# Visual-Inertial Odometry on the MIT Racecars

Alex Kashi Naomi Schurr

***Abstract*—Visual-inertial odometry (VIO) allows a robot to estimate its state, including scale, as it navigates an environment. In this work, we document considerations and results of installing VINS-Fusion, an open-source implementation of VIO, onto the MIT Racecar mobile platform, using a Jetson TX2 Developer kit and a RealSense D455 stereo camera with an inertial measurement unit.**

***Index Terms*—VIO, MIT Racecar, VINS-Fusion, VINS-Fusion-gpu, Voxels, VoxBlox, Mesh**

## I. Introduction

Enabling visual-inertial odometry (VIO) to run onboard an MIT Racecar is an appealing prospect to allow the robot to execute pre-planned trajectories without needing a map or relying on open-loop motor commands.

Like many small autonomous vehicles, the MIT Racecar has limited onboard processing and storage space. In this work, we describe how we manage the available hardware and software resources to achieve VIO waypoint-following on the MIT Racecar and also run loop closure detection and basic mapping functionalities. The three setups we present are 1) VIO on TX2 CPU only, 2) VIO and loop closure on TX2 CPU and GPU, and 3) VIO, loop closure, and mesh building using the TX2 and a laptop base station.

## II. Components and Architecture

### A. Hardware

The relevant hardware for this project includes the RealSense D455 camera and the MIT Racecar platform, including a Jetson TX2, for all setups. Setups 2 and 3 additionally use an external SSD drive, while Setup 3 additionally uses a Velodyne VLP-16 LiDAR and a laptop as a base station.

*1) RealSense D455:* The RealSense D455 includes a pair of infra (depth) cameras and an IMU, which were our primary sensors for VIO. The IR emitter was turned off to avoid interfering with feature detection and tracking.

*2) The MIT Racecar:* The MIT Racecar[1] is a mobile robotics platform designed for robotics education and research. Racecars 37 and 85, the two vehicles used in this work, are each equipped with a Jetson TX2 developer kit, with CPU and GPU, and 32 GB of storage. The racecars have both teleoperated and autonomous driving capabilities, and Racecar 85 has a velodyne LiDAR which we utilized for mapping.

### B. Software

The relevant software for this project includes the RealSense driver software, VINS-Fusion, VINS-Fusion-gpu, Voxblox, and a rudimentary waypoint follower.

*1) librealsense and RealSense Viewer:* The librealsense library has a version[2] for general Linux and a separate version[3] specifically for NVIDIA Jetson devices. We also performed IMU calibration with the provided `rs-imu-calibration.py` script and installed the ROS wrapper[4] for RealSense devices.

The intrinsic parameters for supported camera resolutions are available through the RealSense Viewer GUI, and CAD-based extrinsic parameters can be obtained by running `rs-enumerate-devices -c` with librealsense installed. We considered using a separate calibration package such as kalibr[5], but the published parameters appeared to be sufficient.

*2) VINS-Fusion:* VINS-Fusion[6] is an open-source implementation of VIO based on VINS-Mono [1] and adapted to different sensor configurations. We chose to use stereo cameras plus an IMU and specified their intrinsic and extrinsic parameters in associated configuration files. VINS-Fusion was installed directly on the TX2 for the CPU-only setup.

*3) VINS-Fusion-gpu:* To free CPU for other tasks, we installed VINS-Fusion-gpu[7] by following a tutorial[8] specifically for NVIDIA Jetson devices. VINS-Fusion-gpu was installed on an EXT4-formatted external SSD.

*4) voxblox:* Voxblox[9] was used to represent a volumetric map for our third setup. While installing VINS-Fusion-gpu was intended to open up CPU processing for voxblox, it was ultimately run on a laptop due to computational demands.

*5) Waypoint Follower:* Based on the "parking controller" in the MIT 6.141 Visual Servoing Lab[10], a `waypoint_follower` node was added to listen to odometry messages from the `vins_estimator`, determine when a waypoint was reached, and publish the appropriate next target.

## III. System Architecture

In our pipeline for VIO using the CPU only (Setup 1), the sensors were processed by the RealSense node to provide image and IMU streams to VINS-Fusion, which outputs a pose that is used by the Waypoint Follower to generate a path. We found that VINS-Fusion on the TX2 CPU was not able to process the sensor inputs in realtime with the default camera

[1] https://racecar.mit.edu/

[2] https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution_linux.md

[3] https://dev.intelrealsense.com/docs/nvidia-jetson-tx2-installation

[4] https://github.com/IntelRealSense/realsense-ros/tree/development

[5] https://github.com/ethz-asl/kalibr

[6] https://github.com/HKUST-Aerial-Robotics/VINS-Fusion

[7] https://github.com/pjrambo/VINS-Fusion-gpu

[8] https://github.com/arjunskumar/vins-fusion-gpu-tx2-nano

[9] https://github.com/ethz-asl/voxblox

[10] https://github.com/mit-rss/visual_servoing

resolution and frame rate, so for our experiments using only the CPU, we reduced the camera resolution to 424x240 pixels.

Our GPU-enabled VIO architecture (Setup 2) was similar, with VINS-Fusion-gpu and Loop-Fusion-gpu replacing the CPU version of VINS-Fusion. GPU acceleration supported resolutions of up to 848x480 pixels at 15FPS in real time.

The architecture for the mapping pipeline (Setup 3) is shown in Figure 1. The pose estimate and Velodyne point cloud are sent to the base station, which computes and renders the voxel mesh in realtime. A mesh generated from the the RealSense stereo cam would have be preferred, but point cloud generation on the TX2 using the RealSense D455 was unstable.

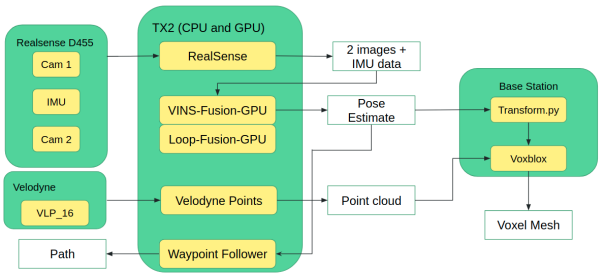


Fig. 1: Architecture for Setup 3, VIO and mapping

## IV. Results

Our VIO experiments were conducted in the common area of the 2nd floor of MIT Building 31. First, a path estimate generated in realtime on CPU is shown in Figure 2. Overlaying the path on the floor plan shows that the estimate is visually to scale. Second, a loop closure detected using GPU-enabled VINS- and Loop-Fusion is shown in Figure 3. Finally, a mesh generated by the base station is shown in Figure 4.

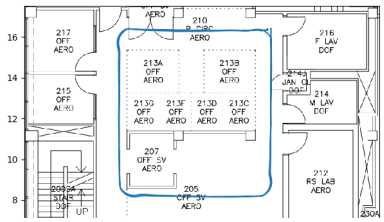

Fig. 2: Waypoint Following with VIO on TX2 CPU (Setup 1). The VINS-Fusion state estimate is aligned to the start position on the test space floor plan, showing true scale.

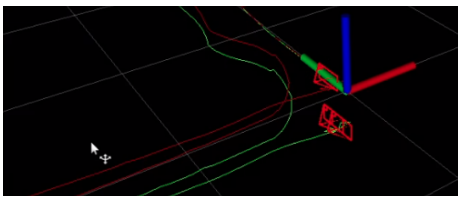

Fig. 3: Loop Closures with GPU-Enabled VIO on TX2 (Setup 2). Raw odometry in green, loop-adjusted estimate in red.

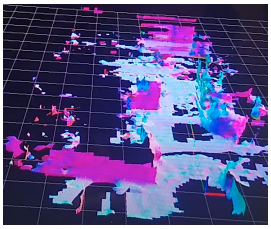

Fig. 4: Mapping on a Base Station with GPU-Enabled VIO on TX2 (Setup 3). Mesh generated with voxblox using the velodyne and single-pass VIO estimate. This mesh corresponds to the top segment of the path in Figure 2.

Results of performing a tighter path consisting of 3 loops with VINS-Fusion-gpu (Setup 2) are shown in Figure 5, with OptiTrack paths provided for comparison. The paths are visually consistent, but the 1 meter end pose error in the third case may be problematic in highly constrained environments.

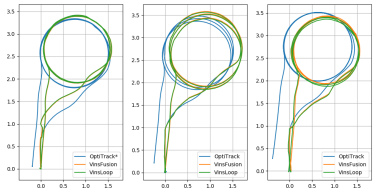

Fig. 5: Three passes running VIO on the same waypoint list. The Racecar's trajectory as captured by an OptiTrack system is shown in blue (manually aligned for the first figure only).

## V. Discussion and Areas for Future Work

In this work, we found that running GPU-enabled VINS-Fusion on a TX2 with the published parameters of the RealSense D455 in conjunction with 640x480px images at 15fps was precise enough for the robot to execute a 60m path including 7 90-degree turns and return to a start location through a standard hallway environment. The OptiTrack experiments showed that the estimate may not be precise enough for tight spaces or difficult paths. Additional experiments could characterize the performance under different conditions, including speeds higher than the 0.5 m/s we used for safety.

This work used a RealSense D455 to simplify the coordination of IMU and camera data. Using the Racecar's IMU and ZED cameras may be interesting given the extrinsics optimization included with VINS-Fusion, though new synchronization issues may arise. A logical extension of our work would be to use the output of Voxblox to inform a more sophisticated path planner that could map and navigate unknown environments and react to its surroundings.


## Acknowledgment

We thank Mubarik Mohamoud, Nathan Hughes, and Dominic Maggio for tips on the racecars, RealSense, and VINS-Fusion GPU, respectively. Thanks to Jingnan Shi and Ziqi Lu for guidance during project check-ins, and to Professor Luca Carlone for an exciting class and access to the racecars.

## REFERENCES

[1] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," IEEE TRANSACTIONS ON ROBOTICS, vol. 34, no. 4 pp. 1004-1020, August 2018.