear as indistinct shapes against the sky, rendering classification based on individual images unreliable. Moreover, edge deployment requires systems that operate with low latency on hardware with limited power.

This chapter describes a methodology for a passive 3D occupancy prediction, classification, and tracking system using multiple cameras. By combining principles from multiview geometry, neural rendering via 3D Gaussian Splatting, and temporal sequence modeling, we propose an architecture that addresses the limitations of detection based on appearance. The approach includes four main components: synthetic data generation, volumetric segmentation, feedforward 3D reconstruction, and temporal classification robust to rotation.

5.2 The Physics of Sensing and the Detection Implications

To engineer a robust solution, one must first quantify the information-theoretic limits of the sensing channel. The challenge in long-range optical surveillance is the degradation of spatial bandwidth as distance increases.

5.2.1 The Spatial Bandwidth Limit

According to the IEC 62676-4 standard, reliable object recognition requires a pixel density of 125 Pixels Per Meter (PPM). In a typical perimeter monitoring scenario with a target at distance $d = 50$ meters, utilizing a standard 2K sensor (1920×1080 pixels) with a 90-degree horizontal field of view, the pixel density drops to approximately 19.2 PPM. At this resolution, a drone with a 50cm wingspan will be represented by 9.6 pixels. The Nyquist-Shannon sampling theorem dictates that high-frequency spatial features (propellers) will be lost due to aliasing. Consequently, the signal entering the classification pipeline is not a structured image of an object, but a low-frequency approximation contaminated by sensor noise and atmospheric point spread functions.

This physical constraint invalidates the concept of deep learning architectures that rely on texture-rich feature extraction. It necessitates an engineering pivot from spatial feature extraction to temporal and geometric feature extraction. While the spatial channel capacity is exceeded, the temporal channel capacity remains available: if a target is tracked over T frames and exhibits characteristic motion, the integration of information over time can recover the discriminative capacity lost in spatial downsampling.

5.2.2 The Signal-to-Noise Ratio and the Failure of Detect-Before-Track

The dominant paradigm in computer vision is Detect-Before-Track (DBT), exemplified by architectures such as YOLO [20], Faster R-CNN [100], and DETR [54]. These systems operate on a decision threshold applied to single frames: if the confidence score of a region exceeds a threshold τ, it is declared a detection; otherwise, it is discarded.

Analysis of the Drone-vs-Bird [21] and Anti-UAV benchmarks reveals the failure of DBT in low signal-to-noise ratio scenarios. When a target occupies fewer than 16×16 pixels, the deep feature downsampling in Convolutional Neural Networks reduces the target representation to a single feature vector, often indistinguishable from background noise. Standard anchor boxes, designed for larger objects like pedestrians or vehicles, fail to generate high Intersection-over-Union proposals for these small targets. Consequently, widely deployed models see their mean Average Precision (MAP) plummet from above 0.95 on large drones to below 0.65 on bird-sized objects at range.

Furthermore, DBT systems are incapable of detecting targets with a Signal-to-Clutter Ratio below approximately 3 to 5 dB. In scenarios involving cloud clutter, haze, or low light, the target signal may be buried within the noise floor. A threshold high enough to suppress false alarms will inevitably suppress the target; a threshold low enough to detect the target will trigger the tracker with false positives, leading to a computational explosion in the data association phase.

5.3 Synthetic Data Generation

The absence of large-scale, annotated, multi-view datasets for drones and birds is a challenge for the development of robust classification systems. We address this limitation through a synthetic data pipeline utilizing NVIDIA Isaac Sim [101].

5.3.1 Dataset Construction

Isaac Sim provides a physically-based simulation environment capable of generating training data that accurately models the optical complexities of real-world aerial surveillance. The synthetic data pipeline comprises the following elements:

1. **Asset Diversity.** We import openly available 3D models of various drones and birds with flight animations. These models capture the geometric and kinematic differences between mechanical and biological flyers that form the discriminative basis for classification.
2. **Environmental Simulation.** The simulation incorporates realistic atmospheric effects including volumetric fog and haze to model contrast reduction at extended ranges, variable lighting conditions from dawn to dusk, and dynamic cloud formations that create the textureless backgrounds characteristic of aerial surveillance scenarios.
3. **Sensor Modeling.** To prevent the network from overfitting to idealized simulations, we apply sensor noise modeling including Gaussian read noise and Poisson shot noise to mimic low-light sensor grain, motion blur from object movement and camera vibration, and rolling shutter artifacts characteristic of CMOS sensors.
4. **Calibration.** We slightly change camera intrinsic and extrinsic parameters to train the network to be robust to calibration drift, a common issue in pole-mounted outdoor cameras exposed to thermal cycling and mechanical vibration.

5.3.2 Ground Truth Generation

The synthetic environment provides ground truth annotations that would be impossible to obtain from real-world data: per-pixel depth maps enabling geometric supervision, instance segmentation masks for each flying object, 3D trajectories with millimeter-level accuracy, and per-frame 3D Gaussian Splatting parameters that serve as reconstruction targets.

5.4 Multi-View Volumetric Track-Before-Detect

To address the limitations of Detect-Before-Track, we engineer a Track-Before-Detect (TBD) pipeline that processes simple pixel differences from multiple frames before making a detection decision, enabling the detection of targets with signal levels too low for conventional methods.

5.4.1 Theoretical Foundation

Classical TBD methods such as Dynamic Programming TBD or Particle Filter TBD are effective but suffer from computational complexity of $O(N \times V \times K)$ for state space N, velocity discretization V, and temporal window K. We address this by using multi-view consistency from our calibrated camera network to constrain the TBD search space.

5.4.2 Volumetric Voting Mechanism

The algorithm proceeds through the following stages:

[**TODO:** make algorithm as in papers]

1. **Monitored Volume Definition.** We define a volume $\mathcal{V} \subset \mathbb{R}^3$ monitored by $N \geq 5$ calibrated cameras with known intrinsic matrices $\mathbf{K}_n$ and extrinsic transformations $[\mathbf{R}_n|\mathbf{t}_n]$. The volume is discretized into a voxel grid with resolution determined by the minimum resolvable depth difference across the camera network.
2. **Ray Marching.** For every pixel $\mathbf{p}$ in every camera n that exceeds a minimal noise floor (significantly lower than a detection threshold), we cast a ray $\mathbf{r}_{n,\mathbf{p}}$ into $\mathcal{V}$. The ray origin is the camera center $\mathbf{c}_n = -\mathbf{R}_n^T\mathbf{t}_n$ and the direction is $\mathbf{d}_{n,\mathbf{p}} = \mathbf{R}_n^T\mathbf{K}_n^{-1}\tilde{\mathbf{p}}$ where $\tilde{\mathbf{p}}$ is the homogeneous pixel coordinate.
3. **Voxel Accumulation.** As rays traverse the volume, they increment the energy state of intersected voxels. The energy contribution is weighted by the pixel intensity above the noise floor, providing a measure of evidence rather than a binary vote.
4. **Geometric Consistency Filter.** A voxel is considered a candidate if and only if it receives energy from at least 3 distinct camera views. This multi-view constraint uses the epipolar geometry of the camera network to filter out single-view noise and phantom artifacts that do not correspond to dynamically moving 3D objects.
5. **Temporal Integration.** The energy of candidate voxels is integrated over a sliding temporal window of length T. A detection is declared only when the accumulated 3D energy density exceeds a threshold, signifying a temporally consistent physical object. This temporal integration boosts the signal-to-noise ratio by a factor of $\sqrt{T}$ while the N-view constraint eliminates the false positives that fail 2D TBD methods.
6. **Visual Hull.** To enhance efficiency, we optionally can sparsely traverse pixel differences, limiting energy accumulation to voxels within the visual hull of the moving object.

5.4.3 Sparse Implementation

[**TODO:** Implement the volumetric representation using sparse tensor structures via the Minkowski Engine / octree?]

5.5 Feed-Forward 3D Gaussian Reconstruction

Once a target is localized via the TBD pipeline, the system must reconstruct its 3D form to facilitate classification. This presents a challenge: the background is predominantly open sky, providing no texture for classical photometric matching.

5.5.1 Architecture Selection: DepthSplat

Given the limitations of classical photogrammetry and Neural Radiance Fields in sparse-view, textureless environments, we select the DepthSplat architecture [71] (see Appendix A for a detailed justification), a feed-forward 3D Gaussian Splatting model. Unlike standard 3DGS [13] which requires per-scene optimization through thousands of gradient descent iterations, DepthSplat predicts Gaussian parameters directly from input images in a single inference pass.

Dual-Branch Design. DepthSplat addresses the textureless sky challenge through a dual-branch architecture that combines complementary depth estimation strategies:

1. **Multi-View Cost Volume Branch:** This branch constructs a plane-sweep cost volume that enforces geometric consistency across views. For each depth hypothesis, features are warped from source views to

the reference view and compared via learned similarity metrics. This provides strong constraints where texture exists but fails in uniform regions.

2. **Monocular Depth Branch:** Based on Depth Anything V2 [102], this branch provides strong depth priors learned from large-scale training data. It predicts plausible depth even in textureless regions where photometric matching fails, preventing the floating artifacts seen in pure cost-volume methods such as MVSplat [69].

The fusion of these branches ensures robustness: the cost volume prevents depth hallucination common in purely generative methods, while the monocular prior prevents artifacts when photometric gradients vanish against the sky.

5.5.2 Domain Adaptation for Aerial Objects

The pre-trained DepthSplat model is adapted for aerial object reconstruction through a two-stage training procedure:

1. **Object-Centric Pre-training.** Following Lin et al. [103], we pre-train on a curated subset of the Objaverse dataset rather than the full collection. Models trained on quality-focused subsets achieve superior performance in object reconstruction tasks compared to those trained on larger, uncurated datasets containing noisy geometry.
2. **Domain Fine-tuning.** The model is subsequently fine-tuned on our synthetic drone and bird dataset, adapting it to the specific geometric characteristics of flying objects including thin structures (rotors, feathers), non-convex shapes, and the characteristic scale range of targets at 10-100 meter distances.
3. **Silhouette Consistency Loss.** To anchor geometry when photometric gradients vanish, we augment the training objective with a silhouette consistency loss that compares rendered alpha masks against segmentation masks from the TBD pipeline. This provides geometric supervision independent of texture.

5.5.3 3D Gaussian Parameterization

The reconstruction output is a set of 3D Gaussian primitives, each parameterized by:

- **Position** $\boldsymbol{\mu} \in \mathbb{R}^3$: The mean location in 3D space.
- **Covariance** $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times 3}$: Decomposed into scale $\mathbf{s} \in \mathbb{R}^3$ and rotation quaternion $\mathbf{q} \in \mathbb{R}^4$ as $\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$ where $\mathbf{R} = \mathbf{R}(\mathbf{q})$ and $\mathbf{S} = \mathrm{diag}(\mathbf{s})$.
- **Opacity** $\alpha \in [0, 1]$: The transparency of the primitive.
- **Spherical Harmonics** $\mathbf{c} \in \mathbb{R}^{48}$: View-dependent color encoded as coefficients up to degree 3 (16 coefficients per RGB channel).

These explicit primitives encode rich geometric and appearance information that forms the basis for subsequent classification.

5.6 Rotation-Robust 4D Classification Architecture

The reconstructed 3D representation is not a static artifact but a dynamic entity evolving over time. The discrimination between a drone and a bird relies on the temporal characteristics of this evolution: the periodic rotation of propellers versus the rhythmic flapping of wings, the rigid-body dynamics of mechanical flight versus the flexible deformation of biological locomotion.

5.6.1 The Challenge of Attribute Heterogeneity

A challenge in 3D Gaussian classification is ensuring consistency across arbitrary viewing orientations. A drone approaching from the north must be classified identically to one approaching from the east. Standard neural networks are not rotation invariant and must learn distinct filters for every possible orientation, requiring massive data augmentation and still failing to generalize to novel viewpoints.

The complexity is amplified by the heterogeneous transformation properties of Gaussian attributes under rotation $\mathbf{R} \in SO(3)$:

- **Positions** transform as vectors: $\boldsymbol{\mu}' = \mathbf{R}\boldsymbol{\mu}$
- **Covariances** transform as tensors: $\boldsymbol{\Sigma}' = \mathbf{R}\boldsymbol{\Sigma}\mathbf{R}^T$
- **Quaternions** transform via quaternion multiplication with the rotation

- **Spherical Harmonics** transform via degree-specific Wigner D-matrices
- **Opacity** remains invariant as a scalar quantity

This mathematical heterogeneity makes rotation-robust 4DGS classification more complex than standard point cloud classification, where all features are either positions or invariant scalars.

5.6.2 Two-Stage Architecture Overview

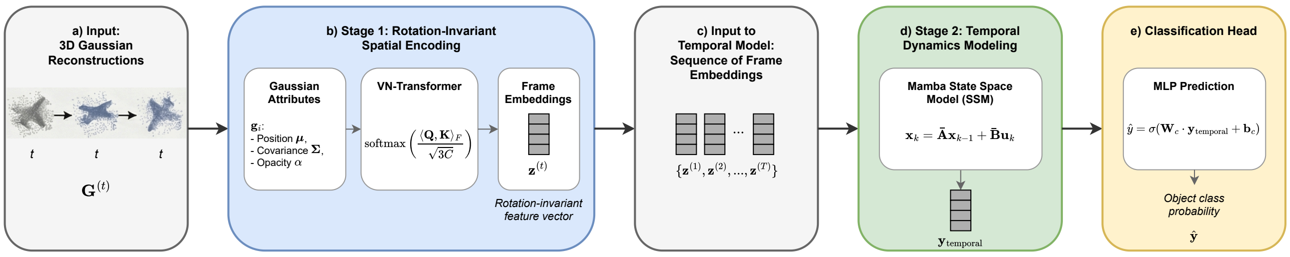
a) Input: 3D Gaussian Reconstructions
t
t
t
$\mathbf{G}^{(t)}$
b) Stage 1: Rotation-Invariant Spatial Encoding
Gaussian Attributes
$\mathbf{g}_i$:
- Position $\boldsymbol{\mu}$,
- Covariance $\boldsymbol{\Sigma}$,
- Opacity α
VN-Transformer
$\mathrm{softmax}\left(\frac{\langle \mathbf{Q}, \mathbf{K} \rangle_F}{\sqrt{3C}}\right)$
Frame Embeddings
$\mathbf{z}^{(t)}$
Rotation-invariant feature vector
c) Input to Temporal Model: Sequence of Frame Embeddings
$\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(T)}\}$
d) Stage 2: Temporal Dynamics Modeling
Mamba State Space Model (SSM)
$\mathbf{x}_k = \bar{\mathbf{A}}\mathbf{x}_{k-1} + \bar{\mathbf{B}}\mathbf{u}_k$
$\mathbf{y}_{\text{temporal}}$
e) Classification Head
MLP Prediction
$\hat{y} = \sigma(\mathbf{W}_c \cdot \mathbf{y}_{\text{temporal}} + \mathbf{b}_c)$
Object class probability
$\hat{\mathbf{y}}$

Figure 5.1: Overview of the 4D classification architecture. (a) The input consists of a sequence of 3D Gaussian reconstructions. (b) In Stage 1, a VN-Transformer encodes geometric features into a rotation-invariant representation. (c) This produces a sequence of frame embeddings. (d) In Stage 2, a Mamba State Space Model captures temporal dynamics. (e) Finally, an MLP classification head predicts the object class.

We adopt a two-stage architecture, illustrated in Figure 5.1, that separates rotation-invariant feature extraction from temporal processing. The process begins with (a) a sequence of 3D Gaussian reconstructions. (b) In Stage 1, a VN-Transformer extracts rotation-invariant spatial encodings, producing (c) a sequence of frame embeddings. These are processed by (d) Stage 2, a Mamba State Space Model (SSM) for temporal dynamics modeling, followed by (e) a classification head for final prediction. This design follows the principle that *invariance should be established early* in the pipeline [94], trading potential discriminative power from full equivariant processing for practical implementation simplicity and training stability.

$$\hat{y} = f_{\text{temporal}}\left(\left\{f_{\text{spatial}}(G^{(t)})\right\}_{t=1}^{T}\right) \tag{5.1}$$

where $G^{(t)} = \{g_i^{(t)}\}_{i=1}^{M_t}$ is the set of M_t Gaussians at frame t, f_{spatial} extracts rotation-invariant per-frame embeddings, and f_{temporal} models the temporal evolution for classification.

5.6.3 Stage 1: Rotation-Equivariant Spatial Encoding

The spatial encoder must process the heterogeneous Gaussian attributes while producing representations that are invariant to the object's 3D orientation. We employ the VN-Transformer architecture [95] which processes geometric features equivariantly, while handling non-geometric attributes through early fusion.

Gaussian Feature Preparation

Each Gaussian g_i provides attributes with different transformation properties under rotation:

- **Type-1 Features (Equivariant)**: Positions $\boldsymbol{\mu}_i \in \mathbb{R}^3$ and the principal axes of the covariance matrix $\boldsymbol{\Sigma}_i$ (obtained via eigendecomposition) are represented as 3D vectors that transform under rotation. These are processed directly by VN-Transformer's equivariant layers.
- **Scalar Features (Invariant)**: Opacity α_i is inherently invariant. Covariance eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ provide rotation-invariant shape descriptors. For spherical harmonics, implementing full Wigner D-matrix transformations adds substantial complexity; instead, we extract band energies $E_l = \sum_{m=-l}^{l} |c_l^m|^2$ for each degree $l \in \{0, 1, 2, 3\}$, yielding 4 invariant scalars per color channel (12 total for RGB) [104].

VN-Transformer Architecture

The Vector Neurons framework [94] lifts scalar neurons to 3D vectors, representing features as $\mathbf{V} \in \mathbb{R}^{N \times C \times 3}$ where each of C feature channels is a 3D vector that transforms coherently under rotation.

VN-Linear Layer. Given weight matrix $\mathbf{W} \in \mathbb{R}^{C' \times C}$ and input $\mathbf{V} \in \mathbb{R}^{C \times 3}$, the operation $\mathbf{V}' = \mathbf{W}\mathbf{V}$ applies weights to the channel dimension while leaving the spatial dimension untouched. Since rotation acts on spatial dimensions and linear combination acts on channels, these operations commute, guaranteeing equivariance.

VN-ReLU Layer. Element-wise nonlinearities break rotation equivariance; the solution learns data-dependent directions for activation. Computing $\mathbf{q} = \mathbf{W}_q\mathbf{V}$ and $\mathbf{k} = \mathbf{W}_k\mathbf{V}$, the output projects $\mathbf{q}$ onto the half-space defined by $\mathbf{k}$ when $\langle \mathbf{q}, \mathbf{k} \rangle < 0$. Both projections transform equivariantly, and their inner product remains invariant, preserving equivariance through the nonlinearity.

Frobenius Attention. VN-Transformer computes attention weights using Frobenius inner products between vector neuron representations:

$$\langle \mathbf{V}^{(i)}, \mathbf{V}^{(j)} \rangle_F = \sum_c \mathbf{V}^{(i,c)} \cdot (\mathbf{V}^{(j,c)})^T \tag{5.2}$$

This inner product is rotation-invariant since $\langle \mathbf{V}^{(i)}\mathbf{R}, \mathbf{V}^{(j)}\mathbf{R} \rangle_F = \langle \mathbf{V}^{(i)}, \mathbf{V}^{(j)} \rangle_F$ by orthogonality $\mathbf{R}\mathbf{R}^T = \mathbf{I}$. Attention weights $\mathrm{softmax}(\langle \mathbf{Q}, \mathbf{K} \rangle_F / \sqrt{3C})$ are therefore rotation-invariant, while value aggregation with scalar weights and equivariant values produces equivariant outputs.

Early Fusion of Scalar Attributes

Following VN-Transformer's design for non-spatial attributes, we extend features from $C \times 3$ to $C \times (3 + d_A)$ matrices, concatenating the 3D vector components with scalar attributes (opacity, eigenvalues, SH band energies) along the feature dimension. The network learns to process both geometric and non-geometric information jointly while maintaining equivariance for the spatial components.

Approximate Equivariance

We introduce ϵ-approximate equivariance through bias terms with $\|\text{bias}\| \leq \epsilon \approx 10^{-6}$. This significantly improves numerical stability on accelerator hardware while maintaining effective rotation invariance. Experiments on ModelNet40 show this controlled relaxation improves accuracy from 91.1% to 95.4% [95].

VN-Invariant Output

To produce rotation-invariant frame embeddings for temporal processing, we apply the VN-Invariant layer: learning a local coordinate frame $\mathbf{T} \in \mathbb{R}^{3 \times 3}$ from the features themselves, then expressing features in this frame via $\mathbf{V} \cdot \mathbf{T}^T$. Since both $\mathbf{V}$ and $\mathbf{T}$ transform under rotation, their product remains invariant.

Frame-Level Aggregation

Per-Gaussian embeddings $\mathbf{h}_i^{(t)}$ are aggregated into a frame-level representation $\mathbf{z}^{(t)}$ via attention-weighted pooling:

$$\mathbf{z}^{(t)} = \sum_{i=1}^{M_t} \alpha_i \mathbf{h}_i^{(t)}, \quad \alpha_i = \frac{\exp(w^T \mathbf{h}_i^{(t)})}{\sum_j \exp(w^T \mathbf{h}_j^{(t)})} \tag{5.3}$$

where w is a learned attention vector. This permutation-invariant aggregation handles the variable number of Gaussians M_t across frames.

5.6.4 Stage 2: Temporal Dynamics Modeling with Mamba4D

The sequence of frame embeddings $\{\mathbf{z}^{(t)}\}_{t=1}^{T}$ encodes the spatial configuration at each time step. The temporal classifier must extract discriminative motion patterns from this sequence to distinguish mechanical from biological flight.

The Computational Bottleneck of Transformers

The standard approach for sequence modeling is the Transformer architecture. However, Transformers suffer from quadratic complexity $O(T^2)$ with respect to sequence length due to the self-attention mechanism. For aerial surveillance, capturing the periodicity of a bird's wing beat (approximately 4–6 Hz) or a drone's maneuver requires a significant temporal window (e.g., $T = 60$ frames at 30 fps). At this length, the attention map becomes memory-prohibitive on edge devices.

Selective State Space Models

We engineer the temporal classifier using the Mamba4D architecture [89], a modern Selective State Space Model representing the first purely SSM-based backbone for 4D point cloud understanding. Mamba4D achieves three engineering advantages:

Linear Complexity. The discrete SSM equations

$$\mathbf{x}_k = \bar{\mathbf{A}}\mathbf{x}_{k-1} + \bar{\mathbf{B}}\mathbf{u}_k \tag{5.4}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k \tag{5.5}$$

provide $O(T)$ linear scaling versus Transformers' quadratic $O(T^2)$. Parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are computed as functions of the input (data-dependent/selective), enabling the model to focus on relevant temporal features.

Selective Forgetting. In a surveillance context, tracks are often noisy or interrupted by occlusion. Mamba's selective mechanism allows the network to effectively reset its hidden state or ignore noisy inputs dynamically, providing robustness that standard RNNs lack.

Hardware Efficiency. The selective scan algorithm is engineered to fuse operations into the GPU's fast SRAM, minimizing High Bandwidth Memory I/O.

Disentangled Spatial-Temporal Processing

Mamba4D disentangles spatial and temporal processing into complementary stages:

Intra-frame Spatial Mamba. This component encodes local geometric structures within short-term clips using spatial state space modeling over the per-frame Gaussian embeddings.

Inter-frame Temporal Mamba. This component captures long-range temporal dependencies across the entire video sequence. The architecture processes frame-level embeddings through stacked Mamba blocks:

$$\mathbf{F}'_l = \mathrm{DW}(\mathrm{Linear}(\mathbf{F}_{l-1})) \tag{5.6}$$

$$\mathbf{F}_l = \mathrm{Linear}(\mathrm{SSM}(\sigma(\mathbf{F}'_l)) \odot \sigma(\mathrm{Linear}(\mathbf{F}_{l-1}))) \tag{5.7}$$

where DW is depth-wise convolution, σ is SiLU activation, and SSM is the selective state space model.

No Correspondence Required. Critically, Mamba4D processes dynamic sequences without requiring explicit point correspondence between frames. This matches our pipeline where feed-forward reconstruction generates a new, unstructured set of Gaussians at each frame. The model learns temporal patterns from the sequence of rotation-invariant spatial configurations, effectively bypassing the need for explicit feature tracking.

Scaling Behavior

Unlike Transformers whose accuracy degrades with sequence length due to attention dilution, Mamba4D accuracy improves with longer sequences: from 92.68% at 24 frames to 93.23% at 36 frames on action recognition benchmarks [89]. This property is essential for capturing the extended temporal patterns that distinguish drone flight dynamics from bird locomotion.

5.6.5 Classification Head

The temporal encoder produces a sequence-level embedding $\mathbf{y}_{\text{temporal}}$ that is passed through a classification head:

$$\hat{y} = \sigma(\mathbf{W}_c \cdot \mathbf{y}_{\text{temporal}} + \mathbf{b}_c) \tag{5.8}$$

where $\hat{y} \in [0, 1]$ represents the probability of the drone class. The model is trained with binary cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log(\hat{y}_n) + (1 - y_n)\log(1 - \hat{y}_n)\right] \tag{5.9}$$

5.6.6 Discriminative Signals in Gaussian Dynamics

The architecture is designed to capture discriminative signals identified through domain analysis:

Surface Geometry. Drones are rigid bodies composed of flat surfaces and slender rotors; birds are organic shapes with curved surfaces. The scale and rotation parameters of Gaussians encode local surface geometry: flat ellipsoids for drone body panels versus elongated ellipsoids for feathers.

Motion Blur Encoding. Propellers spinning at high RPM induce motion blur that manifests as semi-transparent Gaussians in the rotor disk region. This is explicitly encoded in the opacity parameter. Bird wings, while moving, maintain structural coherence and higher opacity.

Rigid vs. Deformable Dynamics. Drone components maintain fixed spatial relationships during flight (rigid-body motion), while bird anatomy exhibits coordinated deformation (wing flexion, body undulation). The temporal evolution of Gaussian spatial distributions captures these different motion patterns.

Periodicity Signatures. Propeller rotation produces high-frequency periodic signals in Gaussian opacity and position, while wing flapping produces lower-frequency oscillations with characteristic asymmetric profiles (fast downstroke, slow upstroke).

5.7 Sim-to-Real Transfer

To bridge the domain gap between synthetic training data and real-world deployment, we employ several transfer learning strategies.

5.7.1 Domain Randomization

The synthetic data pipeline implements domain randomization to prevent overfitting to simulation artifacts:

- **Atmospheric Variation:** Randomized fog density, haze levels, and lighting conditions spanning the operational envelope.
- **Sensor Characteristics:** Variable noise profiles, exposure settings, and lens distortion parameters.
- **Geometric Perturbation:** Camera pose errors within calibration uncertainty bounds, target trajectory variations, and scale randomization.

5.7.2 Robustness Branch

The network architecture includes a secondary robustness objective that penalizes sensitivity to high-frequency image variations. This forces the encoder to focus on low-frequency structural and dynamic features (which transfer well between domains) rather than high-frequency texture details (which transfer poorly).

5.7.3 Rotation Augmentation

During training, consistent rotations are applied across all frames of each sequence to maintain temporal coherence while teaching rotation invariance. All Gaussian attributes are transformed appropriately: positions via matrix multiplication, covariances via similarity transformation, quaternions via composition, and spherical harmonics via Wigner D-matrix application, while opacity remains unchanged.

5.8 Implementation Strategy

5.8.1 Hardware Platform

The system targets the NVIDIA Jetson AGX Orin (64GB) for edge deployment, balancing computational capability with power constraints suitable for remote installation.

5.8.2 Memory Management

The TBD voxel grid exploits sparsity through hash-map or sparse tensor implementations (Minkowski Engine), avoiding memory allocation for empty sky regions. Gaussian parameters are stored and processed in FP16 (half precision) to double throughput and halve memory bandwidth.

5.8.3 Camera Network

The cameras connect via Gigabit Ethernet to ensure raw uncompressed frames reach the processing unit with minimal latency jitter, essential for tight temporal synchronization within ±10ms tolerance.

5.8.4 Pipeline Latency

The complete pipeline targets [**TODO:** sub-Xms] end-to-end latency from frame capture to classification output: [**TODO:** TBD]

5.9 Conclusion

Developing a 3D occupancy and classification system for flying objects requires abandoning the appearance-based assumptions of conventional computer vision. In long-range airspace monitoring, appearance is unreliable, texture is absent, and signals are weak.

This chapter has detailed a methodology that prioritizes geometry and dynamics over appearance. By moving detection to 3D space via volumetric accumulation, we recover signals likely missed by 2D detectors. By utilizing DepthSplat for feed-forward 3D Gaussian reconstruction, we achieve the speed and fidelity required for real-time operation in textureless environments. By constructing rotation-invariant features through Vector Neurons and VN-Transformer, we ensure classification robustness across arbitrary viewing angles. Finally, by modeling the 4D temporal evolution through Mamba4D state space models, we extract the subtle kinematic signatures that distinguish mechanical drones from biological birds with linear computational complexity.

This synthesis of methods represents a theoretically grounded and practically deployable solution to the challenge of unauthorized UAV detection and classification.

6 Experiments and Evaluation

7 Discussion

8 Conclusion

This chapter concludes the thesis with a summary of the main findings and outlines directions for future work.

8.1 Conclusions

This thesis presented an end-to-end pipeline for RGB-only, multi-view detection, classification, and tracking of flying objects (drones vs. birds) targeting robust operation under imperfect calibration, rolling shutter, and partial occlusions. The key contributions are:

- 1
- 2

The experimental evaluation validated the three hypotheses:

- 1
- 2

Ablation studies demonstrated the value of each component: spatiotemporal bundle adjustment, token selection and serialization, DINO distillation, the robustness branch, and synthetic-to-real transfer strategies. The system achieves competitive accuracy on real-world data while maintaining near-real-time throughput (∼10–20 fps on a single GPU).

8.2 Future Work

Several promising directions for future work include:

- 1
- 2

Bibliography

[1] *IEC 62676 – Video Surveillance*, en-US.

[2] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[3] 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

[4] J. Luiten et al., "HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking," *International Journal of Computer Vision*, vol. 129, no. 2, pp. 548–578, Feb. 2021, arXiv:2009.07736 [cs], ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-020-01375-2

[5] 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

[6] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA: MIT Press, 1993, ISBN: 0262061589.

[7] 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

[8] 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

[9] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981. DOI: 10.1038/293133a0

[10] 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

[11] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, vol. 38, no. 3, pp. 199–218, 2000. DOI: 10.1023/A:1008191222954

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 405–421. DOI: 10.1007/978-3-030-58452-8_24

[13] 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

[14] 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

[15] 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

[16] A. Vaswani et al., "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[17] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[18] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep Sets," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[19] A. Gu and T. Dao, "Linear-Time Sequence Modeling with Selective State Spaces," en, 2024.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," en, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 779–788, ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.91

[21] 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

[22] 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

[23] 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

[24] 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

[25] D. Salmond and H. Birch, "A particle filter for track-before-detect," in *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, ISSN: 0743-1619, vol. 5, Jun. 2001, 3755–3760 vol.5. DOI: 10.1109/ACC.2001.946220

[26] F. Meyer et al., "Message Passing Algorithms for Scalable Multitarget Tracking," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018, ISSN: 1558-2256. DOI: 10.1109/JPROC.2018.2789427

[27] 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

[28] 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

[29] 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

[30] 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

[31] 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

[32] 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

[33] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, ISSN: 1051-4651, vol. 2, Aug. 2004, 28–31 Vol.2. DOI: 10.1109/ICPR.2004.1333992

[34] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, ISSN: 1063-6919, vol. 2, Jun. 1999, 246–252 Vol. 2. DOI: 10.1109/CVPR.1999.784637

[35] "Background Subtraction for Moving Cameras," en, in *Background Modeling and Foreground Detection for Video Surveillance*, T. Bouwmans, F. Porikli, B. HÃ¶ferlin, and A. Vacavant, Eds., 0th ed., Chapman and Hall/CRC, Jul. 2014, pp. 133–150, ISBN: 978-0-429-17111-6. DOI: 10.1201/b17223-10

[36] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, Vancouver, Canada, Aug. 1981, pp. 674–679.

[37] G. Farnebck, "Two-Frame Motion Estimation Based on Polynomial Expansion," en, in *Image Analysis*, J. Bigun and T. Gustavsson, Eds., Berlin, Heidelberg: Springer, 2003, pp. 363–370, ISBN: 978-3-540-45103-7. DOI: 10.1007/3-540-45103-X_50

[38] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," en, in *Computer Vision " ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 402–419, ISBN: 978-3-030-58536-5. DOI: 10.1007/978-3-030-58536-5_24

[39] Y. Wang, L. Lipson, and J. Deng, "SEA-RAFT: Simple, Efficient, Accurate RAFT for Optical Flow," 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. 36–54, ISBN: 978-3-031-72667-5. DOI: 10.1007/978-3-031-72667-5_3

[40] A. Kirillov et al., "Segment Anything," en, in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Paris, France: IEEE, Oct. 2023, pp. 3992–4003, ISBN: 979-8-3503-0718-4. DOI: 10.1109/ICCV51070.2023.00371

[41] N. Ravi et al., *SAM 2: Segment Anything in Images and Videos*, arXiv:2408.00714 [cs], Oct. 2024. DOI: 10.48550/arXiv.2408.00714

[42] 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: 978-3-030-58567-9 978-3-030-58568-6. DOI: 10.1007/978-3-030-58568-6_12

[43] 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

[44] 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

[45] 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

[46] 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

[47] 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

[48] 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

[49] 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

[50] 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

[51] 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

[52] 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

[53] 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

[54] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers,” in *Computer Vision “ ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 213–229, ISBN: 978-3-030-58452-8.

[55] 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

[56] X. Weng, J. Wang, D. Held, and K. Kitani, “AB3DMOT: A Baseline for 3D Multi-Object Tracking and New Evaluation Metrics,” *ECCVW*, 2020.

[57] 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

[58] 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,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[59] 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

[60] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, “pixelNeRF: Neural Radiance Fields from One or Few Images,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 2575-7075, Jun. 2021, pp. 4576–4585. DOI: 10.1109/CVPR46437.2021.00455

[61] A. Chen et al., “MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo,” en, in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 14 104–14 113, ISBN: 978-1-6654-2812-5. DOI: 10.1109/ICCV48922.2021.01386

[62] T. MÃ¼ller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” en, *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–15, Jul. 2022, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3528223.3530127

[63] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. M. Sajjadi, A. Geiger, and N. Radwan, “RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs,” en, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 5470–5480, ISBN: 978-1-6654-6946-3. DOI: 10.1109/CVPR52688.2022.00540

[64] F. Warburg, E. Weber, M. Tancik, A. Holynski, and A. Kanazawa, “Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs,” en, in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Paris, France: IEEE, Oct. 2023, pp. 18 074–18 084, ISBN: 979-8-3503-0718-4. DOI: 10.1109/ICCV51070.2023.01661

[65] B. Kerbl, G. Kopanas, T. LeimkÃ¼hler, and G. Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, en, arXiv:2308.04079 [cs], Aug. 2023. DOI: 10.48550/arXiv.2308.04079

[66] 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

[67] 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

[68] S. 3. Team et al., *SAM 3D: 3Dfy Anything in Images*, arXiv:2511.16624 [cs], Nov. 2025. DOI: 10.48550/arXiv.2511.16624

[69] 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

[70] 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

[71] H. Xu et al., "Depthsplat: Connecting gaussian splatting and depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2025, pp. 16 453–16 463.

[72] 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

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

[74] X. Lu, J. Fu, J. Zhang, Z. Song, C. Jia, and S. Ma, *ProSplat: Improved Feed-Forward 3D Gaussian Splatting for Wide-Baseline Sparse Views*, arXiv:2506.07670 [cs], Jun. 2025. DOI: 10.48550/arXiv.2506.07670

[75] 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

[76] 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

[77] 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

[78] 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

[79] 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

[80] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[81] 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

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

[83] 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

[84] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks," en, in *Proceedings of the 36th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, May 2019, pp. 3744–3753.

[85] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira, "Perceiver: General Perception with Iterative Attention," en, in *Proceedings of the 38th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 2021, pp. 4651–4664.

[86] 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

[87] 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

[88] X. Han, Y. Tang, Z. Wang, and X. Li, "<span style="font-variant:small-caps;">Mamba3D:</span> 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

[89] J. Liu et al., "Mamba4d: Efficient 4d point cloud video understanding with disentangled spatial-temporal state space models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2025, pp. 17 626–17 636.

[90] 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

[91] 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

[92] F. Fuchs, D. Worrall, V. Fischer, and M. Welling, "SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1970–1981.

[93] 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

[94] 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

[95] 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

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

[97] 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

[98] 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

[99] 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

[100] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.

[101] NVIDIA, *Isaac Sim*, original-date: 2025-05-28T18:38:18Z, Dec. 2025.

[102] 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

[103] 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

[104] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors," en,

[105] 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: 978-3-031-73234-8 978-3-031-73235-5. DOI: 10.1007/978-3-031-73235-5_1

[106] Z.-X. Zou et al., *Triplane Meets Gaussian Splatting: Fast and Generalizable Single-View 3D Reconstruction with Transformers*, arXiv:2312.09147 [cs], Dec. 2023. DOI: 10.48550/arXiv.2312.09147

[107] W. Wang et al., *VolSplat: Rethinking Feed-Forward 3D Gaussian Splatting with Voxel-Aligned Prediction*, arXiv:2509.19297 [cs], Sep. 2025. DOI: 10.48550/arXiv.2509.19297

[108] J. Xu, S. Gao, and Y. Shan, *FreeSplatter: Pose-free Gaussian Splatting for Sparse-view 3D Reconstruction*, arXiv:2412.09573 [cs], Sep. 2025. DOI: 10.48550/arXiv.2412.09573

[109] B. Smart, C. Zheng, I. Laina, and V. A. Prisacariu, *Splatt3R: Zero-shot Gaussian Splatting from Uncalibrated Image Pairs*, arXiv:2408.13912 [cs], Aug. 2024. DOI: 10.48550/arXiv.2408.13912

[110] Y. Fujimura, T. Kushida, K. Kitano, T. Funatomi, and Y. Mukaigawa, *UFV-Splatter: Pose-Free Feed-Forward 3D Gaussian Splatting Adapted to Unfavorable Views*, arXiv:2507.22342 [cs], Aug. 2025. DOI: 10.48550/arXiv.2507.22342

[111] H. Xu, D. Barath, A. Geiger, and M. Pollefeys, *ReSplat: Learning Recurrent Gaussian Splats*, arXiv:2510.08575 [cs], Dec. 2025. DOI: 10.48550/arXiv.2510.08575

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

A Feed-Forward 3D Gaussian Splatting Architecture Selection

This appendix analyzes feed forward 3D Gaussian Splatting (3DGS) architectures to determine the optimal approach for our multi camera flying object reconstruction pipeline. The primary objective is improving classification accuracy through high quality 3D reconstructions that faithfully represent observed geometry. We intend to retrain a feed forward model on curated Objaverse objects with domain specific fine tuning on drone and bird assets.

A.1 Design Principles

Before evaluating architectures, we establish two guiding principles for architecture selection.

A.1.1 The Importance of Depth Supervision

Feed forward 3DGS models predict Gaussian parameters (position, scale, rotation, opacity, color) directly from image features. The quality of these predictions depends critically on how well the model learns geometric relationships during training.

Depth as geometric anchor. Gaussian position $\mu \in \mathbb{R}^3$ requires accurate depth estimation to place primitives correctly in 3D space. Without explicit depth supervision, models must learn depth implicitly from photometric reconstruction loss alone. This indirect signal proves insufficient for accurate geometry, particularly in textureless regions where photometric gradients vanish.

Training stability. Depth supervision provides dense per pixel gradients that stabilize training. Models trained with depth loss converge faster and generalize better to novel scenes. DepthSplat demonstrates that combining monocular depth priors (Depth Anything V2) with multi view cost volumes yields 1.08 dB PSNR improvement over geometry only approaches.

Cross dataset transfer. For retraining on custom datasets (Objaverse + drone/bird assets), depth supervision enables learning transferable geometric representations. Models trained with depth annotations on synthetic data transfer more reliably to real imagery than those trained purely on photometric loss.

A.1.2 Rejection of Diffusion Based Reconstruction

Several recent methods employ diffusion models for 3D reconstruction, including Zero123, One2345, and diffusion enhanced Gaussian splatting variants. We explicitly exclude these architectures from consideration.

Hallucination risk. Diffusion models generate plausible completions for unobserved regions by sampling from learned distributions. For novel view synthesis, this produces visually appealing results. For classification, hallucinated geometry introduces features that do not exist in the input observations.

Classification pipeline integrity. Our pipeline classifies objects based on reconstructed Gaussian features. If the reconstruction model hallucinates drone like geometry for an ambiguous input, the classifier receives fabricated evidence. Conversely, if a novel drone design was absent from diffusion training, the model may generate incorrect geometry, causing misclassification.

Principle. We require reconstruction methods that represent *only what is observable* in the camera feeds. Uncertainty should propagate as uncertainty (e.g., low opacity, high variance), not as hallucinated structure. This rules out any architecture where a generative prior fills in unobserved regions.

A.2 Architecture Taxonomy

Feed forward 3DGS methods bypass per scene optimization by directly predicting Gaussian parameters from input images. We categorize these into four architecture families.

A.2.1 Pixel Aligned Architectures

Pixel aligned methods treat each source pixel as a potential 3D Gaussian, predicting per pixel attributes before unprojection.

Probabilistic Depth Methods

pixelSplat [66] (CVPR 2024 Best Paper Honorable Mention) introduced generalizable Gaussian splatting for image pairs. The method predicts Gaussian primitives through probabilistic depth estimation along epipolar lines. For each pixel, depth values are sampled via triangulation, and cross attention with frequency encoded positional embeddings identifies correspondences. Rather than predicting single depth values, pixelSplat outputs discrete probability distributions over depth buckets, using a reparameterization trick where Gaussian opacity equals bucket probability. The probabilistic formulation handles depth ambiguity gracefully but produces floating artifacts in textureless regions where all depth hypotheses appear equally valid.

Splatter Image [67] reconstructs 3D Gaussians from single views at 38 FPS via image to image translation. The method uses purely 2D convolutional operators to predict one Gaussian per pixel. Single view operation necessarily relies on learned priors for depth and unobserved geometry, limiting applicability to our multi view setup.

Cost Volume Methods

MVSplat [69] replaces probabilistic depth with explicit cost volume representations. Plane sweeping constructs a 3D volume where each voxel stores cross view feature similarity at that depth hypothesis. Geometry emerges where feature variance across views is minimized. MVSplat achieves 26.39 PSNR on RealEstate10K with 12M parameters (10$\times$ fewer than pixelSplat) at 22 FPS. The geometric inductive bias improves cross dataset generalization but requires sufficient texture for feature matching.

GPS Gaussian [70] introduces iterative disparity estimation inspired by RAFT Stereo with 3D correlation volumes. Critically, it estimates depth for both source views to enable symmetric Gaussian representation. This eliminates floating artifacts from asymmetric depth uncertainty.

DepthSplat [71] (CVPR 2025) demonstrates bidirectional synergy between depth estimation and Gaussian Splatting. The architecture employs two complementary branches:

- **Multi view branch:** Plane sweep stereo cost volumes providing geometric constraints from view correspondence
- **Single view branch:** Features from pretrained Depth Anything V2 providing monocular depth priors

The fusion of monocular priors with multi view geometry achieves 27.47 PSNR on RealEstate10K, a 1.08 dB improvement over MVSplat. DepthSplat's dual branch design is particularly relevant for our retraining strategy: the monocular branch can be fine tuned on aerial imagery while the multi view branch provides geometric grounding.

Limitation for textureless backgrounds. All cost volume methods compute photometric variance to identify surfaces. For objects against uniform sky, depth hypotheses produce similar low variance costs, degrading geometric signal. This limitation motivates either (1) operating on object crops where the target provides texture, or (2) incorporating silhouette based constraints.

A.2.2 Transformer Based Large Reconstruction Models

Large Reconstruction Models (LRMs) tokenize input images and process them through transformer blocks trained on massive datasets.

GS LRM [72] patchifies input images and processes concatenated multi view tokens through sequential transformer blocks. Per pixel Gaussian parameters are decoded directly from output tokens. The model predicts high quality primitives from 2 to 4 posed images in 0.23 seconds on A100 GPU.

LGM [105] fuses 4 fixed view inputs via transformer backbone, generating objects in approximately 5 seconds.

Assessment. Transformer architectures learn strong semantic priors from Objaverse training. Their training distributions (centered objects, close range) diverge from surveillance scenarios. However, transformer based multi view fusion is amenable to retraining on domain specific data. GS LRM's architecture provides a template for custom training pipelines.

A.2.3 Hybrid and Volumetric Architectures

These methods construct intermediate 3D representations before decoding Gaussians.

Triplane Meets Gaussian Splatting [106] decodes both explicit point clouds and implicit triplane features. Gaussian attributes are queried from the triplane at point locations. This balances explicit geometry with implicit feature continuity.

VolSplat [107] predicts a voxel grid and generates Gaussians from voxel features. The voxel aligned approach avoids view bias artifacts inherent to pixel aligned unprojection. Multi view consistency improves over pixel aligned alternatives.

Assessment. Volumetric approaches provide natural integration with voxel based detection pipelines. The 3D grid representation enables seamless handoff from detection to reconstruction.

A.2.4 Pose Free and Unfavorable View Architectures

FreeSplatter [108] and **Splatt3R** [109] predict Gaussians without known camera parameters. Splatt3R builds on MASt3R, predicting pixel aligned pointmaps from uncalibrated pairs (4 FPS).

UFV Splatter [110] adapts to "unfavorable views": off center objects, unusual angles, and non standard framing. Standard Objaverse trained models fail on such inputs; UFV Splatter's robustness addresses a practical surveillance requirement.

Assessment. Our cameras are calibrated, so pose free capability is unnecessary. However, UFV Splatter's unfavorable view robustness is valuable since flying objects appear at arbitrary frame positions.

A.2.5 Recurrent Refinement Architectures

ReSplat [111] introduces a feed forward recurrent network that iteratively refines 3D Gaussians without explicit gradient computation. The method uses DepthSplat as its initialization model but operates in a 16× subsampled 3D space, producing 16× fewer Gaussians than per pixel methods. The key insight is that the Gaussian splatting rendering error serves as a feedback signal, guiding recurrent updates to improve reconstruction quality.

On the DL3DV benchmark with 8 input views at 512×960 resolution, ReSplat achieves 27.70 PSNR after 4 iterations, a 3.53 dB improvement over DepthSplat (24.17 PSNR) while using only 246K Gaussians compared to DepthSplat's 3.9M. The recurrent refinement also improves generalization to unseen datasets, view counts, and image resolutions.

Assessment. ReSplat represents a promising direction for our pipeline. The iterative refinement could help address textureless background challenges by progressively correcting reconstruction errors. The reduced Gaussian count (16× fewer) aligns well with edge deployment constraints. However, the authors have not released code or pretrained models as of December 9th, 2025, despite stating their intention to do so. Furthermore, the paper explicitly notes that ReSplat targets scene level reconstruction rather than object centric settings. We therefore cannot adopt ReSplat for our current implementation but consider it a promising future direction once code becomes available and object centric adaptation is explored.

A.3 Training Considerations

We plan to retrain a feed forward model on curated data. This section identifies architecture properties relevant to custom training.

A.3.1 Data Requirements

Objaverse curation. The full Objaverse dataset contains more than 800K objects with variable quality. Effective training requires filtering for: (1) complete geometry without holes, (2) realistic materials, (3) appropriate scale. Aircraft, vehicles, and animals provide useful shape priors.

Domain specific assets. Drone CAD models (DJI, custom builds) and rigged bird models enable fine tuning. Synthetic rendering in Isaac Sim provides ground truth depth, camera poses, and segmentation masks.

A.3.2 Depth Supervision Strategy

Synthetic data. Isaac Sim provides perfect depth maps for all rendered views. Training with $\mathcal{L}_{\text{depth}} = \|\hat{d} - d_{\text{gt}}\|_1$ alongside photometric loss anchors geometric learning.

Real data. Depth Anything V2 or similar monocular estimators provide pseudo ground truth for real imagery. DepthSplat's architecture naturally accommodates this: the monocular branch processes pretrained depth features while the multi view branch provides geometric refinement.

A.3.3 Architecture Retrainability

Not all architectures are equally amenable to retraining:

- **MVSplat / DepthSplat:** Modular design with clear separation of encoder, cost volume, and decoder. Well suited for component wise fine tuning.
- **GS LRM:** Monolithic transformer requires full retraining or careful layer freezing. Higher compute requirements.
- **pixelSplat:** Epipolar attention mechanism is geometry aware but entangled with probabilistic depth. Moderate retraining complexity.

A.4 Architecture Selection: DepthSplat

This section presents our comparative analysis leading to the selection of DepthSplat as the reconstruction backbone. We evaluate candidate architectures against our pipeline requirements: geometric fidelity for classification, adaptability to aerial domains, robustness to textureless backgrounds, and practical considerations including open source availability.

A.4.1 Elimination of Transformer Based Approaches

GS LRM offers strong learned priors and competitive reconstruction quality (0.23s inference on A100). However, several factors disqualify it for our application.

Reproducibility concerns. The original authors (Adobe Research) have not released official code or pretrained weights. Only an unofficial third party implementation exists[1], which implements only Stage 1 training and lacks evaluation code. For a research pipeline requiring custom retraining, reliance on incomplete unofficial implementations introduces unacceptable risk.

Training distribution mismatch. GS LRM's 8×8 patch tokenization assumes close range objects (1.5 to 2.8 normalized units) filling the frame. Flying objects at distances of 10 to 100 meters appear as small targets with limited resolution, often failing to meet the minimum density required by the patchification scheme. Retraining would require architectural modifications beyond simple fine tuning.

A.4.2 Limitations of MVSplat for Object Centric Reconstruction

MVSplat provides an attractive baseline: minimal parameters (12M), fast inference (22 FPS), and explicit geometric grounding via cost volumes. However, the architecture exhibits fundamental limitations for our object centric setting.

Scene level design assumptions. The MVSplat authors explicitly state their method "mainly focuses on... scene-level reconstruction" [69]. The architecture assumes sufficient overlap among input views with continuous background geometry. These assumptions are violated when reconstructing isolated objects against sky.

[1]https://github.com/InternRobotics/gs-lrm-unofficial

Floating artifacts without background. When applied to segmented objects or scenes lacking background geometry, MVSplat produces floating Gaussian artifacts. The cost volume relies on photometric consistency across the entire image. Without background texture to anchor depth hypotheses, spurious matches propagate to incorrect 3D locations. This behavior is documented in community discussions and follow up work (MVSplat360) which notes "performance degradation with low texture or repetitive patterns."

No monocular fallback. In regions where multi view matching fails (textureless sky, specular surfaces), MVSplat has no secondary depth signal. The architecture lacks the monocular prior that could provide reasonable depth estimates when geometric correspondence degrades.

A.4.3 Advantages of DepthSplat

DepthSplat addresses the limitations of both transformer and pure cost volume approaches through its dual branch architecture.

Complementary depth estimation. The fusion of monocular priors (Depth Anything V2) with multi view cost volumes provides robustness across challenging conditions. When cost volume matching degrades in textureless regions, the monocular branch provides reasonable depth estimates. Conversely, when monocular predictions are ambiguous, multi view geometry provides correction. This complementarity yields 1.08 dB PSNR improvement over MVSplat on RealEstate10K (27.47 vs. 26.39).

Modular retrainability. The clear separation between monocular encoder, cost volume module, and Gaussian decoder enables targeted fine tuning:

1. The Depth Anything V2 backbone can be fine tuned on aerial imagery to improve monocular priors for drones and birds
2. The cost volume decoder can be retrained on Isaac Sim synthetic data with ground truth depth
3. Individual components can be frozen during ablation studies

Open source availability. DepthSplat provides official code and pretrained weights[2], maintained by ETH Zurich's Computer Vision Group. Pretrained models are available on HuggingFace, enabling immediate experimentation before committing to full retraining.

A.4.4 Consideration of ReSplat for Future Work

ReSplat [111] represents the current state of the art in feed forward Gaussian splatting, achieving 3.5 dB PSNR improvement over DepthSplat through recurrent refinement. The method builds directly on DepthSplat's architecture for initialization, making it a natural extension of our chosen approach.

The iterative refinement mechanism is particularly appealing for our application. The rendering error feedback could help correct reconstruction artifacts in textureless sky regions by progressively improving Gaussian placement. The 16× reduction in Gaussian count also aligns with edge deployment constraints on NVIDIA Jetson platforms.

However, we cannot adopt ReSplat for our current implementation due to two practical constraints. First, the authors have not released code or pretrained models as of December 9th, 2025, despite stating "We will release our code, pre-trained models, training and evaluation scripts." Second, the paper explicitly notes that ReSplat targets scene level benchmarks, and the comparison with SplatFormer reveals that adapting such methods to object centric settings is "particularly challenging" and requires careful normalization and grid size tuning.

We consider ReSplat a promising direction for future work once code becomes available and object centric adaptation strategies are developed.

A.4.5 Addressing Textureless Backgrounds

While DepthSplat's monocular branch provides some robustness to textureless regions, sky backgrounds remain challenging. The DepthSplat authors acknowledge that cost volume methods "inherently suffer from the limitation of feature matching in challenging situations like texture-less regions" [112].

[2] https://github.com/cvg/depthsplat

We propose augmenting the training objective with a silhouette consistency loss:

$$\mathcal{L}_{\text{silhouette}} = \text{BCE}\left(\sum_i \alpha_i \cdot G_i(\mathbf{p}), M_{\text{gt}}(\mathbf{p})\right) \tag{A.1}$$

where α_i is Gaussian opacity, $G_i(\mathbf{p})$ is the 2D Gaussian contribution at pixel $\mathbf{p}$, and M_{gt} is the ground truth foreground mask. This loss encourages the reconstructed Gaussians to match object silhouettes even when photometric gradients vanish, providing supervision signal independent of texture.

Isaac Sim provides perfect segmentation masks for synthetic training data, enabling this supervision without additional annotation effort. Similar silhouette based constraints have proven effective in optimization based methods (SplaTAM, GS SFS) but remain unexplored in feed forward architectures.

A.4.6 Alternative Considerations

UFV Splatter addresses unfavorable viewing conditions relevant to surveillance (off center objects, non standard framing). However, as a recent preprint (July 2025), it lacks the maturity and community validation of DepthSplat. Its pose free design may sacrifice geometric precision when calibration is available, as in our setup. We consider UFV Splatter's "recentering" strategy as a potential data augmentation technique rather than a primary architecture choice.

GPS Gaussian provides symmetric depth estimation eliminating floating artifacts. However, it targets human reconstruction with dense multi view capture (4+ synchronized cameras at close range), diverging from our sparse wide baseline setup.

pixelSplat pioneered feed forward Gaussian splatting but requires 80GB VRAM (A100/H100) for training, limiting accessibility. The probabilistic depth formulation handles ambiguity gracefully but does not incorporate monocular priors for textureless fallback.

A.5 Excluded Architectures

The following architectures are explicitly excluded:

Diffusion based methods (Zero123, One2345, diffusion enhanced 3DGS): Hallucination of unobserved geometry corrupts classification features.

Single view methods (Splatter Image, SAM 3D): Reliance on learned priors for depth and unobserved regions conflicts with multi view setup.

Pose free methods (Splatt3R, FreeSplatter): Sacrifice geometric precision when calibration is available.

A.6 Conclusion

For classification oriented 3D reconstruction, architecture selection prioritizes geometric fidelity over visual plausibility. Depth supervision during training proves critical for learning transferable representations. Diffusion based methods are excluded to prevent hallucinated features from corrupting downstream classification.

Table A.1 summarizes our comparative analysis. DepthSplat emerges as the optimal choice due to its dual branch design combining monocular priors with multi view geometry, official open source implementation, and modular architecture supporting targeted fine tuning. The addition of silhouette consistency loss during retraining addresses the inherent cost volume limitation for textureless backgrounds.

Table A.1: Comparative analysis of feed forward 3DGS architectures for aerial object reconstruction. DepthSplat provides the best combination of performance, modularity, and practical availability for our object centric retraining requirements.

Criterion	**DepthSplat**	**MVSplat**	**GS LRM**	**pixelSplat**	**ReSplat**
PSNR (RealEstate10K)	27.47	26.39	28.10	25.89	**29.75**
PSNR (DL3DV, 8 views)	24.17	22.49	—	—	**27.70**
Parameters	354M	**12M**	305M	125M	223M
Inference speed	10 FPS	**22 FPS**	4.3 FPS	11 FPS	7 FPS
Monocular depth prior	✓	✗	✗	✗	✓[†]
Multi view cost volume	✓	✓	✗	✗	✓[†]
Object centric support	✓	✗[*]	✓	✓	✗[‡]
Textureless robustness	Partial[§]	✗	✗	✗	Improved
Iterative refinement	✗	✗	✗	✗	✓
Official open source	✓	✓	✗[‖]	✓	✗[**]
Modular retrainability	✓	✓	✗	Partial	Unknown
Training hardware	A6000	RTX 4090	A100	A100 80GB	GH200
Selected	✓				

[*]MVSplat produces floating artifacts on segmented objects without background geometry (scene level design).
[†]ReSplat uses DepthSplat for initialization, inheriting its dual branch design.
[‡]ReSplat explicitly targets scene level benchmarks; object centric adaptation is noted as "particularly challenging."
[§]Monocular branch provides fallback; silhouette loss proposed for further improvement.
[‖]Only unofficial third party implementation available.
[**]Code not released as of December 9th, 2025, despite authors' stated intention.