VROOM - Visual Reconstruction over Onboard Multiview

Yajat Yadav
yajatyadav@berkeley.edu

Varun Bharadwaj
varunbharadwaj@berkeley.edu

Jathin Korrapati
jkorr@berkeley.edu

Tanish Baranwal
tekotan@berkeley.edu

Abstract

We introduce VROOM, a system for reconstructing 3D models of Formula 1 circuits using only onboard camera footage from racecars. Leveraging video data from the 2023 Monaco Grand Prix, we address challenges in the videos such as high-speed motion, sharp turns of perspectives from a singular camera perspective. Our pipeline utilizes different methods such as DROID—Slam, AnyCam, and Monst3r and combine preprocessing techniques such as different methods of masking, temporal chunking, and resolution scaling to account for dynamic motion and computational constraints. We show Vroom is able to partially recover the track and vehicle trajectories in complex environments. These findings indicate the feasibility of using onboard video for scalable 4D reconstruction across multiple agents in real-world settings.

1. Introduction

1.1. F1

In this project we aim to design an algorithm to reconstruct a Formula 1 circuit from the onboard cameras of the racecars. The dataset that we are using is the onboard of a race with 20 cars racing around a 2.074-mile-long circuit in the principality of Monaco.

The main challenges we tackle using Formula 1 video data is high speed 3-D reconstruction and 3-D reconstruction from multiple views. Formula 1 cars are known for their high cornering speed, which makes the problem of 3-D reconstruction difficult because the cars are moving quickly through high detail twists in the circuit.

The specific reason we chose to do the Monaco Formula 1 track is because of the small distance between the cars and the walls. This gives higher quality optical flow because of the closer distance between the moving objects and the camera. Furthermore, the Monaco track has a lot of tighter twists and turns, especially in the second sector s (turns 5-12), really testing how well the model can learn the 3-D reconstruction from the images.

Figure 1. Example onboard video

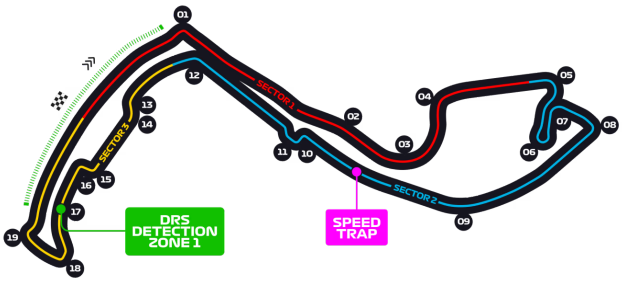


Figure 2. Monaco Track Layout

1.2. SLAM

Simultaneous Localization and Mapping (SLAM) is a key part in many robotic and autonomous systems, which requires the system to keep learn both its surroundings, and where it is within its environment. In our project, we try and calculate the track layout as well as the location of the car on the track. We are trying to calculate the racing line, the path that the car takes as it makes it way through the track. As we try and learn the camera extrinsics, we are learning

the racing line that the cars are the fastest line to take in order to minimize distance travelled and maximize speed.

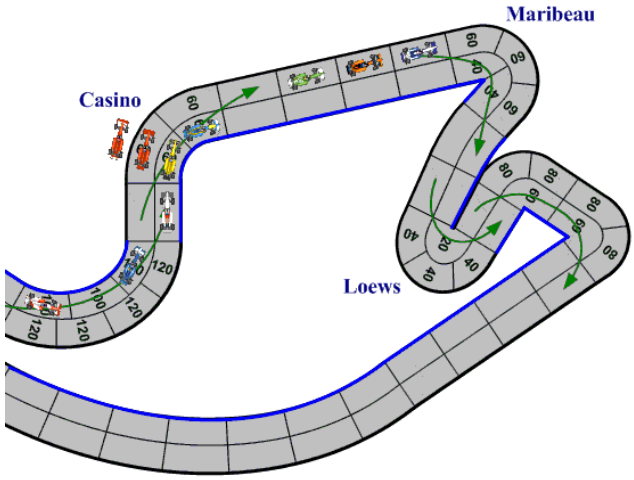


Figure 3. Racing Line

1.3. 3D Reconstruction

Along with SLAM, we perform a 3d reconstruction in order to create a dense 3d point cloud of the entire track. To do this we combine the depth maps + estimated position-wise point cloud from the SLAM outputs along with the estimated camera extrinsics. This dense 3d reconstruction can be combined with the estimated point clouds from all other onboard cameras to develop a 3d reconstruction of all 20 cars and the track through time.

2. Related Work

Traditional SLAM and Structure-from-Motion (SfM) pipelines have provided reliable camera tracking and 3D scene reconstruction for decades, leveraging geometric priors and hand-engineered features. However, the reliance on well-textured scenes, static environments, and calibrated cameras limits their performance in unconstrained scenarios. Recent years have witnessed a transition toward learning-based SLAM systems that offer improved robustness, generalization, and scalability. We survey this evolution, with a focus on monocular and multi-camera SLAM pipelines, culminating in the latest dynamic-scene and camera-agnostic models.

2.1. Classical and Early Learning-Based SLAM.

Foundational systems like ORB-SLAM [1, 6] and LSD-SLAM [3] popularized keyframe-based mapping, bundle adjustment, and direct methods. ORB-SLAM3 extended this pipeline to support multi-camera and visual-inertial setups with remarkable accuracy. In parallel, learning-based methods such as PoseNet [4] and SfM-Learner [11] introduced direct pose and depth regression from images using deep networks, enabling unsupervised learning from video via view synthesis. However, early learned approaches struggled with scale ambiguity, drift, and generalization.

2.2. Differentiable SLAM.

A breakthrough came with DROID-SLAM [9], which demonstrated that classical SLAM principles could be embedded in an end-to-end differentiable framework. DROID-SLAM introduces a dense, differentiable bundle adjustment layer that iteratively refines pose and depth via learned updates. Its recurrent optimization backbone, inspired by optical flow methods like RAFT [8], enabled robust SLAM performance across monocular, stereo, and RGB-D modalities without retraining. DROID-SLAM outperformed traditional pipelines on static benchmarks such as TUM RGB-D and TartanAir, establishing a new standard for learned SLAM.

2.3. Dynamic Scene SLAM.

While DROID-SLAM is robust in static environments, its assumptions break down in the presence of dynamic objects or low-parallax motion. MegaSaM [5] addresses these limitations by introducing dynamic-scene modeling within the SLAM loop. It employs monocular depth priors, per-pixel motion probability masks, and a learned curriculum that transitions from static to dynamic training. Furthermore, it jointly optimizes intrinsic camera parameters at inference time, allowing for uncalibrated, casually captured video. MegaSaM demonstrates state-of-the-art pose and depth estimation on challenging dynamic datasets, outperforming optimization-heavy pipelines like Casual-SfM.

2.4. Feed-Forward 4D Reconstruction.

Complementary to optimization-based methods, MonST3R [10] proposes a feed-forward pipeline that directly estimates 3D pointmaps per video frame, adapting the static-scene architecture DUSt3R [2] to dynamic settings. Rather than explicitly modeling object motion, MonST3R predicts per-frame 3D structure aligned in a common coordinate system, with temporal consistency enforced via a lightweight global alignment step. Trained on a modest corpus of dynamic scenes, MonST3R offers a compelling trade-off between simplicity, speed, and robustness, producing temporally coherent 3D reconstructions even under object motion.

2.5. Generalizing Across Camera Models.

Most SLAM pipelines assume calibrated pinhole cameras, limiting deployment in real-world applications with wide-FOV or unknown intrinsics. UniK3D [7] tackles this by learning a spherical 3D representation coupled with

a camera-agnostic ray encoding via spherical harmonics. This disentangles geometry prediction from camera intrinsics, enabling monocular 3D reconstruction across fisheye, panoramic, and perspective lenses without calibration. While not a full SLAM system, UniK3D's universal camera model offers an important building block for future multi-camera SLAM and mapping frameworks.

3. Methods

3.1. Data

Using F1 TV, we first obtained 20 onboard videos for each driver from the 2023 Monaco Grand Prix race. We decided to focus on the Monaco track for it's high fidelity features, like the changing background buildings, as well as varied turns and elevations in the track. This track would serve as a robust benchmark ensuring our method can succesfully reconstruct F1 tracks as well as the cars' motion in general.

3.2. Preprocessing

3.2.1 Downsampling

The original video data consisted of 1280x720, 50 FPS video, with each about 80 seconds long. As we were limited by compute and hoped to iterate on our method faster, we downsampled the final resolution to 512x144, as well as lowered the FPS to 24 FPS. By testing against several, short 5 second chunks of the original video, we found this downsampling of resolution and FPS had marginal imapct on the final reconstruction quality.

3.2.2 Masking

The onboarding camera videos, as seen in Figure 1 were what we initially used as data input with no changes. However, the car remains stationary throughout the entire race with respect to the camera, and this interfered with all the data-based SLAM methods that we tried. A common issue was that the learned 3D reconstruction would end up having a red "cylinder" throughout the track looking like the car.

To counteract this, we implemented a masking technique that would block a portion of the frame (i.e. the top 30% or top 50%). The intuition was removing the stationary car from the original input data sequence might help reconstruct the motion better relative to the F1 track which would improve results.

We also experimented with other masking methods, such as masking just the car (Figure 4), carving out specific masks, or masking higher/lower fraction of the frame, and found that masking out the bottom half of the frame (Figure 5) performed the best. Once we discovered this, we added this masking to our list of preprocessing methods.

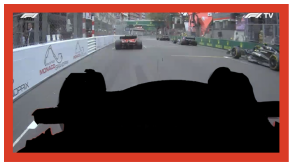


Figure 4. Masking Car

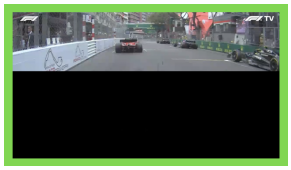


Figure 5. Masking Bottom Half

3.2.3 Video Chunking

Finally, because we were limited by the compute, we had to tradeoff feeding longer videos but at lower FPS into the various methods we tried. With experimentation, we found that lowering the FPS too much (lower than 12) would lead to terrible reconstruction, likely because the cars' motion was extremely fast in the world. Thus, we kept our 24 FPS video but chunked it into overlapping segments. After some experimentation, we settled on about 5 second chunks, with 1 frame of overlap. So, our final pipeline consisted of processing these video chunks individually and then combining each of the learned camera motions and point clouds into one (described in section 3.5).

3.2.4 Smarter Chunking

While the naive chunking approach above (just taking 5 seconds at a time) did improve the quality of our reconstruction, the point cloud would often start to deterioate on the turns. Upon analyzing the outputs, we found this to be the issue when a turn would get cut off and be present in two contiguous chunks. Thus, informed with this, we revised our chunking algorithm to only split the video on straight segments. This way, we ensured that each turn was fed into our method with enough context so the 3D reconstruction could accurately capture the curvature of the turn.

3.3. Race Reconstruction

With our processed data, we tried a few different methods from the literature to obtain both 1) 3D reconstructions of the F1 track, and 2) the camera motion in the track. In this project, we mainly focused on getting these methods to work using one car's onboard video; combining multiple cars' viewpoints to further refine the 3D reconstruction and car movements is a future direction that we hope to explore.

3.3.1 Droid-SLAM

The first method we tried to adapt to was Droid-SLAM. Droid-SLAM estimates camera poses and scene geometry from videos and utilizes the principles of feature extraction with a CNN and uses recurrent bundle adjustment to optimize on the data. We thought using Droid-SLAM would help us in estimating the relative motion of the frame to

generate the trajectory of the car to rebuild the track. Our intention was to first use Droid-SLAM as a baseline method to see how well we could generate the reconstruction based on our data. However, we found that due to our compute not having GUI support it was difficult to visualize reconstruction, even for the examples given in the official repository. Droid-SLAM from the limited examples we have also does not reproduce motion, but just the static elements of the track. Before figuring out the visualization problem, we got AnyCam and Monst3r working, so we ditched our efforts with Droid-SLAM.

3.3.2 AnyCAM

The next method we focused on adapting for our problem was AnyCam. AnyCam employs a transformer-based network to output per-frame camera poses and intrinsics. Being trained on large-scale real-world camera movements and being much faster at test-time than iterative SLAM methods, AnyCam seemed like a promising candidate.

3.3.3 Monst3r

The best results we got were using Monst3r, a library developed to estimate the geometry of scenes with motion. Monst3r is an addition to Dust3r that computes time-indexed point cloud and camera motion given a video as an input. The main issues we faced while using Monst3r was its high memory usage. We were unable to load the entire video (even at 1fps) using a NVIDIA H200 132GB VRAM GPU, and saw subpar results with a longer video but too low of a FPS (Figure 6). In order to fix this, we had to apply the masking, downsampling, and chunking described in the preprocessing section.

After using Monst3r to process each chunk of video, we then utilized the overlap between chunks in oder to transform one chunk's camera extrinsics into another camera's reference frame. By repeating this process repeatdely with all chunks, we were able to "stitch" together the trajectories and obtain a single camera trajectory. Furthermore, we adjusted the point-cloud rendering to use these transformed extrinsics to reconstruct the entire 3D scene for the F1 track. As desired, the end result was a 3D model for the entire track, as well as the car's trajectory.

We next aimed to address the problem of "closing the loop", ie ensuring that combining together each chunk's reconstruction still leads to a global reconstruction consistent with the real world. To accomplish this, we began extending Monst3r to instead process the video in chunks, and then perform bundle adjustment across chunks in order to prevent each reconstruction from slightly drifting off. Since monst3r builds a graph of frames and then performs bundle adjustment with pairs of frames with an edge between them, we modified monst3r to instead do the following: for each chunk, we add edges between frames in the chunk, perform bundle adjustment to refine the chunk's reconstruction. Then, we remove these edges and randomly add edges between frames in this chunk and frames in other chunks, and then perform bundle adjustment with these pairs of frames. The idea with this approach is to alternate between refining the local, chunk-level reconstruction, and refining the reconstruction's alignment with the global-level reconstruction. By repeating this alternating process multiple times for each chunk, we hoped that the "drift-off error" we were noticing happening between distant chunks' reconstructions would be mitigated. However, we were unable to finish up this modification and thoroughly test this in time, and this would be an interesting future direction to ensure our chunk-wise 3D reconstruction approach produces a globally coherent reconstruction.

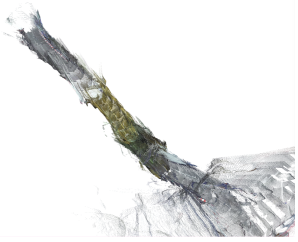


Figure 6. Poor Track reconstruction with 3 FPS

4. Results

4.1. AnyCAM

We found that AnyCAM was not a robust SLAM solution in practice. It performed poorly not only on our F1 dataset, but also on dashcam-style footage resembling the examples presented in the original paper. Despite extensive sanity checks, including testing on clean, front-facing driving sequences, AnyCAM consistently failed to recover plausible camera trajectories. These failures suggest that the method may be sensitive to scene content or initialization, and raise concerns about the reproducibility of its reported performance. Figures 7 and 8 illustrate a significant mismatch between AnyCAM's predicted camera motion and the ground truth trajectory, particularly around turns.

4.2. Monst3r

Using the preprocessing and the chunking that we described, we were able to get the final circuit map that we have shown in figure 9. As is clear, the track does not loop back around, and there is still significant work to be done in order to "close the loop". However, each individual segment's reconstruction was very close to the ground truth map, as seen in Figure 10 for turns 3, 4, and 5.

Figure 7. Predicted Camera Motion by AnyCam

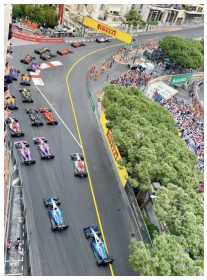


Figure 8. Ground Truth Turn

Figure 9. Full Track Reconstruction using Monst3r

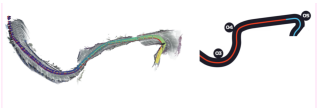


Figure 10. Turns 3/4/5 Prediction and Ground Truth

5. Conclusion

Our outputs from Monst3r show significant progress towards creating a 3d reconstruction of the first lap of the 2023 Formula 1 Monaca Grand Prix. However, there is still signficant progress to be made to fully fix the outputs.

5.1. Future Improvements

The main improvements that we were working on was to make the global reconstruction better. The main thing we were trying to do was figure out an efficient keyframe sampling method that we could then use to do global bundle adjustment. We began implementing a 2 stage process where we loop between intrachunk alignment, global bundle adjustment with keyframes only (due to Monst3r's high memory usage), followed by more iterations of intrachunk alignment and global adjustment.

Furthermore, we have onboard cameras from more than just 1 camera. If we can extend this method to track the poses and learn the point clouds using multiple views. Doing this would allow us to generate a reconstruction of not just the track but also the 20 cars. Furthermore, if we had multiple views we could get rid of the masking as other views would be able to tell that the car was not a static object on the track. This would also be valuable as no masking would mean we get to use the high-quality optical flow information of the ground right next to the car to get stronger estimates of the cameras' trajectories.

References

[1] Carlos Campos, Rafael Elvira, Juan J Gómez Rodríguez, JMM Montiel, and JD Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 2

[2] Qianqian Chen, Yida Wang, and Yebin Zhang. Dust3r: Dual-scale transformer for sparse-to-dense matching. In *CVPR*, 2024. 2

[3] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, pages 834–849, 2014. 2

[4] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, pages 2938–2946, 2015. 2

[5] Jiarong Li, Tianyuan Shen, Xiaowei Zhou, and Marc Pollefeys. Megasam: Accurate, fast and robust structure and motion from casual dynamic videos. In *CVPR*, 2024. to appear. 2

[6] Raul Mur-Artal, JMM Montiel, and JD Tardos. Orb-slam: A versatile and accurate monocular slam system. In *IEEE Transactions on Robotics*, volume 31, pages 1147–1163, 2015. 2

[7] Nicola Piccinelli, Gabriel Baatz, and Arno Knapitsch. Unik3d: Universal camera monocular 3d estimation. *arXiv preprint arXiv:2401.06125*, 2025. 2

[8] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419, 2020. 2

[9] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. In *NeurIPS*, 2021. 2

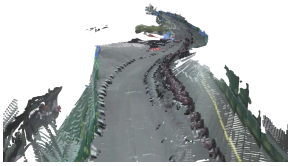
[10] Yida Wang, Qianqian Chen, Wei Chen, Yu Zhang, and Yebin Zhang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arXiv:2403.04974*, 2024. 2

[11] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, pages 1851–1858, 2017. 2

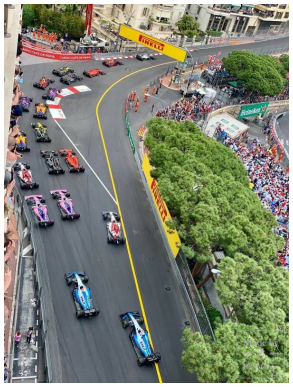
A. Additional Figures

Turn 8 Reconstruction

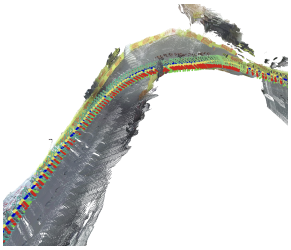
Turn 8 Ground Truth



Turn 1 Reconstruction

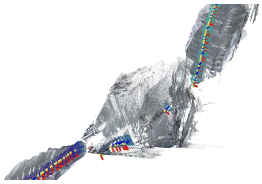


Turn 1 Ground Truth

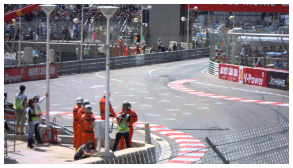


Turn 18 Reconstruction

Turn 18 Ground Truth



Failure Case: Turn 15+16 Reconstruction



Turn 15+16 Ground Truth