Distributed 3D Reconstruction of Aerial Footage

Ihor Barakaiev
Stanford University
igorb@stanford.edu

Abstract

Reconstructing detailed 3D models from small-scale imagery is well-established via methods such as COLMAP, DUSt3R, and MASt3R. Extending these approaches to extensive aerial surveys, however, remains challenging due to their high computational and memory demands. We present a pipeline that (1) partitions long drone videos into coherent chunks of consecutive frames using a novel LightGlue-based temporal matcher, (2) processes each chunk independently with DUSt3R and MASt3R to obtain dense point maps, and (3) merges the per-chunk reconstructions by computing pairwise relative transforms. This modular strategy enables scalable, high-fidelity 3D reconstruction over potentially thousands of frames, requiring only RGB frames as input. Our system introduces significant improvements in runtime and modularity over baseline methods.

In battlefield zones, disaster areas, and other time-critical environments, there is a growing need for fast, accurate 3D reconstructions of large-scale terrain. Drone footage provides a rich source of visual data, but existing pipelines often require hours to process—or fail entirely due to GPU memory limits and computational bottlenecks, especially when dealing with thousands of high-resolution frames. In such scenarios, teams need updated 3D maps within minutes, using only lightweight equipment and standard RGB video. This pressing need for scalable, near real-time reconstruction motivates our work: a modular, distributed pipeline that transforms long drone videos into coherent 3D scenes efficiently and reliably, without sacrificing fidelity.

Traditional 3D reconstruction pipelines work well for short image collections but scale poorly to long aerial drone surveys, where pairwise comparisons grow quadratically with the number of frames. A single 60-minute flight can yield over 3,600 frames at 1 FPS, making naïve approaches both computationally infeasible and memory-intensive. Even state-of-the-art learning-based systems such as MASt3R and DUSt3R exhaust GPU memory after a few dozen images if building a dense scene graph with pairwise matches.

However, aerial footage is highly redundant: frames far apart in time rarely overlap, and matching all pairs is both wasteful and unnecessary. MASt3R addresses this with a "sliding window" strategy that approximates linear runtime by limiting the number of pairwise comparisons. However, this window is slid in a fixed, somewhat arbitrary manner and still requires keeping the entire scene in memory—making distribution across machines impossible and eventually hitting memory limits again. This motivates our key idea: instead of sliding an arbitrary window, we (a) pre-process long videos into overlapping, visually coherent *chunks*, (b) process each chunk independently with an off-the-shelf 3D reconstructor, and (c) merge the resulting local scenes using frame-level alignment.

Our pipeline consists of three main stages:

- We design a LightGlue-based temporal matching module to extract overlapping chunks from a video (up to 24 frames in a chunk).
- We apply classic pairwise MASt3R to reconstruct the scene in local coordinate frame.
- We align individual chunk reconstructions in global frame based on camera poses of overlapping frames.

In summary, we (1) develop a scalable, chunk-based reconstruction framework that distributes MASt3R/DUSt3R across coherent subsets of video, and (2) demonstrate that our method achieves similar quality to baseline methods while significantly improving runtime and modularity.

Abandoned Approach: Real-Time Gaussian Splatting. The original goal of this project was to support real-time rendering from novel angles using 3D Gaussian Splatting instead of point clouds. This method offers continuous radiance field rendering with efficient memory usage. However, the official implementation [1] only supports rendering on Linux or Windows platforms. As a macOS user without access to GUI-supported Linux instances (e.g., via AWS headless VM), this path proved impractical.

We attempted to use OpenSplat [9], a cross-platform implementation with macOS support, but its rendering speeds

were significantly below real-time. We further investigated DroneSplat [8], which uses Gaussian Splatting enhanced by voxel-guided optimization and dynamic masking. Unfortunately, DroneSplat did not scale well to our large aerial datasets. Moreover, global alignment of multiple Gaussian scenes turned out to be a significantly harder research problem compared to point cloud merging. In light of these technical and scalability limitations, we opted for a simpler and more robust point cloud-based approach. After all, the primary goal of the paper was 3D reconstruction, not necessarily realtime 3D rendering from novel angles. Additionally, our pipeline provides sufficient improvement for 3D point cloud reconstruction which would be very useful as input to 3D gaussian splatting as well.

1. Related Work

COLMAP [6]. COLMAP is a feature-rich Structure-from-Motion (SfM) and Multi-View Stereo (MVS) system that has become a de facto standard for offline 3D reconstruction from unordered image collections. It supports incremental, global, and hierarchical SfM pipelines, combining SIFT-based feature extraction with exhaustive matching, geometric verification, and bundle adjustment. Despite its robustness and completeness, COLMAP struggles to scale to very long image sequences due to quadratic growth in image pair evaluations and memory-heavy bundle adjustment. Our pipeline addresses these scalability issues by chunking videos into coherent segments and processing each independently, which avoids COLMAP's global optimization bottlenecks.

SuperGlue [5]. SuperGlue reframes local feature matching as a graph optimization problem using a transformer-like architecture with self- and cross-attention. It achieves strong performance under extreme viewpoint and illumination changes and introduces optimal transport for partial assignment matching. However, it is computationally demanding and memory-intensive, making it less practical for long video sequences or real-time applications. Our use of LightGlue retains the matching robustness of SuperGlue while improving efficiency and scalability.

LoFTR [7]. LoFTR proposes a detector-free dense matching architecture that directly correlates dense image grids, bypassing keypoint detection. It achieves state-of-the-art performance in low-texture regions and under significant appearance changes. Nonetheless, the dense nature of its architecture imposes a high memory and runtime cost, limiting its viability for high-resolution or large-scale applications like aerial video. In contrast, our approach uses sparse, adaptable matchers (e.g., LightGlue) to maintain efficiency across long footage.

LightGlue [3]. LightGlue builds on SuperGlue's transformer-based architecture but introduces adaptive computation and early pruning of unmatchable points, drastically improving speed and scalability. It replaces Sinkhorn normalization with faster partial assignment logic and is lightweight enough for deployment in real-time or large-scale scenarios. We leverage LightGlue as the backbone of our chunk discovery module, enabling us to efficiently identify overlapping sequences of frames in long drone videos.

DUSt3R [10]. DUSt3R is a transformer-based pipeline for dense stereo matching that can infer camera poses, pixel correspondences, and depth maps from unposed images. Its geometry-aware architecture learns dense 3D representations directly from image pairs. DUSt3R's strengths lie in producing accurate pointmaps from limited data without explicit calibration. However, it is memory-bound and does not scale well to thousands of frames. We use DUSt3R and its successor MASt3R within individual chunks to produce dense reconstructions where their memory constraints are manageable.

MASt3R [2]. MASt3R extends DUSt3R with a dual-head architecture that regresses both 3D pointmaps and dense descriptors per image. It introduces a matching-aware training loss to improve pose accuracy and robustness. While MASt3R achieves strong results on benchmark scenes, it is not designed for long-form input and fails under memory constraints when reconstructing dense pairwise matches across large datasets. By applying MASt3R within small, coherent chunks, our method preserves reconstruction fidelity while enabling distributed and scalable inference.

DINOv2 [4]. DINOv2 is a self-supervised vision transformer trained for general-purpose feature extraction across downstream tasks like classification and segmentation. Its embeddings are resilient to texture and appearance changes, making it suitable for tasks like scene clustering. We experimented with using DINOv2 to cluster drone frames by appearance, but found that its emphasis on semantic similarity (e.g., sky vs. ground) did not reliably correspond to spatial continuity—an essential factor for SfM. Thus, we opted for overlap-aware chunking based on feature matches.

3D Gaussian Splatting [1]. Gaussian Splatting offers real-time rendering of radiance fields by representing scenes with anisotropic Gaussians rather than explicit geometry. This enables continuous view synthesis with impressive visual fidelity and interactive framerates. However, existing pipelines assume a single, coherent scene and lack tools for globally aligning splats across disconnected chunks or

Figure 1. GPS trajectories for all flights over Lake Lagunita.

flights. Our attempt to use Gaussian Splatting revealed that even small stitching errors lead to visual artifacts, limiting its usability in large-scale multi-chunk reconstruction tasks.

DroneSplat [8]. DroneSplat adapts Gaussian Splatting for drone imagery by integrating voxel-guided stereo priors and masking out dynamic elements. It performs well in short, continuous drone sequences, but assumes consistent forward motion and global trajectory coherence. When applied to our distributed chunk-based pipeline, DroneSplat exhibited stitching artifacts and performance degradation, highlighting its limitations in fragmented or multi-session drone footage.

OpenSplat [9]. OpenSplat is a macOS-compatible Gaussian Splatting viewer that offers lower hardware requirements compared to the Linux-only reference implementation. While it allowed us to test splatting on macOS, its rendering performance was significantly below real-time, especially on high-resolution aerial reconstructions. We ultimately abandoned splatting-based rendering in favor of point cloud fusion due to both computational and alignment challenges.

2. Data

We conducted several 30–120 second flights over Lake Lagunita using a single DJI Mini 3 Pro, alternating between forward-facing (at different angles) and downward orientations. Each video was recorded at 4K 30 fps. From each video, we sampled one frame per second and parsed synchronized telemetry captions to recover GPS (latitude, longitude, altitude) and camera parameters (ISO, shutter speed, aperture, focal length, etc.). This yields a chronologically ordered set of geo-tagged frames, each paired with its metadata, which serves as the input to our chunking and reconstruction stages. There were a total of 237 resulting frames.

Figure 2. Sample frames from several separate flights.

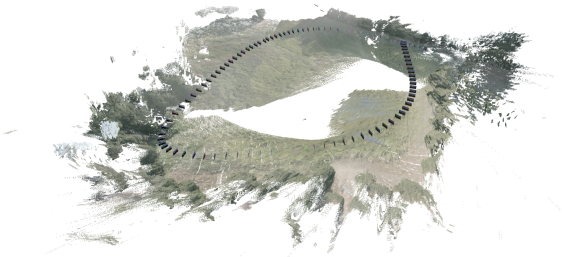

Figure 3. MASt3R reconstruction of a circular flight over Lake Lagunita using "logwin" mode.

3. Methods

3.1. Baseline: MASt3R on Full Sequence Without Chunking

For a particular flight over Lake Lagunita (displayed in red in Figure 1), we collected 110 frames (sampled at 1 FPS). Given $N = 110$ sampled frames, a full pairwise run requires approximately $2\binom{N}{2} \approx 12{,}000$ MASt3R comparisons. Each pair invokes binocular reconstruction to regress dense point maps and descriptors. On a single H100 GPU, this is estimated to take 45 minutes but results in crashing due to memory exhaustion. To mitigate this, we ran MASt3R with "logwin" mode, which works by sliding a dynamic window. This approach took 10 minutes and yielded the 3D reconstruction in Figure 3.

Lower-FPS Baseline. We also evaluated a downsampled version of the same video at 0.5 FPS (55 frames) to assess whether reducing input density could alleviate MASt3R's runtime and memory limitations. When run in "logwin" mode, the reconstruction failed entirely due to sparse keypoints and weak matches across wide baselines, producing a fragmented and unusable point cloud (Figure 4).

We then reran MASt3R at 0.5 FPS using full pairwise matching. This completed in 8 minutes and produced a denser point cloud (Figure 5), but the final reconstruction suffered from poor stitching between camera poses, result-

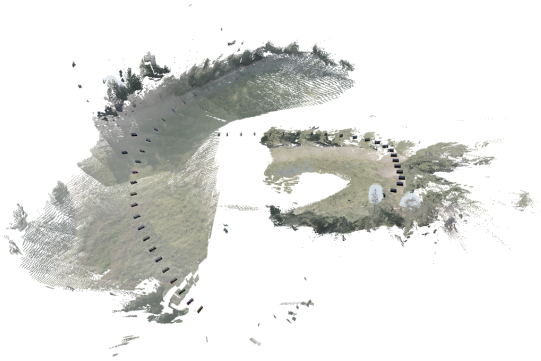

Figure 4. Failed reconstruction using MASt3R "logwin" mode at 0.5 FPS. Low keypoint density caused collapse.

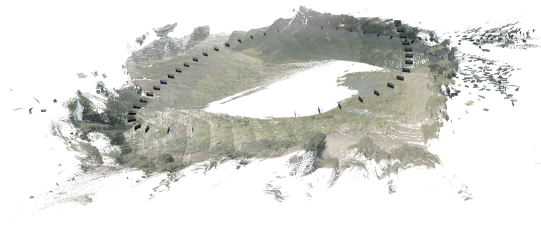

Figure 5. Accurate reconstruction using MASt3R with full pairwise mode at 0.5 FPS. Runtime: 8 minutes.

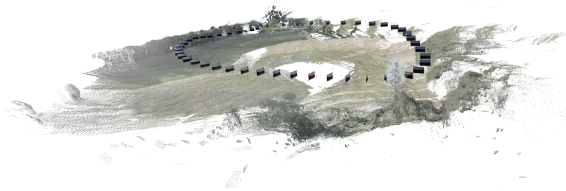

Figure 6. Alternate view of the 0.5 FPS MASt3R pairwise reconstruction. Misaligned camera poses reveal poor global consistency.

ing in inconsistent global structure (Figure 6). The issues were especially visible from wider viewing angles.

Limitations.

- **Quadratic growth.** Cost grows as $\mathcal{O}(N^2)$ in time and memory. In "logwin" mode, cost grows at approximately $\mathcal{O}(N)$ but is not parallelizable and will eventually hit memory limitations for long footages.

- **Redundant comparisons.** Most image pairs are non-overlapping and produce no useful matches when using pairwise comparison. This is partially mitigated with the sliding window approach, but some comparisons may still be redundant.

- **Quality**: "logwin" mode works at the expense of accuracy, which failed to produce a coherent global scene for the downsampled 0.5 FPS baseline.

3.2. Chunked Reconstruction Pipeline

To overcome these limits, we first partition videos into overlapping, temporally coherent *chunks*, each processed independently. Chunks reduce pairwise explosion and enable distributed computation.

Chunking offers two core benefits:

- **Locality.** Within each chunk of size $m \ll N$, the cost is $\mathcal{O}(m^2)$.

- **Parallelism.** Chunks can be processed independently and merged downstream.

3.3. LightGlue-Based Chunk Discovery

To partition long videos into coherent sequences for independent 3D reconstruction, we developed a custom LightGlue-based chunking algorithm.

Given a chronologically ordered list of video frames, our method iteratively selects a base frame I_b, then adaptively determines the longest subsequent sequence $\{I_b, I_{b+1}, ..., I_{b+\ell}\}$ such that all frames in the chunk share sufficient visual overlap with the base. This is determined via feature matching.

To find this length ℓ, we use a three-stage search:

1. **Initial probe:** We attempt to match frame I_b with a probe frame I_{b+k}, where $k = 4$ initially. If the number of matches exceeds a threshold (200), the probe is considered successful.

2. **Exponential search:** We then double k until the number of matches falls below the threshold or a maximum offset is reached, identifying the upper bound of coherent chunk length.

3. **Binary search:** A final binary search is conducted in the valid interval to pinpoint the longest viable chunk $\ell = k^* + 1$.

We resize all images to 720p before matching, and use SuperPoint + LightGlue for efficient feature extraction and matching. The entire chunking process runs on a single GPU in less than 30 seconds for 100+ frames.

Once a chunk is identified, we slide the base index forward by $\lfloor 0.75 \cdot \ell \rfloor$ to ensure 25% overlap between successive chunks. This guarantees that shared frames can be used for alignment during the stitching stage.

Alternative methods. We also experimented with using LoFTR and SuperGlue instead of LightGlue. While they produced valid matches, they were significantly slower in

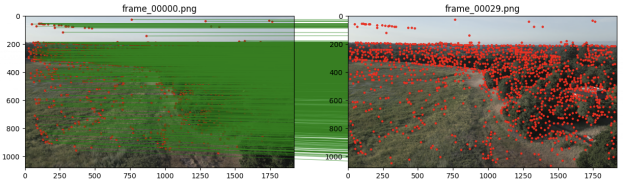

Figure 7. Matching features between two consecutive frames in a chunk.

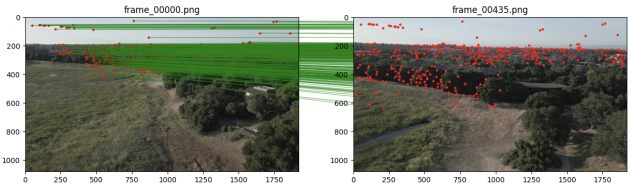

Figure 8. Matching features between first and last frames in a chunk.

both inference time and memory usage, making them impractical for long videos. We additionally tested DINOv2 embeddings for unsupervised clustering of frames, aiming to identify scene similarity without temporal constraints. However, this approach often grouped frames based on superficial cues (e.g., dominant sky pixels, camera angle) rather than shared scene content. More critically, DINOv2-based clusters lacked frame-to-frame overlap, which is essential for reliable alignment in downstream stitching.

3.4. Per-Chunk 3D Reconstruction

Each chunk is processed using the MASt3R pipeline without modification with full pairwise scene reconstruction. This yields one point cloud and set of inferred camera poses per chunk.

3.5. Scene Stitching and Alignment

Once each chunk has been reconstructed into its own scene with inferred camera poses and point cloud, we align and merge them pairwise. Chunks that share overlapping frames are identified, and their common frames are used as anchors to compute a relative transform between the two coordinate systems.

Given camera-to-world matrices $\{\mathbf{C}_i^{(1)}\}$ from chunk 1 and $\{\mathbf{C}_i^{(2)}\}$ from chunk 2 for the same image filenames, we estimate a rigid transform $\mathbf{T}_{2\to1}$ such that:

$$\mathbf{C}_i^{(1)} \approx \mathbf{T}_{2\to1} \cdot \mathbf{C}_i^{(2)}$$

The transform is computed using averaged relative camera poses with rotation averaging to maintain orthogonality. This procedure works well for flat terrains and stable camera orientations.

Limitations. This merging algorithm is not globally optimal. It assumes roughly planar terrain and limited camera tilt variation. In uneven terrain or with large viewpoint changes, stitching accuracy may degrade. A more principled approach would involve estimating a global similarity transform using algorithms like Umeyama's method for rigid alignment. Additionally, MASt3R estimates camera intrinsics to be slightly different across scenes, so an additional rescaling step is needed for better accuracy and alignment. With the current approach, we observed some minor spatial gaps for parts of the scene that were stitched.

This entire process is implemented in our provided stitching script (see Appendix).

4. Experiments

We evaluated our pipeline on two variants of a circular drone flight over Lake Lagunita: one sampled at 1 FPS (110 frames), and a downsampled version at 0.5 FPS (55 frames). These experiments demonstrate both the scalability and robustness of our chunked reconstruction approach.

110-frame reconstruction. On the full-resolution sequence (110 frames at 1 FPS), we used LightGlue to partition the video into 9 overlapping chunks, each containing 10–23 frames. The chunking step completed in approximately 20 seconds. Each chunk was then processed independently using full pairwise MASt3R reconstruction. On a single H100 GPU, the entire reconstruction took approximately 6 minutes, and would take just under 3 minutes when parallelized across 8 GPUs. The stitched final reconstruction is shown in Figure 9.

55-frame reconstruction (0.5 FPS). To assess whether our pipeline performs reliably under reduced frame density, we repeated the process on a downsampled version of the same video. MASt3R failed to reconstruct this sequence in "logwin" mode due to sparse keypoints and wide baselines (Figure 4). Even with full pairwise matching, while the reconstruction ran in 8 minutes and yielded a dense point cloud, it suffered from poor stitching and global misalignment across camera poses (Figures 5 and 6).

In contrast, our chunked pipeline completed the same task in 4 minutes and 40 seconds (or roughly est. 2 minutes with parallelization) and produced a clean, coherent reconstruction with no noticeable drop in quality (Figure 10). These results suggest that chunked reconstruction is not only more efficient, but also more stable than baseline methods when operating on sparse inputs.

Scalability. The benefits of chunked reconstruction scale significantly with input size. For a hypothetical 3-hour drone video sampled at 1 FPS (10,000 frames), MASt3R would be unable to process the full sequence even using "logwin" mode due to memory constraints. Our pipeline, however, would complete in under one hour (20 minutes to

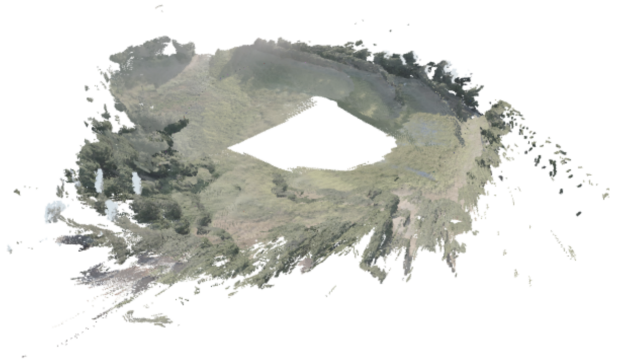

Figure 9. Final merged reconstruction of the circular drone flight using chunked reconstruction and scene alignment.

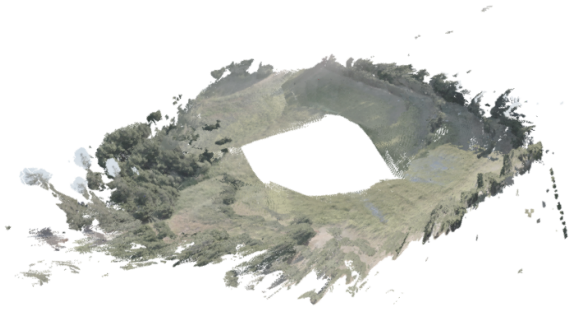

Figure 10. Reconstruction of the circular drone flight using our pipeline at 0.5 FPS (55 frames). Note no noticeable drop in quality.

Method	FPS	Frames	1 GPU	8 GPUs
MASt3R (pairwise)	1.0	110	-	-
MASt3R (logwin)	1.0	110	10 min	-
Ours	1.0	110	**6 min**	$\approx$ 3 min
MASt3R (pairwise)	0.5	55	8 min	-
MASt3R (logwin)	0.5	55	-	-
Ours	0.5	55	**4:40 min**	$\approx$ 2 min
MASt3R (pairwise)	1.0	10,000	-	-
MASt3R (logwin)	1.0	10,000	-	-
Ours	1.0	10,000	$\approx$ 7.5 hr	$\approx$1.75 hr

Table 1. Runtime comparison between MASt3R with sliding window ("logwin") and our chunked reconstruction method. Our pipeline supports parallelization and scales efficiently. The metrics for 10000 frames and for 8 GPUs are estimated.

chunk, 75 minutes to reconstruct in parallel, and 1–3 minutes to stitch).

We also tested our pipeline on a linear trajectory consisting of six chunks spanning a path from the BBQ pit to the Jerry house. Each chunk was reconstructed independently and then merged using overlapping frames. Results are shown in Figures 11 and 12.

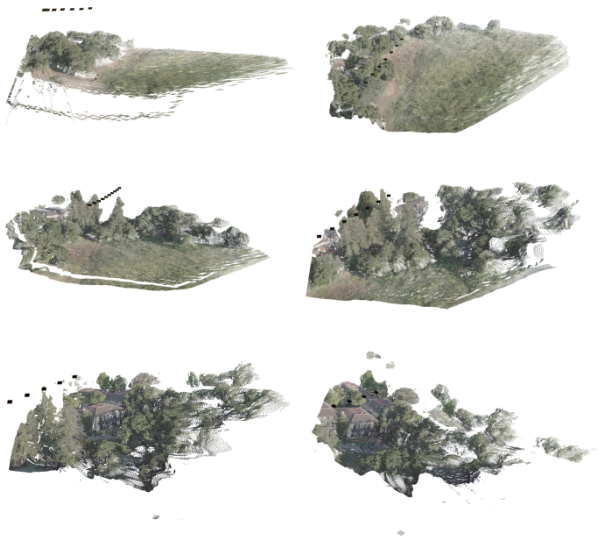

Figure 11. Per-chunk 3D reconstructions of a linear flight path.

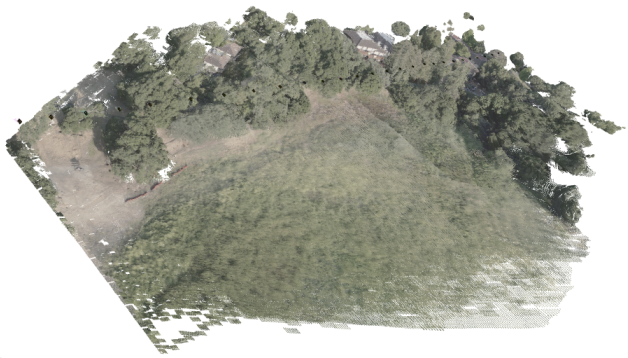

Figure 12. Merged 3D reconstruction from six chunks, stitched using shared camera poses.

5. Discussion

Our experimental results support the hypothesis that chunk-based 3D reconstruction provides a scalable and efficient alternative to traditional monolithic approaches like MASt3R's "logwin" mode. By decoupling the reconstruction problem into smaller, coherent subsets of frames and applying dense stereo methods locally, we drastically reduce memory usage, improve runtime, and enable parallelism across machines—without significant loss in accuracy.

An important insight from our work is that temporal overlap is a reliable proxy for spatial continuity in stable aerial footage. LightGlue, as a fast and lightweight matching backbone, proved essential for efficiently discovering these temporal chunks without exhaustively computing all pairwise comparisons. While our method currently uses vi-

sual overlap to segment chunks, it opens the door to hybrid strategies that leverage additional priors (e.g., GPS proximity, motion estimation, or learned similarity) for more general-purpose chunking in the future.

We also found that merging reconstructions from different chunks—while practical and effective in many cases—introduces slight stitching artifacts, especially when camera motion is unstable or terrain varies significantly in elevation. This limitation suggests the need for future global optimization techniques, such as bundle adjustment across chunk boundaries or similarity transform refinement using both visual and geometric cues.

Additionally, while our pipeline is currently optimized for point cloud outputs, the same chunk-based decomposition could be used as preprocessing for more advanced 3D representations such as Gaussian Splatting or neural radiance fields (NeRFs). We believe this modularity is a strength of the approach: it decouples scene partitioning and reconstruction from downstream rendering or visualization techniques. Gaussian Splatting, however, would require significant new contributions to be able to "stitch" multiple scenes together.

Finally, our current evaluation was conducted on footage collected under relatively stable conditions—forward or downward-facing drone paths with consistent altitude and camera parameters. Applying our system to more chaotic or FPV-style videos remains a challenge due to erratic motion and lack of consistent visual overlap, which can disrupt both chunk discovery and reconstruction. Addressing these failure cases would require more sophisticated motion estimation, dynamic frame selection, or learned models robust to viewpoint discontinuities.

Overall, our system demonstrates strong empirical performance and lays the groundwork for future work in scalable, real-time 3D mapping from aerial video.

6. Conclusion

We propose a scalable, modular framework for reconstructing large-scale 3D scenes from long drone videos. By partitioning data into geometrically coherent chunks and reconstructing each chunk independently with MASt3R, we achieve near-linear runtime and memory scaling. Furthermore, unlike the "logwin" mode in MASt3R, this approach is highly parallelizable.

Our method enables efficient processing of large-scale, high-resolution aerial datasets using only RGB frames. Reconstruction quality is preserved while drastically improving runtime and modularity. We show that even for relatively small datasets, our chunk-based strategy achieves comparable accuracy with substantial efficiency gains.

Limitations. Our pipeline assumes smooth camera motion and continuous visual overlap. It performs best on stable cinematic drone footage. Aggressive FPV footage with flips, sudden turns, or sky-dominated frames may result in failed matching and incoherent chunks. Similarly, stitching relies on shared frames and can degrade with large baselines or inconsistent intrinsics.

Future work. One direction is to incorporate GPS, IMU, and altitude priors to guide chunk merging. These priors could enable clustering chunks by spatial proximity without relying solely on overlapping frames. Camera poses could be coarsely initialized using telemetry, and chunks could be projected onto the ground plane to identify overlaps. Another direction is to perform global bundle adjustment across all merged chunks to improve alignment accuracy. We also plan to explore learning-based pose alignment methods and the integration of real-time chunking on embedded systems.

Moreover, chunking and stitching could both be further accelerated via divide-and-conquer parallelism. Since chunks are processed independently, they can be distributed across cores or GPUs in a tree-based fashion. Similarly, the stitching step—currently sequential—could be implemented as a parallel reduction over chunk pairs, allowing end-to-end distributed reconstruction at even larger scales.

Acknowledgements

We gratefully acknowledge the open-source implementations that served as the foundation for this work: MASt3R [1] and LightGlue [2]. Our pipeline uses adapted versions of these models for chunk-wise dense stereo and feature matching.

References

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *arXiv preprint arXiv:2308.04079*, 2023.

[2] V. Leroy, Y. Cabon, and J. Revaud. Grounding image matching in 3d with mast3r. *arXiv preprint arXiv:2406.09756*, 2024.

[3] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys. Lightglue: Local feature matching at light speed. *arXiv preprint arXiv:2306.13643*, 2023.

[4] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[5] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020.

[1] https://github.com/naver/mast3r

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

[6] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, 2021.

[8] J. Tang, Y. Gao, D. Yang, L. Yan, Y. Yue, and Y. Yang. Dronesplat: 3d gaussian splatting for robust 3d reconstruction from in-the-wild drone imagery. *arXiv preprint arXiv:2503.16964*, 2024.

[9] P. Toffanin. Opensplat. GitHub, 2023. https://github.com/pierotofy/OpenSplat.

[10] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. DUSt3R: Geometric 3d vision made easy. *arXiv preprint arXiv:2312.14132*, 2023.