Technical Report 138

Cooperative Mapping for Automated Vehicles

Research Supervisor:
Todd Humphreys
Wireless Networking and Communications Group

October 2017

Data-Supported Transportation Operations & Planning Center (D-STOP)

A Tier 1 USDOT University Transportation Center at The University of Texas at Austin

D-STOP is a collaborative initiative by researchers at the Center for Transportation Research and the Wireless Networking and Communications Group at The University of Texas at Austin.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
D-STOP/2017/138		

4. Title and Subtitle	5. Report Date
Cooperative Mapping for Automated Vehicles	October 2017
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Todd Humphreys, Matthew Murrian, Lakshay Narula, and Michael Wooten	Report 138

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Data-Supported Transportation Operations & Planning Center (D-STOP) The University of Texas at Austin 1616 Guadalupe Street, Suite 4.202 Austin, Texas 78701	11. Contract or Grant No. DTRT13-G-UTC58

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Data-Supported Transportation Operations & Planning Center (D-STOP) The University of Texas at Austin 1616 Guadalupe Street, Suite 4.202 Austin, Texas 78701	14. Sponsoring Agency Code

15. Supplementary Notes

Supported by a grant from the U.S. Department of Transportation, University Transportation Centers Program.

16. Abstract

Localization is essential for automated vehicles, even for simple tasks such as lanekeeping. Some automated vehicle systems use their sensors to perceive their surroundings on-the-fly, such as the early variants of the Tesla Autopilot, while others such as the Waymo car navigate within a prior map. The latter approach is beneficial in that it helps the system to expect the expected, that is, it relieves the system of perceiving static features. However, making and updating such accurate prior maps using a specialized vehicle fleet is expensive and cumbersome. Techniques for Simultaneous Localization And Mapping (SLAM) are an attractive solution to this problem. SLAM uses visual and other sensors for creating and updating maps as the robot/vehicle navigates within the map. This project explores the possibility of using multiple vehicles to perform cooperative SLAM for improving and updating the map formed using optical cameras, radar, IMU, and GNSS. It is assumed that raw data from these sensors can be shared among the vehicles over a wireless link either via V2V or V2I communications.

17. Key Words	18. Distribution Statement
V2I, V2V, lanekeeping, AV, automated vehicle	No restrictions. This document is available to the public through NTIS (http://www.ntis.gov): National Technical Information Service 5285 Port Royal Road Springfield, Virginia 22161

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	XX	

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

Acknowledgements

The authors recognize that support for this research was provided by a grant from the U.S. Department of Transportation, University Transportation Centers.

.

COOPERATIVE MAPPING FOR AUTOMATED VEHICLES
D-STOP REPORT – 05 OCT 2017

TODD HUMPHREYS, MATTHEW MURRIAN, LAKSHAY NARULA, AND MICHAEL WOOTEN

Abstract. Localization is essential for automated vehicles, even for simple tasks such as lane-keeping. Some automated vehicle systems use their sensors to perceive their surroundings on-the-fly, such as the early variants of the Tesla Autopilot, while others such as the Waymo car navigate within a prior map. The latter approach is beneficial in that it helps the system to *expect the expected*, that is, it relieves the system of perceiving static features. However, making and updating such accurate prior maps using a specialized vehicle fleet is expensive and cumbersome. Techniques for Simulatneous Localization And Mapping (SLAM) are an attractive solution to this problem. SLAM uses visual and other sensors for creating and updating maps as the robot/vehicle navigates within the map. This project explores the possibility of using multiple vehicles to perform cooperative SLAM for improving and updating the map formed using optical cameras, radar, IMU, and GNSS. It is assumed that raw data from these sensors can be shared among the vehicles over a wireless link either via V2V or V2I communications.

1. Introduction

All major automakers today are engrossed in the development and integration of software and sensors that enable automated vehicles. Localization within a map is one of the primary operations that automated vehicles must perform, either to navigate from one location to another, or, more interestingly, to interact with their surroundings within a mapped environment. Prior high-definition digital maps allow the vehicle to *expect the expected*, that is, they relieve the system of the need to classify static features.

Satellite-based navigation sensors have historically been the unrivalled sensor of choice for localization. However, the high-reliability decimeter-level accuracy demanded by automated vehicles for lane-keeping and other applications has significantly changed this landscape. In fact, in most automated vehicles being developed, the GPS/GNSS system is a secondary sensor whose only role is to loosely constrain (within a few meters) the primary sensor data to a global reference when building a digital map. Other vehicular sensors such as visual cameras and LiDAR are being used as primary systems for vehicle localization within the prior map.

The need for accurate digital maps has spurred dedicated map-making campaigns involving fleets of specialized mapping vehicles. Mapping vehicles typically employ state-of-the-art high-performance sensors that are too expensive to be installed on consumer vehicles. Although these

Date: October 5, 2017.

exquisite maps do enable sub-decimeter-accurate within-map localization, their construction and use comes with important limitations:

(1) In current practice, sub-decimeter vehicle localization within a prior map is critically dependent on optical cameras and LiDAR. LiDAR is known to fail in heavy rain, snow, and fog. Optical cameras are vulnerable to poor lighting conditions and are easily blinded by bright light. Moreover, the previously-mapped roadside environment can be significantly altered by the build up of snow or sand. Thus, even after severe weather subsides, a vision- and LiDAR-dependent system may have difficulty locating the host vehicle within the prior map. Such weather-induced difficulties for within-map localization cannot be dismissed as negligibly rare: many populated regions of the globe are routinely subject to punishing weather.
(2) While mapping using a specialized fleet is feasible for urban cities, it is time-consuming and cumbersome to map, and, more importantly, maintain the maps of entire countries/continents. A key enabler for large-scale up-to-date maps will be enlisting the help of the very vehicles who need the map – consumer vehicles – to build and update the map. However, consumer vehicles will only be equipped with low-cost consumer-grade sensor suites. The performance of such sensors in creating high-precision maps has not been explored.
(3) Automated vehicles would also need accurate information on the position, velocity, and intent of their neighboring vehicles. While this information can be inferred using the sensors on the vehicle, in certain situations, for example at a blind corner or during a left turn manouver, it might be beneficial for the vehicles to communicate this information to each other on a wireless channel. For data exchange between vehicles, position and velocity information must be referenced to a common coordinate frame – preferably a global reference frame. However, the maps commonly generated using optical cameras and LiDAR are, as mentioned earlier, only loosely tied to the global frame (e.g., WGS-84), with an exact correspondence that differs from provider to provider: Waymo, Uber, and HERE would each assign different coordinates to the same physical object, and these coordinates could differ by a meter or more – far too large a discrepancy for coordinated automated driving.

Unlike optical cameras and LiDAR, GPS/GNSS is agnostic to weather elements, lighting conditions, etc. Thus, it is a natural complement to vision and LiDAR-based sensing. GNSS works in all weather conditions and is globally referenced. Its chief impairments, signal blockage and multipath, mostly occur in urban areas where vision, LiDAR, and radar sensors can aid the mapping and localization solution. Also, as a globally-referenced source of absolute position and velocity, precise GPS/GNSS is very useful for sharing location and velocity data among vehicles, and for calibrating the other sensors. For example, it can be used for calibration of extrinsic and intrinsic camera parameters.

Radar is another sensor that works in all weather conditions. It remains useful for collision avoidance in poor weather, but may not be accurate enough for tight lane-keeping and for locating the vehicle in open areas where roadside features are scarce. Nonetheless, it provides

information that is useful for all-weather navigation. Surprisingly, radar has remained mostly unexplored for the purposes of mapping and localization.

In this project, we explore a sensor fusion scheme using the GNSS standard positioning service (SPS), radar, and visible-light cameras for decimeter-accurate globally-referenced collaborative sparse mapping using low-cost sensors, and for decimeter-accurate globally-referenced localization in the resulting map.

The rest of this report is organized as follows: enabling technologies for collaborative mapping, and related prior work is outlined in Section 2. Section 3 discusses the progress made towards collaborative mapping as a part of this project. Our plan for future steps is detailed in Section 4.

2. Enabling Technologies and Prior Work

This section reviews the most important enabling technologies for solving the problem of collaborative mapping. A brief review of each technology is provided, along with references to prior work that is relevant to the problem at hand.

2.1. **Visual Simultaneous Localization and Mapping (SLAM).**

Visual SLAM is a technique that uses images captured from a visible-light camera as the source of information to estimate the pose of a vehicle or a robot in an environment, while at the same time creating a 3-dimensional representation of the environment from the 2-dimensional images.

Visible-light camera(s) are a part of every autonomous driving vehicle in conception or production. With modest power consumption and at a low cost, these cameras enable some of the most critical operations such as

- *Local motion planning*: steering control, velocity control, etc.
- *Obstacle avoidance*: obstacle detection, semantic segmentation, road detection, etc.
- *Lawful driving*: sign and signal detection and recognition, etc.

Visible-light cameras, in essence, are the eyes of an autonomous vehicle. In fact, some autonomous driving systems use end-to-end machine learning algorithms to directly control the vehicle based primarily on the camera feed [1]. Some autonomous vehicles also employ active sensors such as LiDAR to assist in the above operations. Active sensors can simplify the underlying estimation and mapping stages. Such simplification is achieved, in part, by more complex data acquisition, that is, by recovering dense 3-dimensional point clouds using laser scans. However, an important drawback of such active sensors is that they are expensive at the time of writing, and consume much more power than passive sensors such as visible-light cameras and GPS/GNSS. As a result, sensors such as LiDAR were not considered for solving the SLAM problem in this project.

The problem of visual SLAM has been studied extensively in the computer vision community, and many effective systems have been proposed [2–10]. In order to choose one of these designs, we had to answer the following two questions:

(1) *Sparse or dense?*: Visual SLAM algorithms are usually classified as spare, semi-dense, and dense. Sparse SLAM algorithms [2–4] use only distinctive features such as corners and edges, while dense SLAM [6, 10] algorithms use every pixel in the image. For the purpose of localization, the sparse point cloud generated using sparse SLAM algorithms is sufficient. Dense reconstruction is more appealing to the human eye, but does not provide any tangible benefit to vehicle localization, while consuming much more computational resources. Moreover, it has been shown that there is an evolutionary path to dense 3-dimensional reconstruction once a sparse map has been generated [4]. As a result, we prefer the sparse feature-point-based SLAM in this project.
(2) *Filtering or batch estimation?*: In a famous paper, Strasdat [11] concluded that keyframe bundle adjustment (batch non-linear optimization) outperforms filtering techniques such as the EKF, and gives the most accuracy per unit of computing time. This paper noted that the having a high number of features points per frame improves accuracy more than having a large number of frames. Most recent state-of-the-art visual SLAM algorithms use the bundle adjustment approach in favor of sequential filtering. As a result, we consider bundle adjustment-based non-linear optimization for visual SLAM.

In this project, we use the bundle adjustment technique for nonlinear optimization. Some basics of bundle adjustment are discussed next.

2.1.1. *Bundle Adjustment.* Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. It is a common technique used in photogrammetry and computer vision for Structure from Motion (SfM) and 3-dimensional reconstruction. A tutorial paper on bundle adjustment by Triggs et al. [12] is an excellent reference. Appendix 6 of the Hartley & Zisserman textbook [12] is another good resource on this topic.

Map point 3D locations $\boldsymbol{x}_{\mathcal{S}}^{p_j} \in \mathbb{R}^3$ and keyframe poses $(\boldsymbol{x}_{\mathcal{S}}^{\mathcal{C}_i}, \boldsymbol{q}_{\mathcal{S}}^{\mathcal{C}_i})$, where $\mathcal{S}$ stands for the SLAM frame, p_j is the jth map point, and $\mathcal{C}_i$ is the ith keyframe, are optimized minimizing the reprojection error with respect to the matched keypoints $\boldsymbol{x}_{i,j} \in \mathbb{R}^2$. The error term for the observation of a map point j in a keyframe i is

$$\boldsymbol{e}_{ij} = \boldsymbol{x}_{ij} - \pi_i(\boldsymbol{x}_{\mathcal{S}}^{p_j}, \boldsymbol{x}_{\mathcal{S}}^{\mathcal{C}_i}, \boldsymbol{q}_{\mathcal{S}}^{\mathcal{C}_i}) \tag{1}$$

where π_i is the projection function

$$\pi_i(\boldsymbol{x}_{\mathcal{S}}^{p_j}, \boldsymbol{x}_{\mathcal{S}}^{\mathcal{C}_i}, \boldsymbol{q}_{\mathcal{S}}^{\mathcal{C}_i}) = \begin{bmatrix} f_{iu}\frac{x_{ij}}{z_{ij}} + c_{iu} \\ f_{iv}\frac{y_{ij}}{z_{ij}} + c_{iv} \end{bmatrix} \tag{2}$$

$$[x_{ij}, y_{ij}, z_{ij}]^T = R\left(\boldsymbol{q}_{\mathcal{S}}^{\mathcal{C}_i}\right)^T \left(\boldsymbol{x}_{\mathcal{S}}^{p_j} - \boldsymbol{x}_{\mathcal{S}}^{\mathcal{C}_i}\right) \tag{3}$$

where $R(\cdot)$ denotes the rotation matrix corresponding to the argument, and (f_{iu}, f_{iv}) and (c_{iu}, c_{iv}) are the focal length and principal point associated to camera i. The cost function

to be minimized is

$$C = \sum_{ij} \rho(\boldsymbol{e}_{ij}^T \boldsymbol{\Omega}_{ij}^{-1} \boldsymbol{e}_{ij}) \tag{4}$$

where ρ can be the standard least squares cost function or a more robust cost function like Huber or Tukey, and $\boldsymbol{\Omega}_{ij} = \sigma_{ij}^2 \boldsymbol{I}_{2\times 2}$ is the covariance of the feature measurements.

When solving the minimization problems arising in the framework of bundle adjustment, the normal equations have a sparse block structure owing to the lack of interaction among parameters for different 3D points and cameras. This can be exploited to gain tremendous computational benefits by employing a sparse variant of the Levenberg Marquardt algorithm which explicitly takes advantage of the normal equations zeros pattern, avoiding storing and operating on zero-elements.

2.1.2. *ORB-SLAM2.* ORB-SLAM2 [4] is a feature-based SLAM system that operates in real time, in small and large, indoor and outdoor environments. ORB-SLAM2 supports monocular, stereo, and RGB-D cameras, and has features like map reuse, loop closing, and relocalization. The source code for this system is open-source. This code was used as a basis for visual SLAM development in this project. Some of the important algorithms and terminology related to ORB-SLAM2 are summarized here.

Oriented FAST and rotated BRIEF (ORB) are binary features invariant to rotation and scale (in a certain range), resulting in a very fast recognizer with good invariance to viewpoint. ORB-SLAM2 uses the ORB features for all tasks: tracking, mapping, relocalization, and loop closing. This makes the system more efficient, simple and, reliable.

ORB-SLAM2 employs two major containers to process the visual data: map points and keyframes. Each map point p_j stores

- Its 3D position $\boldsymbol{x}_{\mathcal{S}}^{p_j}$ in the SLAM coordinate system.
- The viewing direction $\boldsymbol{n}_j$, which is the mean unit vector of all its viewing directions (the rays that join the point with the optical center of the keyframes that observe it).
- A representative ORB descriptor $\boldsymbol{D}_j$, which is the associated ORB descriptor whose Hamming distance is minimum with respect to all other associated descriptors in the keyframes in which the point is observed.
- The maximum $d_{\max}$ and minimum $d_{\min}$ distances at which the point can be observed, according to the scale invariance limits of the ORB features.

Each keyframe K_i stores

- The camera pose $(\boldsymbol{x}_{\mathcal{S}}^{\mathcal{C}_i}, \boldsymbol{q}_{\mathcal{S}}^{\mathcal{C}_i})$, which is a rigid body transformation that transforms points from the camera to the SLAM coordinate system.
- The camera intrinsics, including focal length and principal point.
- All the ORB features extracted in the frame, associated or not to a map point, whose coordinates are undistorted if a distortion model is provided.

Map points and keyframes are created with a generous policy, while a later very exigent culling mechanism is in charge of detecting redundant keyframes and wrongly matched or not trackable map points. This permits a flexible map expansion during exploration, which boost tracking robustness under hard conditions (e.g. rotations, fast movements), while its size is bounded in continual revisits to the same environment, i.e. lifelong operation.

The ORB-SLAM2 system runs three threads in parallel: tracking, local mapping and loop closing. The tracking is in charge of localizing the camera with every frame and deciding when to insert a new keyframe. The local mapping processes new keyframes and performs local BA to achieve an optimal reconstruction in the surroundings of the camera pose. The loop closing searches for loops with every new keyframe. If a loop is detected, a similarity transformation informs about the drift accumulated in the loop.

Covisibility information between keyframes is represented as an undirected weighted graph. Each node is a keyframe and an edge between two keyframes exists if they share observations of the same map points (at least 15). To operate in real-time, ORB-SLAM2 performs a *windowed* BA. The local BA optimizes the currently processed keyframe, K_i, all the keyframes connected to it in the covisibility graph, $\mathcal{K}_i$, and all the map points seen by those keyframes. All other keyframes that see those points but are not connected to the currently processed keyframe are included in the optimization but remain fixed.

2.2. **Radar Odometry and SLAM.** Even though radar is now a common sensor on most luxury cars, the utility of radar for localization has not been explored in depth. Schuster et al. [13] have recently proposed radar SLAM with a graph optimization framework that is very similar to visual SLAM. However, they present it as a standalone radar system, and do not integrate it with visual maps.

They use multiple short range radars around the vehicle to make a map of a parking lot. Their approach to feature detection uses time history with odometery to build a "measurement grid" with geographic segmentation and intensity or occupancy values determined by the number of detections within each segment. Finally, as with visual SLAM, a graph optimization library is used. Localization accuracy within 1 meter is reported when the environemnt has not changed very significantly.

Barjenbruch et al. [14] present a radar odometry technique that also shows promising results. They formulate the problem as an optimization, and match the point clouds obtained in consecutive radar scans. Errors of 0.03 m/s in velocity, 0.435 degrees/s in yaw, and 0.37% position error with distance travelled is reported.

In conclusion, most common method of using radar for mapping involves building occupancy grids. These do not scale well for global maps because they segment the world into geographic block which specify the resolution and require present/not present for every block. So if a single bit is used to represent that information on a 1 cm 2D grid, the map will use storage on the order of 3 GB per square mile while, the majority of the grid will simply say "not occupied" (i.e. 0). For example, a map of the city limits of Austin, TX would need 815 GB.

Odometry (or Ego-Motion) based techniques using radar sensors show promise and should fuse well with SLAM techniques because the referenced work takes a similar approach of least squares minimization to determine the translation and rotation of the vehicle with respect to fixed landmarks. This is analogous to feature based tracking to determine translation/rotation of the vehicle with visual cameras. In this case, the radar has a direct measurement of delta position and velocity and an indirect measurement of heading.

There is no literature on fusing visible-light cameras and FMCW radar to aid with map making or augmentation of visual SLAM routines. We believe there is room for contribution here.

3. Project Accomplishments

3.1. **The Sensorium.** In order to collect data for this project, we put together a sensor suite, which we named the Sensorium. The Sensorium can be mounted on any vehicle, and can be powered using a car battery. Figure 1 shows the Sensorium mounted on a Toyota 4Runner.

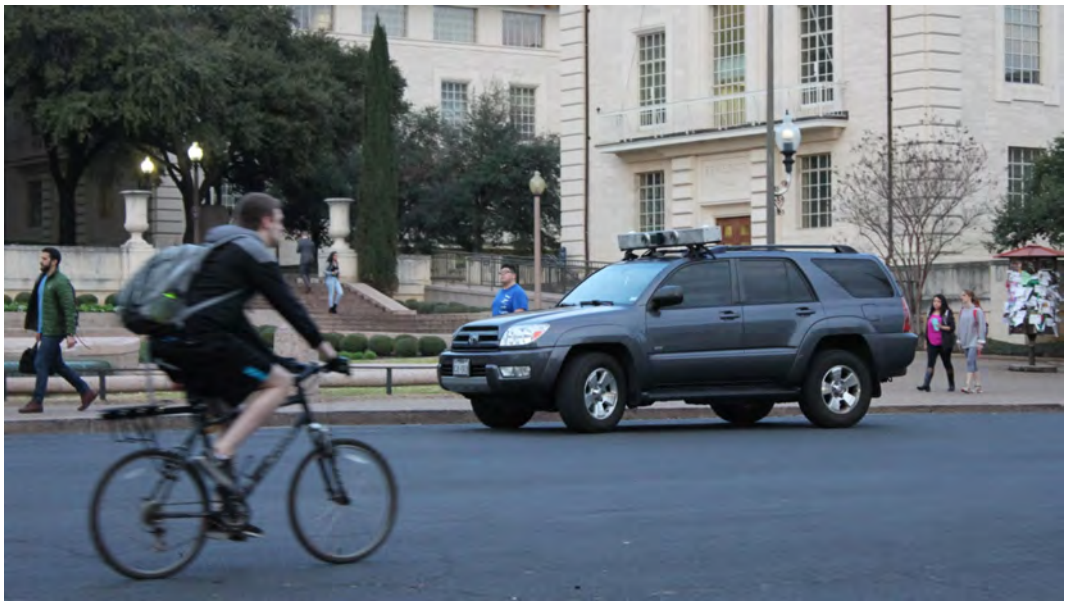


Figure 1. The Sensorium mounted on a Toyota 4Runner.

3.1.1. *Hardware.* The Sensorium is equipped with a stereo camera rig, two dual-frequency GNSS antennas, an automotive radar, and a smartphone-grade IMU. A custom in-house GNSS front-end, called the RadioLynx, samples and pre-conditions the signals received from the GNSS antennas. The RadioLynx also serves the purpose of triggering the cameras, which enables us to time tag the images with GPS time to within a few tens of nanoseconds.

The Sensorium also has an Intel NUC installed to process all the data captured using these sensors. The NUC also runs GRID, our in-house software-defined precise GNSS receiver. The

precise GNSS will be used as a ground truth in the accuracy analysis of the results. Precise GNSS is enabled by the Longhorn Dense Reference Network, a unique reference network that we have put together in Austin TX to provide ionospheric and tropospheric corrections to mobile receivers such as the Sensorium.

Figure 2 summarizes the hardware components of the Sensorium.

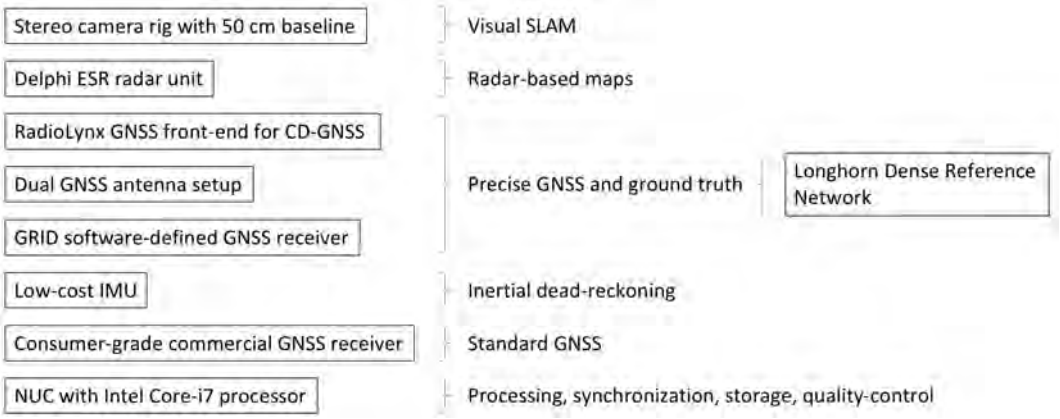


FIGURE 2. Hardware components of the Sensorium.

3.1.2. *Software Architecture.* The architecture of the sensor fusion system is built around the Robotics Operating System (ROS). We make heavy use of the ROS topic architecture to communicate between the different nodes in the sensor fusion system. This enables us to easily add another sensor and/or only use a subset of the existing sensors without making any significant changes to the communication between nodes.

We have also developed a software suite for collating and quality-checking the data produced by the sensors. A radar parser utility parses radar messages from the CAN bus. An image grabber grabs and stores up to 30 fps data from the stereo camera rig and publishes them on a ROS topic. These images can be used for real-time processing or can be stored to a ROS bag. The image grabber also verifies that no frames are being dropped. We also log raw IF GNSS data for offline processing. In addition, the IMU data is also stored. We also have a commercial (ublox) GNSS receiver which is fed the same GNSS signals as the RadioLynx. We also store the output of this receiver.

For a sensor fusion system, it is critical that all data be time-synchronized. To achieve this, we set up an NTP server on the NUC that is driven by the GPS timing output of GRID. In addition, we trigger the cameras directly from the RadioLynx board. With all this in place, all data from all sensors is synchronized to within a fraction of a millisecond.

3.2. **Improvements to UT GRID.** In order to enable collaborative mapping, we have had to customize GRID to enable more functionality. This section outlines some of the improvements we have made to GRID.

3.2.1. *Loosely-coupled GNSS/IMU.* Contrary to the usual GNSS applications, just the 3D position of the vehicle is not sufficient to generate maps. One must have knowledge of the full 6-DoF pose of the vehicle. We use our dual-antenna set up and the IMU to estimate the full pose of the vehicle in ECEF. It must be noted that we do not make use of LDRN for this, and our technique is a blend of code and carrier positioning but is not dependent on any outside reference network.. The IMU is loosely-coupled with GNSS in a complementary filter. The implementation follows a standard approach which can be found in [15].

3.2.2. *PPP-Lite.* We have transformed GRID SPS receiver into a code-phase precise point positioning (PPP) [16] engine. This futher helps to constrain visual SLAM without needing a reference network. PPP-Lite can be used in real-time (for GPS signals; not yet for Galileo). We now use precise orbits and ionospheric models provided by the International GNSS Service (IGS) to refine the SPS solution. (PPP-Lite is in testing stages, and has not been used for the results presented in the next section.)

3.3. **Collaborative Visual Mapping.** This section details the work done in this project towards the goal of collaborative mapping based on visual SLAM. As mentioned in Section 2, we developed on the open-source ORB-SLAM2 system.

3.3.1. *Map Reuse.* To enable collaborative mapping for vehicles driving through an area non-concurrently, they must be able to save and load the map created during the first session. Unfortunately, ORB-SLAM2 only supports single session mapping. As a part of this project, we developed the feature of saving and loading maps in to ORB-SLAM2.

As explained in Section 2, map points and keyframes are the basic containers used by ORB-SLAM2. These containers are inter-linked through pointers, based on the covisibility criterion. In order to reuse the saved map, we must also save all these linkages, along with the basic information such as camera poses and map point positions. Similarly, when loading the map, we must recreate all the pointer-based connections in order to perform bundle adjustment on these nodes.

Following the ROS-based architecture discussed above, we make the saved map available on a ROS topic. This published map can be saved by another node, and can also be used by any other node.

Figure 3 shows how multi-session or crowd-sourced mapping is enabled by the ability to save a map and load an existing map prior to the next session. The disconnected sections combine to form a self-consistent whole due to common ECEF reference, as described next.

3.3.2. *GNSS-Aided-Visual SLAM.* Visual SLAM algorithms build the map in a local frame. These maps cannot be intelligibly shared among vehicles. Furthermore, visual SLAM algorithms are known to drift with distance travelled. GNSS, while not very accurate under multipath, does not drift. GNSS/INS measurements can be used to correct the drift in visual SLAM.

(A) Map created after first session.

(B) Map created after loading the first map and appending to it.

FIGURE 3. Multi-session or crowd-sourced mapping enabled by map saving and loading.

The GNSS-aided-visual SLAM implementation in this project follows the design proposed in [17]. An overview of the system is presented in Figure 4.

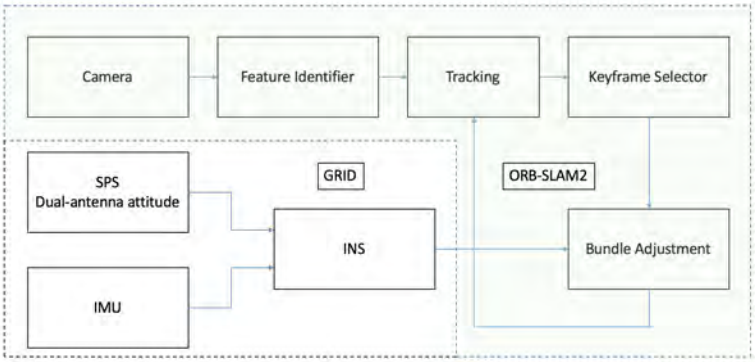


FIGURE 4. GNSS/INS-aided-SLAM system block diagram.

Following the GNSS/INS implementation described earlier, there are four coordinate frames in the system: the GNSS/INS frame, $\mathcal{I}_i$, the camera frame, $\mathcal{C}_i$, the SLAM frame, $\mathcal{S}$, and the global Earth Centered Earth Fixed (ECEF) frame, $\mathcal{G}$. The camera frame is centered at the left camera center, with z-axis along the boresight, x-axis pointing towards the right, and y-axis pointing downwards. The SLAM frame $\mathcal{S}$ is centered and oriented such that the position of the first keyframe (prior to any optimization) is at the origin and the attitude quaternion of the first keyframe is the identity quaternion. Thus, $\mathcal{S}$ is fixed relative to $\mathcal{G}$, but $\mathcal{I}_i$ and $\mathcal{C}_i$ change relative to $\mathcal{G}$ as the vehicle moves. The subscript i denotes the ith keyframe.

The state vector to be estimated is given as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_{\mathcal{S}}^{\mathcal{C}_1} & \boldsymbol{q}_{\mathcal{S}}^{\mathcal{C}_1} & \dots & \boldsymbol{x}_{\mathcal{S}}^{\mathcal{C}_N} & \boldsymbol{q}_{\mathcal{S}}^{\mathcal{C}_N} & \boldsymbol{x}_{\mathcal{S}}^{p_1} & \dots & \boldsymbol{x}_{\mathcal{S}}^{p_M} \end{bmatrix}^T$$

where N is the number of keyframes under optimization, M is the number of map points under optimization, and $\boldsymbol{x}_{\mathcal{S}}^{p_j}$ is the 3D position of the jth map point in the SLAM frame. One may argue that the state can directly be estimated in the ECEF frame $\mathcal{G}$. However, our experiments suggest that the large numbers associated with ECEF positions can cause numerical stability issues.

The GNSS/INS provides measurements of the position ($\tilde{\boldsymbol{x}}_{\mathcal{G}}^{\mathcal{I}_i}$) and attitude ($\tilde{\boldsymbol{q}}_{\mathcal{G}}^{\mathcal{I}_i}$) at the IMU center, and not the camera center ($\tilde{(\cdot)}$ indicates a measurement). However, the pose of the camera in the IMU frame $\mathcal{I}_i$, ($\boldsymbol{x}_{\mathcal{I}_i}^{\mathcal{C}_i}, \boldsymbol{q}_{\mathcal{I}_i}^{\mathcal{C}_i}$), is the same for all i and can be physically measured or estimated. Note that this transformation must be known accurately, since any errors will reflect directly as errors in the map. Also note that as of now, the full 6-DOF pose provided by the GNSS/INS is only directly used for determining the SLAM frame $\mathcal{S}$. Only the global position output of the GNSS/INS is used directly as measurements in the current implementation, with the attitude being used indirectly, as shown in the equations below. We transform the GNSS/INS position to get a direct measurement of the camera center in $\mathcal{S}$ as follows:

$$\tilde{\boldsymbol{x}}_{\mathcal{S}}^{\mathcal{C}_i} = R\left(\boldsymbol{q}_{\mathcal{S}}^{\mathcal{G}}\right) \tilde{\boldsymbol{x}}_{\mathcal{G}}^{\mathcal{C}_i} + \boldsymbol{x}_{\mathcal{S}}^{\mathcal{G}}$$

where

$$\tilde{\boldsymbol{x}}_{\mathcal{G}}^{\mathcal{C}_i} = \tilde{\boldsymbol{x}}_{\mathcal{G}}^{\mathcal{I}_i} + R\left(\tilde{\boldsymbol{q}}_{\mathcal{G}}^{\mathcal{I}_i}\right) \boldsymbol{x}_{\mathcal{I}_i}^{\mathcal{C}_i}$$

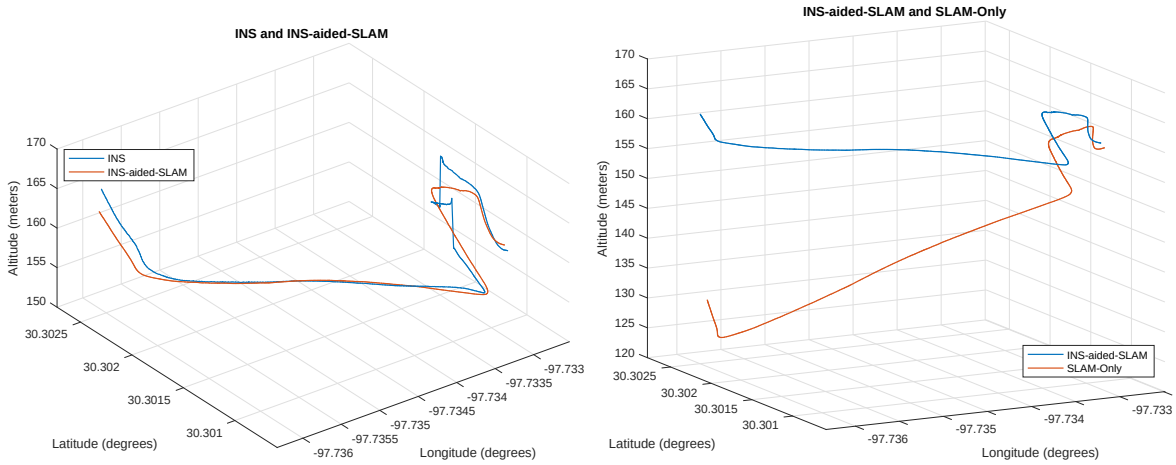
The measurement covariance is also transformed to $\mathcal{S}$ using the appropriate rotations. Numerical differentiation is being used to compute the partial derivatives of the measurement model with respect to the state elements. Using the attitude as a separate direct measurement is being worked on.

The images captured by the stereo rig are rectified before being passed to the estimator. The feature measurement on the 2D image plane is modeled as explained in Equation 2. The covariance associated with these measurements is provided by the ORB feature detector.

Figure 5 shows the output of (a) GNSS/INS, (b) GNSS/INS-aided-SLAM, and (c) SLAM-only plotted using Google Earth. The SLAM-only output is plotted on this ECEF map by passing it through the same $\mathcal{S}$ to $\mathcal{G}$ transform as used for the GNSS/INS-aided-SLAM output. It is clear that the SLAM-only solution drifts away from the road. With GNSS/INS measurements, the drift of visual SLAM is contained.

Figure 5 does not show the altitude variation for the three trajectories. This information is shown in Figures 6a and 6b. In Figure 6a, observe how the GNSS/INS output towards the beginning (right) is discontinuous jumps wildly in altitude. This is because the IMU biases may not have been estimated with high accuracy in the beginning of the dataset. Also, the vehicle was stationary at those locations, and it is common for the SPS GNSS output to wander during stationary scenarios. Even so, the GNSS/INS-aided-SLAM output is smooth. In Figure 6b, observe that the SLAM-only trajectory has a downward pitch as compared to the GNSS/INS-aided-SLAM trajectory. This is because the SLAM-only system is only provided an initial pose, and any error in that single epoch is not corrected because no further measurements are provided. On the other hand, with GNSS/INS-aided-SLAM, the continuous pose measurements are able to correct any small errors in the initial pose. This example reaffirms our belief that

FIGURE 5. Google Earth visualization of the GNSS/INS (yellow), GNSS/INS-aided-SLAM (green), and SLAM-only (red) trajectories.



(A) 3D visualization of GNSS/INS and GNSS/INS-aided-SLAM trajectories.

(B) 3D visualization of GNSS/INS-aided-SLAM and SLAM-only trajectories.

combined minimization of GNSS/INS and SLAM cost functions is beneficial for the overall system.

However, the GNSS/INS-aided-SLAM results shown in Figure 5 were not obtained using the default ORB-SLAM2 implementation of windowed BA. It was found that the ORB-SLAM2 windowing technique, while not incorrect, is ill-suited to integration with GNSS. The reason

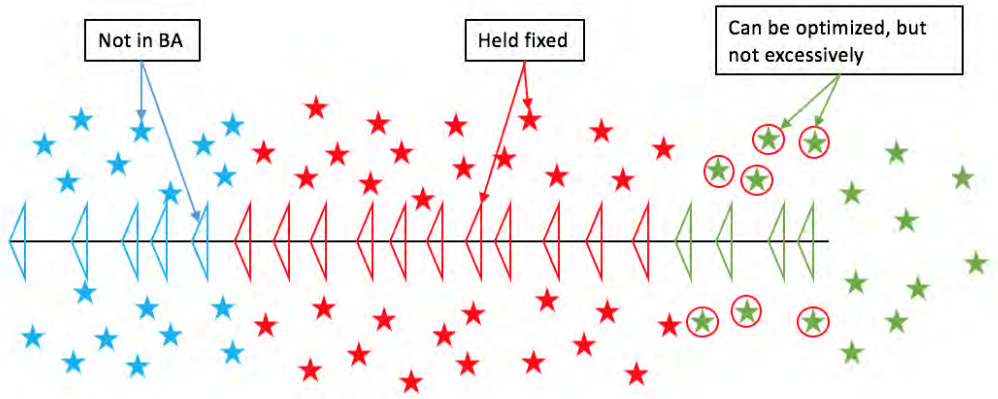


FIGURE 7. ORB-SLAM2 windowing scheme is not suitable for GNSS/INS-SLAM integration.

is explained in Figure 7. The ticks on the horizontal axis represent keyframes, while the stars represent map points. The right-most tick represents the latest keyframe. As mentioned earlier, ORB-SLAM2 uses covisibility information to choose windows for BA.

Green color in Figure 7 indicates vertices (keyframes and map points) that will be optimized in the windowed BA iteration. Red indicates vertices that participate in BA, but are held fixed and will not be optimized. Blue indicates vertices that do not participate in windowed BA. The most recent vertices are green, as expected. A large number of red vertices trail them. The red tail is inherently much longer than the green head. The reason is that a keyframe must share at least 15 map points in common with the *latest* keyframe in order to be green, but only needs to see any one of the green map points in order to be red. The red-circled green stars are the map points that are visible from both red and green keyframes. The red keyframes, being fixed, do not allow large movements in the red-circled green map points. This ensures that BA is only able to optimize the green vertices such that the overall result is smooth. However, this is not ideal for our GNSS-SLAM integration because the optimizer does not get to revisit the keyframes that may have been incorrectly position but have now been fixed. Consequently, the windowed BA scheme follows the GNSS/INS trajectory in the beginning, when there are no fixed vertices, but is stuck with the SLAM-only trajectory once a history of fixed (red) vertices prevents any deviation from the SLAM tracking thread.

Following the above discussion, global BA was implemented and performed at each keyframe step. The results were shown in Figures 5, 6a, and 6b, but the algorithm runs much slower, as expected.

In order to provide a longer window to BA while still keeping the computation manageable, one may consider other techniques such as fixed time- or distance-based windowing.

- *Fixed time window*: In this approach, a window is taken over all keyframes spawned in the last N seconds. However, this would fail if the vehicle were to stay stationary for a while.
- *Fixed distance window*: In this approach, a window is considered over all keyframes that appear within some radius of the approximate location of the current keyframe. However, this may include unrelated keyframes, such as ones from another parallel street, that may not influence the current map under consideration at all.
- *Fixed keyframe-length window*: In this approach, one could consider a window over the latest N keyframes. However, this does not extend well to the case where the same location is visited again. This is also the case with a fixed time window.

Another advantage of the default covisibility-based windowing is that it automatically extends to the case when a mapped area is visited again, that is, keyframes from a previous session are automatically connected to the current keyframes if they share at least 15 map points in common. Thus, it makes sense to use the covisibility-based approach but we must modify it such that a larger window of keyframes is available for optimization. To achieve this, we implement a deeper search over the covisibility graph. More specifically, one can specify the depth, n, over which the window must span. For example, for $n = 1$, the function includes all N_0 keyframes that are connected to the current keyframe, and also all the keyframes that are connected to these N_0 keyframes.

Another concern that this larger window addresses is that of the error correlation in the SPS GNSS output. Multipath in GNSS usually leads to time- and distance-correlated errors that can last for several meters. It is important that the keyframe window spans a larger distance than the GNSS error decorrelation distance. This is a promising research area for future work and is discussed later in Section 4.

Earlier in this section, it was observed how the GNSS/INS-aided-SLAM provided a smooth output even when the GNSS/INS measurements were quite noisy. Figure 8 shows the altitude time history of the GNSS/INS GNSS/INS-aided-SLAM trajectories. It is clear that feeding back the visual SLAM based output to the GNSS/INS can help correct SPS errors, as well as aid in better estimation of the IMU biases. Closing the loop around visual SLAM is another promising area of future research.

Figure 9a shows the performance of windowed BA with the proposed windowing scheme with $n = 3$. The GNSS/INS-aided-SLAM trajectory with global BA is also shown for comparison in Figure 9b. The performance penalty as compared to global BA is small, and exacerbated by the fact that the vehicle made a number of turns in the initial period, thereby cutting off the window soon for the initial keyframes..

3.3.3. *Union of Pose Associated Keyframes (UPAK).* Taking keyframes created over different sessions and performing a collective BA is at the heart of collaborative mapping. This becomes straightforward once covisibility-based windowed BA is set up. Consider a scenario where a vehicle revisits an area that has been previously mapped. The vehicle spawns its own keyframes based on some spawning algorithm. Since the older keyframes will have at least some covisibility

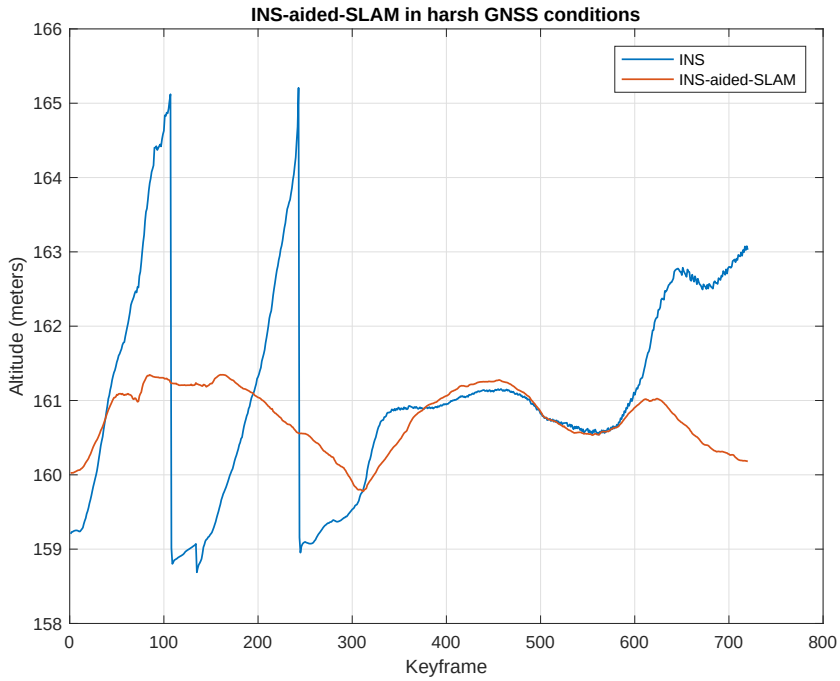
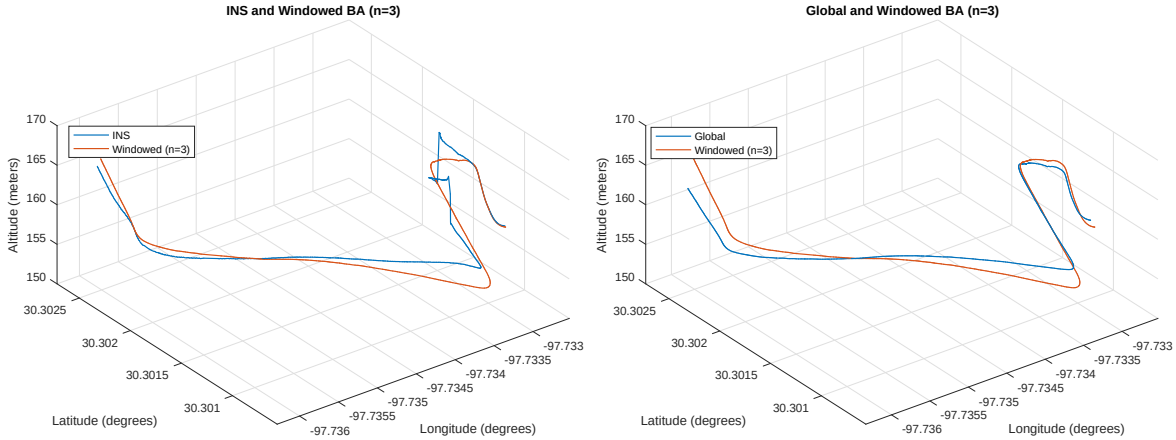


FIGURE 8. Altitude time history of the GNSS/INS and GNSS/INS-aided-SLAM trajectories.



(A) 3D visualization of GNSS/INS and GNSS/INS-aided-SLAM trajectory obtained using windowed BA with $n = 3$.

(B) 3D visualization of GNSS/INS-aided-SLAM trajectories obtained using global BA and windowed BA with $n = 3$.

with these new keyframes, they will automatically be connected to these new keyframes. In fact, in the second session, even keyframes that are geometrically ahead of the vehicle can also be connected to the newly spawned keyframe, resulting in a strong network of connected frames. A windowed BA can then be performed on this union of keyframes with associated GNSS/INS poses.

Figure 10 shows an example of a second session in an already mapped area.

Figures 11a and 11b show the SLAM-only output after two sessions through the same area. As expected the two trajectories are very close to each other. This indicates that the visual SLAM algorithm is self-consistent. However, as mentioned earlier, any errors in the initial pose used to map the ECEF frame persist in a SLAM-only system. As a result, the output from both sessions drifts from the GNSS/INS output.

Another important thing to notice is that the GNSS/INS trajectory is not repeatable over the two sessions. The two traces are different by as much as 3 meters. This is not unexpected when using SPS GNSS.

Figures 12a and 12b show the GNSS/INS-aided-SLAM output after the same two sessions. The value of UPAK is evident in these charts. The two sessions are still self-consistent, as expected from visual SLAM. But at the same time, including GNSS/INS output from both sessions minimizes the cost such that the overall map and localization is almost an average of the two different GNSS/INS trajectories. Two sessions is a small dataset to arrive at any conclusions about the accuracy of the system, but the initial results are promising. It is also easy to imagine how this can be extended to multiple vehicles at the same time, with each vehicle experience a somewhat different multipath environment.

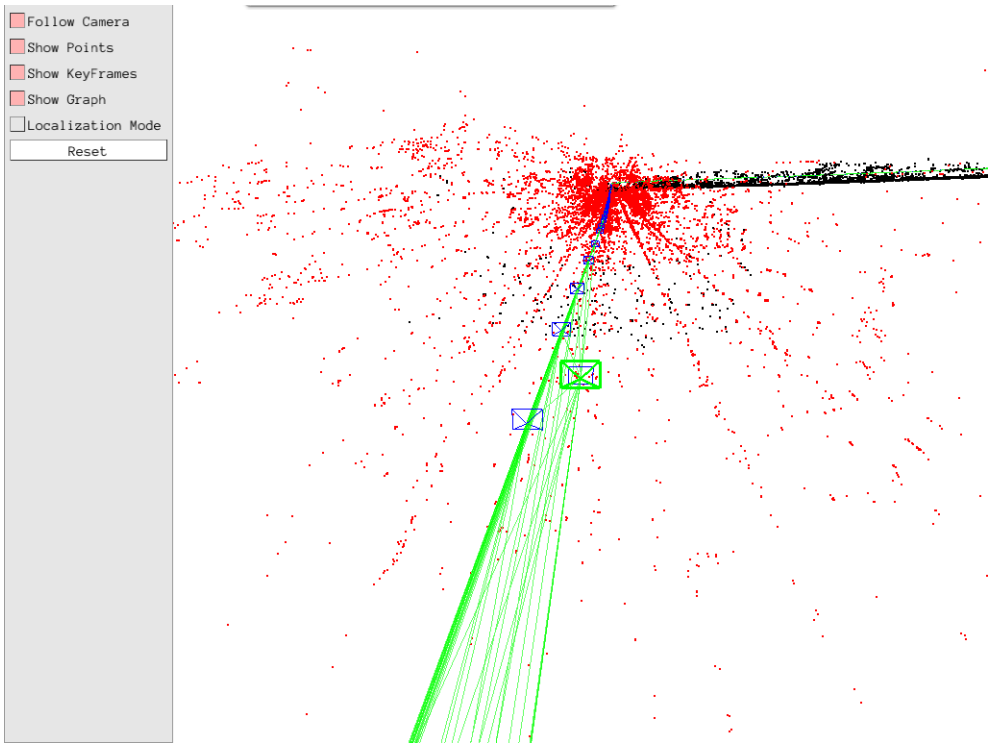
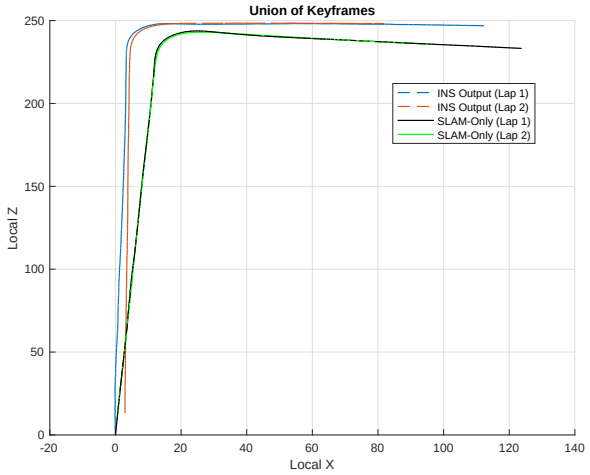
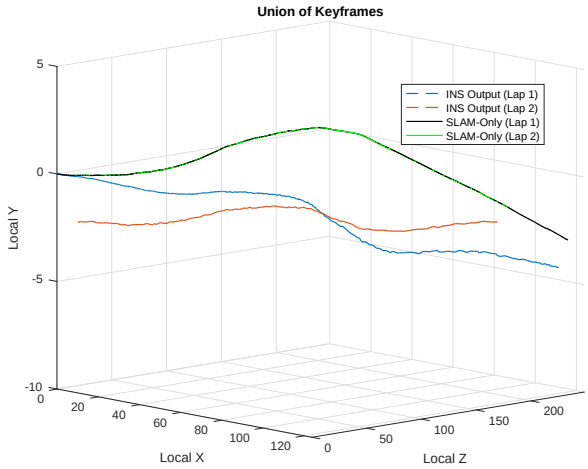


FIGURE 10. Keyframes from different sessions connected based on covisibility information.

3.3.4. *Localization in Prior Map.* Localization in a mapped area is a feature that is already built in to ORB-SLAM2. Figure 13 shows this feature at work with the data collected using

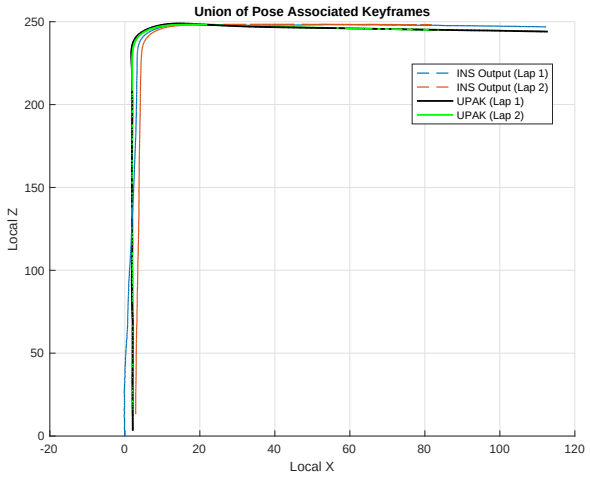


(A) Top view of SLAM-only output after two sessions.

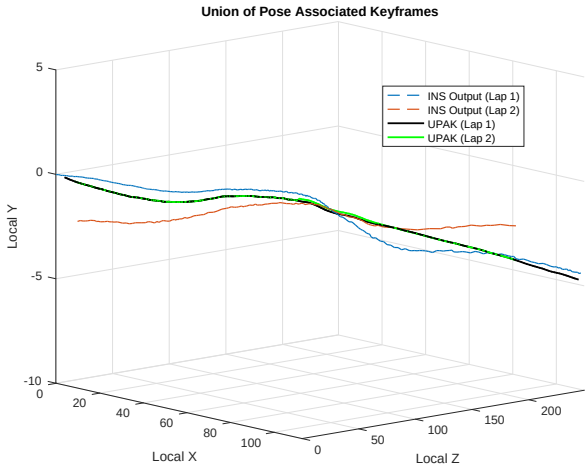


(B) Error in initialization pose persists in SLAM-only trajectory.

FIGURE 11. SLAM-only trajectory after two sessions through an area.



(A) Top view of the UPAK output.



(B) UPAK minimizes the cost for both sessions.

FIGURE 12. Trajectories obtained after BA over UPAK.

the Sensorium. The green keypoints represent matches from the previous map. The abundance of such matches is encouraging, even if the time difference between the two sessions was not large. It must be noted that the traffic during the second session is different, and the algorithm correctly does not match to any features on the vehicles. The blue keypoints are new features detected in the second session. These features are not added to the map in localization-only mode, and are only used for visual odometry.

3.3.5. *Map Point Scoring.* Another attractive application of collaborative mapping is using a survival-of-the-fittest approach to map points. Over multiple sessions, the system can infer map

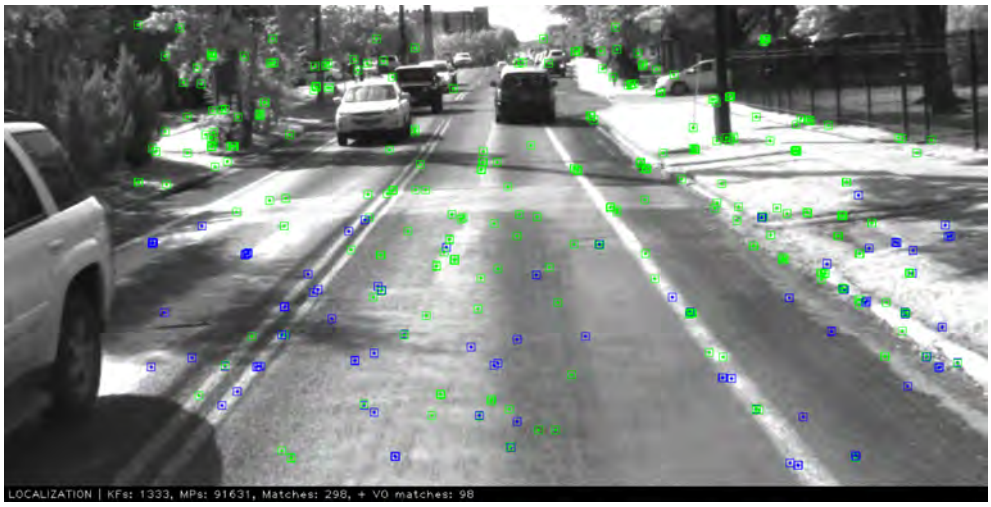


FIGURE 13. Localization in a prior map. Green keypoints represent matches from the prior map. Blue keypoints represent new detected features.

points that are ephemeral and prune them out of the map. This is not currently implemented in ORB-SLAM2.

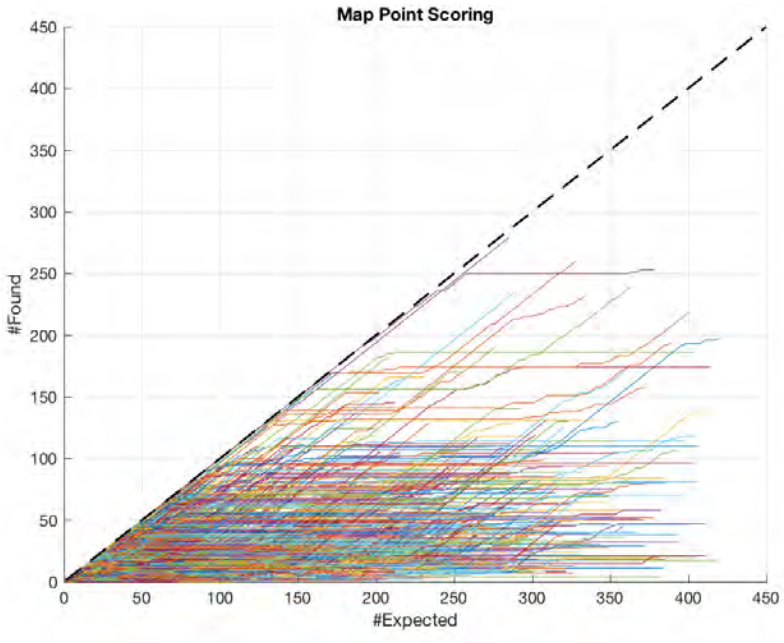


FIGURE 14. Map point scores after two sessions.

Again, using the same strategy with two sessions, Figure 14 shows the time history of visibility of all map points. The horizontal axis has the number of times a given map point was *expected* to be seen, that is, the number of times when the feature was in the field of view of the camera. The vertical axis has the number of times the feature was actually detected by the system.

The chart is cluttered because more than 10,000 time histories have been plotted, but is still informative. There are four categories of lines of this chart:

(1) The dashed black line is the ideal line, which is just a straight line with slope 1.
(2) Some time histories keep on growing continuously through both sessions (map point 559 in Figure 15). These are good map points that are always found when they are expected to be found.
(3) Some time histories rise, then become flat, and then start rising again during the next session (map point 5032 and 9241 in Figure 15). These are map points that were seen during the first session, then were not found for some period when they should have been found (because of obstructions, or other factors), but are seen again during the second session. These are a little hard to see in the cluttered region.
(4) Some time histories rise, but then go flat, and never rise even during the second session (map point 163 in Figure 15). These must be the map points that were spawned on transient stationary objects, like stopped cars.

Figure 16 shows the physical locations of the example map point time histories shown in Figure 15.

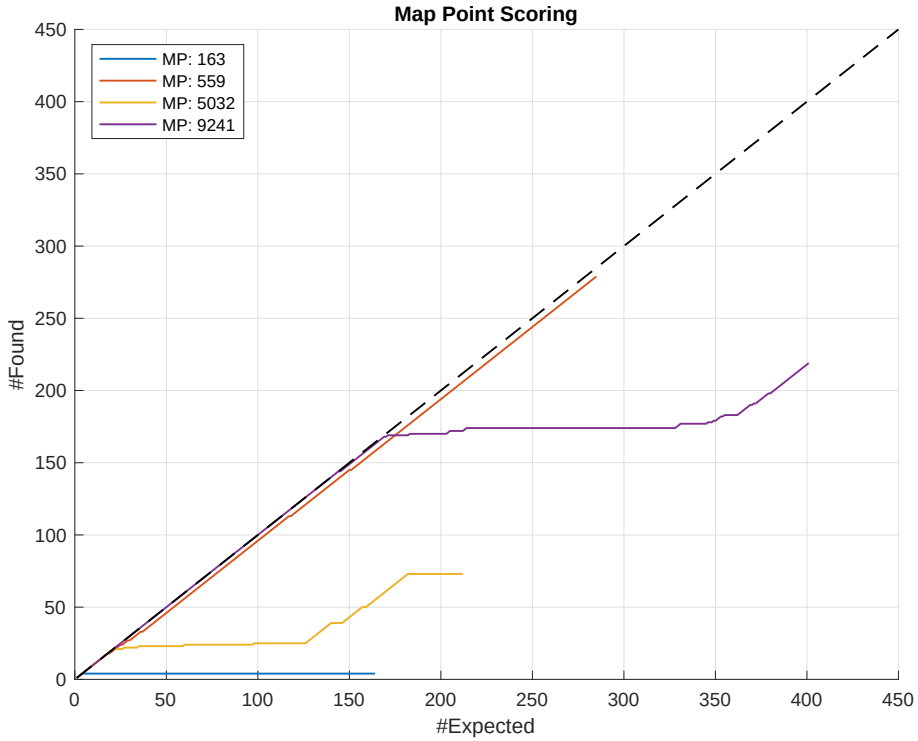
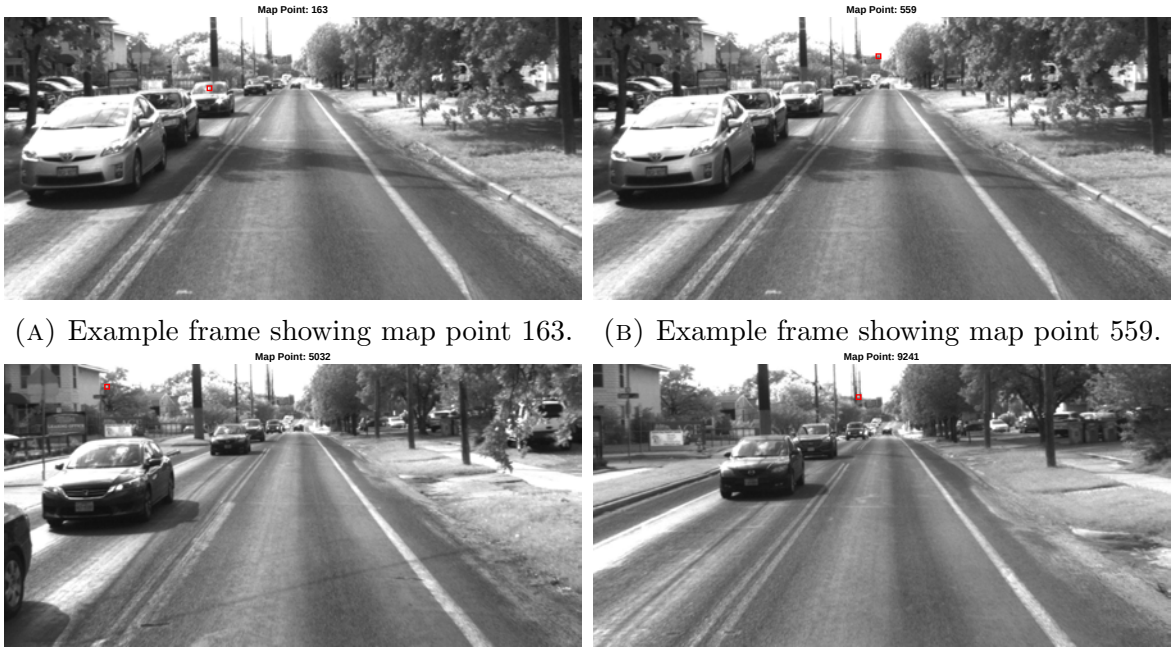


FIGURE 15. Map point scores of some selective map points after two sessions.

3.4. **Radar Mapping.** As mentioned in Section 2, radar sensors are indifferent to changing weather conditions, are inexpensive (as compared to LiDAR) and can be placed behind the



(A) Example frame showing map point 163. (B) Example frame showing map point 559.

(C) Example frame showing map point 5032. (D) Example frame showing map point 9241.

FIGURE 16. Example frames showing map point locations. Observe that the map points with long time histories are usually near infinity. Also note that a spurious map point 163 that spawned on a vehicle dies down quickly, and can be pruned. Finally, some map points like 5032 are expected to be seen, even when they might be obstructed by a building. This must be taken care of before pruning the map point.

bumpers of the vehicle. As a result, they are well established in the automotive industry, but have not yet been studied much for localization applications.

We believe that a combination of radar and inertial-aided-GNSS is vital for all-weather localization. To this end, in this project we started to explore the possibility of mapping using radar.

Using the Delphi ESR radar on the Sensorium, we logged the returned radar targets. The radar natively returns these targets in the body-fixed frame, but the availability of the GNSS/INS system, and the known transformation between the radar and GNSS/INS, enables us to plot the radar returns on an ECEF grid.

Figure 17 shows the radar returns as plotted on an ECEF grid. We also show a Google Earth view of the same area. Note that the radar targets were received over multiple sessions. The clustering of the targets indicates the repeatability of the radar detections. These results are encouraging, and we will integrate radar with visual SLAM in the future.

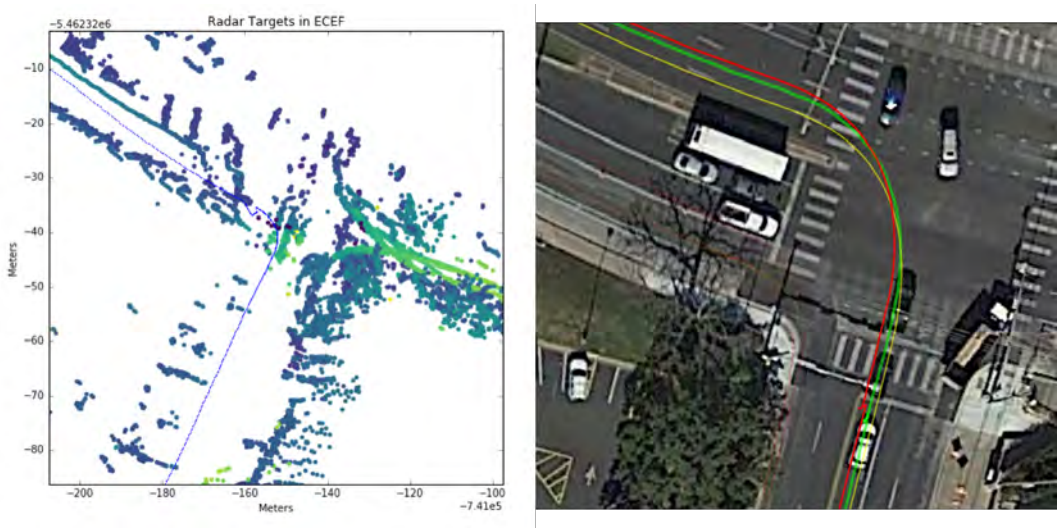


FIGURE 17. Radar returns from multiple sessions plotted on an ECEF grid.

4. Future Work

4.1. **Error-Correlation-Aware Windowing.** As mentioned in Section 3, integration of the GNSS/INS and visual SLAM needs a windowing technique that is aware of the time correlation in the GNSS/INS output. In addition, we also need a windowing technique that is aware of the availability of GNSS data. When GNSS data are sparse or totally unavailable, one needs to extend the BA window. This kind of apadtive windowing has not been studied in the literature, and we are in a good position to address this problem.

4.2. **Situation-Aware Map Slices.** Let us define a *situation* as a set

$$\{\text{location, time-of-day, day-of-year, weather, extraordinary-event}\}$$

We imagine having multiple versions of the map of an area, each applying to a subset of the above space. Each version is called a *slice*

$$\mathcal{M}_i = \mathcal{M}(\mathcal{L}_i \subseteq \text{location}, \mathcal{T}_i \subseteq \text{time-of-day}, \mathcal{D}_i \subseteq \text{day-of-year}, \mathcal{W}_i \subseteq \text{weather}, \mathcal{E}_i \subseteq \text{extraordinary-event})$$

where $\mathcal{M} = \cup_i \mathcal{M}_i$. We may send only these slices to vehicles, or may apply a mask at the vehicle that will activate a particular slice. There is clearly a benefit to working with slices: a map slice tailored for a given situation would offer more features, and hence more robustness.

One can also think of composing predictive maps: "Tomorrow will by overcast and the 3rd day of SWSX. Here is a map slice prepared based on a composition of historical data."

4.3. **Lifelong Mapping.** Following the discussion on map point scoring earlier, the feature point pruning process is fairly clear: if a certain feature point gets downvoted enough (by images compared against relevant map slices), then the feature point should be eliminated from the relevant slice.

Consider the steady state when an area has been well mapped. We must still continue to look out for new features. The strategy for birthing new map points in a mapped area is less clear. It depends on how new keyframes will be introduced in a mapped area.

4.4. **Non-Real-Time Mapping in Exploration.** Real-time localization in unmapped areas may not be a strict requirement for automated vehicles. In our experiments with BA, we observed that performing infrequent BA iterations with large batches of keyframes led to reasonable results, even with poor initial guesses. Thus, a possible architecture is to forgo localization as a real-time capability in unexplored regions, and perform BA with large window of keyframes. Once a map has been created using this strategy, then further refinement of the map and localization within the map can be performed in real-time.

4.5. **Radar Mapping.** Radar mapping has shown promise, but has not been included in the vision-based maps yet. We plan on implementing this in the next phase of the project.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, IEEE, 2007.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[5] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," *Robotics: Science and Systems VI*, 2010.

[6] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[7] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, 2007.

[9] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pp. 469–476, IEEE Computer Society, 2006.

[10] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320–2327, IEEE, 2011.

[11] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual SLAM: why filter?," *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[12] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmenta modern synthesis," in *International workshop on vision algorithms*, pp. 298–372, Springer, 1999.

[13] F. Schuster, C. Keller, M. Rapp, M. Haueis, and C. Curio, "Landmark based radar slam using graph optimization," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pp. 2559–2564, IEEE, 2016.

[14] M. Barjenbruch, D. Kellner, J. Klappstein, J. Dickmann, and K. Dietmayer, "Joint spatial-and Doppler-based ego-motion estimation for automotive radars," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 839–844, IEEE, 2015.

[15] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[16] J. Kouba and P. Héroux, "Precise point positioning using IGS orbit and clock products," *GPS solutions*, vol. 5, no. 2, pp. 12–28, 2001.

[17] D. P. Shepard and T. E. Humphreys, "High-precision globally-referenced position and attitude via a fusion of visual SLAM, carrier-phase-based GPS, and inertial measurements," in *Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION*, pp. 1309–1328, IEEE, 2014.

The University of Texas at Austin

E-mail address: lakshay.narula@utexas.edu