

Advanced SLAM Integration and Optimisation for Multi-Modal Vehicles



Final Year Research Project Thesis in partial fulfilment of
Masters of Professional Engineering

Eduardo Arteaga Guadarrama

21295129



THE UNIVERSITY OF
**WESTERN
AUSTRALIA**

School of Electrical, Electronic and Computer Engineering

Supervised by Professor Dr. Thomas Bräunl

Submitted on Friday 1st November 2019

Word Count: 7112

TABLE OF CONTENTS

<i>Index of Figures</i>	4
<i>Index of Tables</i>	4
<i>Nomenclature</i>	5
<i>Abstract</i>	6
1. <i>Introduction</i>	7
1.1 Project Goals	7
1.2 Problem Identification	8
1.2.1 Advanced LiDAR SLAM	9
1.2.2 Advanced Visual SLAM.....	10
1.2.3 Map Memory/Recognition	10
2. <i>Literature Review</i>	11
3. <i>Design Architecture & Resources</i>	14
3.1 SAE Overview	14
3.1.1 On-Board Computer	14
3.1.2 Software Platform - ROS.....	15
3.2 Stereo-Visual SLAM Architecture	15
3.2.1 Stages of Stereo-Visual SLAM Architecture:.....	16
3.3 Multi-LiDAR SLAM	17
3.3.1 Stages of Multi-LiDAR SLAM Architecture:	18
3.4 Mapping	19
3.5 Resources: Hardware & Software	20
4. <i>Enabling Works</i>	21
4.1 SAE Autonomous Test Drives	21
4.2 Hardware Installation	21
4.3 Wheel Replacements	22
4.4 Mono-Visual SLAM Preliminary Results	22
5. <i>Design Implementation</i>	24
5.1 Stereo-Visual SLAM	24
5.1.1 Synchronised Capture.....	24
5.1.2 Camera Calibration	25
5.1.3 Depth Imaging	26
5.2 Multi-LiDAR SLAM	Error! Bookmark not defined.
5.2.1 Velodyne Point Cloud.....	Error! Bookmark not defined.
5.3 Multi-LiDAR SLAM	27
5.3.1 Velodyne Point Cloud Testing	27
6. <i>Experiments</i>	28

6.1	Strobe Signal Testing.....	28
6.2	Frame Comparison Drop Test.....	28
7.	<i>Conclusions</i>	30
7.1	Future Work and Recommendations.....	30
8.	<i>References</i>	31

INDEX OF FIGURES

Figure 1 - The REV Team SAE Vehicle.....	7
Figure 2 - 2019 SAE Autonomous High-Level Architecture.....	8
Figure 3 - Front Mounted SICK (Left) & Central Mounted LUX (Right) LiDARs	9
Figure 4 - Dual Camera mounting on SAE	10
Figure 5 – Example of Point Cloud course Mapping [3].....	10
Figure 6 - Predicted sensor technologies in future self-driving cars.....	11
Figure 7 - ORB SLAM2 Trajectory and Environment Reconstruction, with loop closures [22]	12
Figure 8 - Left & Right Image Disparity (blue) [29].....	13
Figure 9 - Stereo Camera Parameters [23].....	13
Figure 10 - High-Level System: Sensors, Communications and Power	14
Figure 11 - Stereo-Visual SLAM System Architecture	15
Figure 12 - Example of SGBM generated stereo disparity map [30].....	16
Figure 13 - Multi-LiDAR SLAM System Architecture	17
Figure 14 - Indoor LiDAR Mapping example using Cartographer.....	18
Figure 15 – SAE Autonomous Test Drive.....	21
Figure 16 - Redesigned 12V Wiring.....	21
Figure 17 - SAE Rear Wheel replacement	22
Figure 18 - Mono ORB SLAM2 Testing	23
Figure 19 - Primary to Secondary Camera GPIO Connections	25
Figure 20 - Example of Checkerboard Calibration	26
Figure 21 - Left and Right Test Images.....	26
Figure 22 - Depth Image.....	27
Figure 23 - Puck LiDAR Imaging using VeloView	27
Figure 24 - Strobe Signal as visualised on oscilloscope.....	28
Figure 25 - Drop Test Setup	29
Figure 26 - Drop Test Example Frames and disparity.....	29

INDEX OF TABLES

Table 1 - Table of project resources.....	20
---	----

NOMENCLATURE

UWA	University of Western Australia
SAE	Society of Automotive Engineers
REV	The Renewable Energy Vehicle Project
SLAM	Simultaneous Localisation And Mapping
GUI	Graphical User Interface
ROS	Robot Operating System (middleware for robotic collaborative development)
TX1	Nvidia Jetson TX1 (platform on which the high-level system is built)
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LiDAR	Light Detection And Ranging
KITTI	Karlsruhe Institute of Technology & Toyota Institute at Chicago
ORB	Oriented-fast, Rotated Brief
SGBM	Semi-Global Block Matching
FLIR	Forward Looking Infrared Cameras (Name of the camera manufacturer)
GPIO	General Purpose Input / Output
SDK	Software Development Kit
FPS	Frames Per Second
API	Application Programming Interface
DC-DC	Direct Current to Direct Current Converter

ABSTRACT

The Localisation and Mapping problem is a crucial but difficult-to-address issue for advanced autonomous driving; identifying a vehicle's position usually requires a map and mapping its surroundings requires an accurate estimate of its location. A simultaneous approach to engaging this paradoxical problem, commonly referred to as 'SLAM', has previously been enabled by software frameworks alongside increasingly powerful and accessible sensors such as RADAR, GPS and LIDAR. However, despite the associated increase in computational complexity, Vision Cameras have emerged as a competitive tool for acting as the primary sensor for the sensor-suites required for tomorrow's autonomous vehicles. This is primarily attributed to the comparatively low cost, and the sensory richness available (akin to the human eye). This paper presents a design and implementation of a parallel Vision and LIDAR Stereo-SLAM system to demonstrate the capabilities of advanced vision-based sensor-suites. The designed system is geared towards use in performance vehicles, to accommodate a wide range of driving modes and scenarios. Using dual cameras and multiple lidars, the sensors are integrated into a software framework with rapid computational performance to accommodate for real-time autonomous driving. The final output of this system is a node that interacts with other modules in the vehicle's high-level control system, such as the navigation and path planning, to allow real-world autonomous driving. Results are shown as functional performance tests and visualisations created from field test-drives.

1. INTRODUCTION

This section introduces the research project background, goals and scope of problems to be addressed

1.1 Project Goals

The SAE–Autonomous project, as it is known today, began in 2013 when the Electric SAE car (**Figure 1**) belonging to the REV Project Team was modified to drive-by-wire under the command of a high-level control system. In successive years the SAE car was adapted to navigate a pre-programmed route using GPS waypoints (2015), after which LiDAR (2017) and Vision Cameras (2018) were explored as sensors for localisation and navigation. At the beginning of 2019, the SAE could demonstrate the ability to detect obstacles (e.g. traffic cones) and navigate to avoid them in real time.



Figure 1 - The REV Team SAE Vehicle

The goals for advancing the SAE Autonomous project this year focus on improving performance and autonomous capability in the following two scenarios:

- Driving on the UWA campus internal roads
- High-performance driving on a competition style racetrack

While some work has previously been done to integrate and test the sensor-based systems for autonomous driving on the SAE, this year's team has put significant focus on improving their integration and performance.

The new SAE Autonomous team responsible for pursuing these goals consists of two electrical/electronic and three mechanical engineering students and will be able cover a wide variety of sub-topics to play to their strengths. The team members, Eduardo Arteaga (the author), Junwen Huang, Jiajian Shao, Jia Yu and Tuo Zhang, will each concentrate on one of the five key topics identified. A simple overview of these topics and their interrelations can be seen in **Figure 2** below.



Figure 2 - 2019 SAE Autonomous High-Level Architecture

This research paper is dedicated to upgrading the SAE Localisation and Mapping (SLAM) capabilities and performance for real-time use in road and track driving. This will be realised through the following goals:

- Identify the localisation sensors suited to each driving application
- Exploration of advanced processing methods of sensor-outputs
- Integration of system with ROS (the SAE’s control/module operating system)

1.2 Problem Identification

Both LiDAR and Camera sensors have previously been explored [1] [2] for autonomous driving on the SAE vehicle, providing object detection capabilities. However, the results are not yet sufficiently complex or robust enough to build a descriptive map of the dynamic driving environments specified in **Project Goals** or allow planning of complex manoeuvres with the navigation system. There appeared to be two approaches to explore for advancing the vehicle’s localisation: the fusion of different sensors to provide higher accuracy localisation or investigation of more advanced localisation methods one sensor type at a time. To investigate both of these paths for the SAE, this thesis focuses on *advanced sensor methods* while a fellow teammate’s research (Jia Yu) focused on *sensor fusion*.

As such, the problems relating to this thesis that need solving have been divided into three distinct outputs: An advanced LiDAR-based SLAM output, an advanced camera-based SLAM output (commonly known as Visual SLAM) and a “Map-Memory” to store maps from previous drives. Additionally, these must integrate with the existing (and concurrently developed) software and hardware architectures to achieve real time autonomous driving. These problems are further characterised as follows:

1.2.1 Advanced LiDAR SLAM

As will be covered in the **Literature Review** in more detail, *Localisation and Mapping* is essential for an autonomous robot or vehicle to perform. Previously, use of LiDAR for SLAM has been explored on the SAE vehicle using single front-facing LiDARs. A low front-mounted SICK LiDAR and a raised central-mounted LUX LiDAR (**Figure 3**) have been independently trialled.

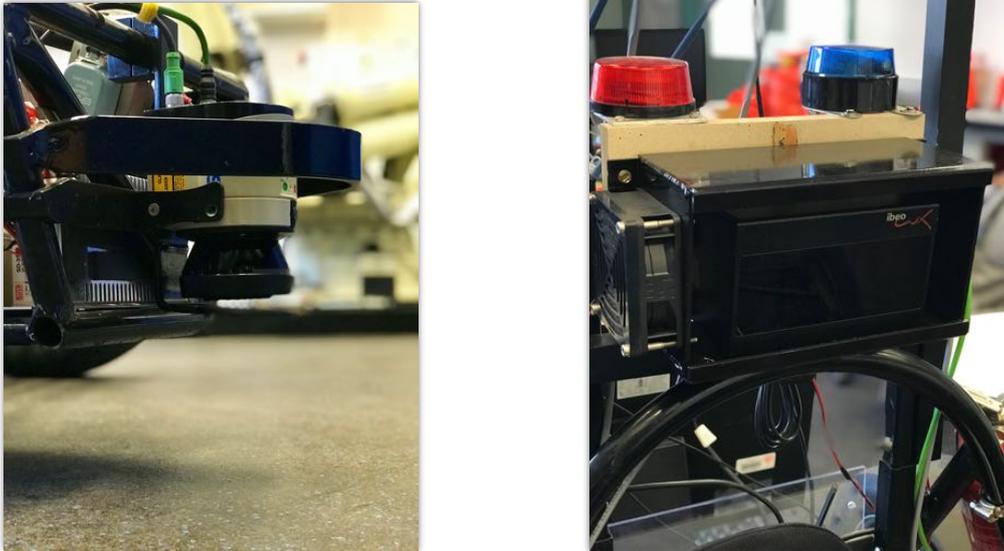


Figure 3 - Front Mounted SICK (Left) & Central Mounted LUX (Right) LiDARs

Some issues that have been identified with previous methods include:

- A small field of vision due to the angular range limitations of the LiDARs,
- A trade-off between long-range and short-range object detection dependent on the mounting position

To address this the REV team has acquired a 360° LiDAR with high point density. The strategy covered in this thesis is to integrate the 360° LiDAR with the high-level system and combine the point cloud produced with that of the front-mounted SICK to produce a comprehensive image of vehicle surroundings, coined Multi-LiDAR SLAM.

1.2.2 Advanced Visual SLAM

Similar to the LiDAR case, a single camera has previously been used to extract information from the environment and map surroundings. However it has been shown that there are still obstacles to overcome [2], as the mapping methods are computationally intensive and are adequate mainly for static environments at low speeds.

This thesis explores the use of multiple cameras (Stereo-Visual SLAM) to address the level-of-detail and computational-runtime issues found with Mono-Visual SLAM. The existing camera mounting is pictured in **Figure 4**.

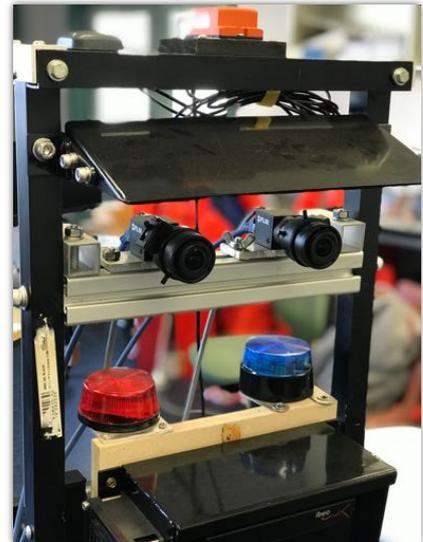


Figure 4 - Dual Camera mounting on SAE

1.2.3 Map Memory/Recognition

One of the key goals for high-performance track driving is the ability for the vehicle to navigate a new track and subsequently navigate it faster and more efficiently upon encountering it again. The first steps to achieving this are to integrate the SLAM with the car's navigation and provide the capability of storing a route's map so it can repeat its course later (e.g. **Figure 5**). Ultimately the task of recognising a route and performing it at higher speed will involve sensor fusion and advanced navigation algorithms.

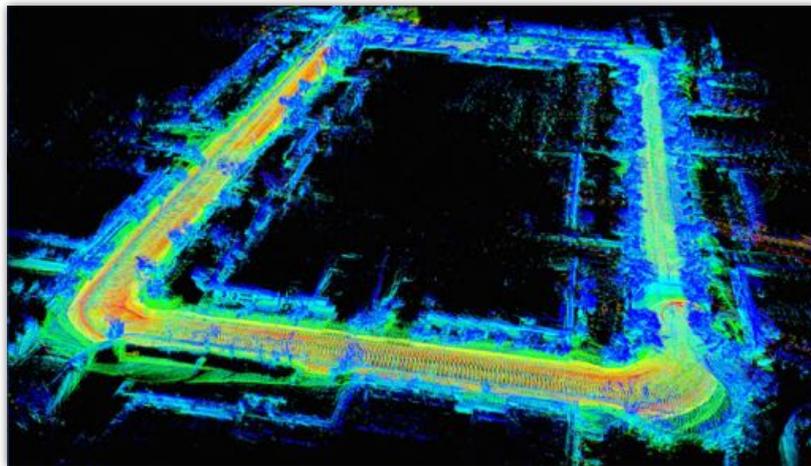


Figure 5 – Example of Point Cloud course Mapping [3]

2. LITERATURE REVIEW

In a nutshell, the essential feature that sets autonomous vehicles apart is their ability to self-navigate. To fully solve this problem the *localisation*, *mapping* and *path planning* need to be addressed at the same time. Generally, these tasks cannot be decoupled [4] as a vehicle must know where it is in relation to its surroundings and vice-versa to plan and execute navigation. SLAM is a subset of this problem, involving Simultaneous Localisation And Mapping [5], and can be paradoxically complex because for a robot to know it's localisation it needs an consistent map, but to generate a map it needs an accurate estimate of its localisation [6]. There are multiple technologies that have been investigated for producing SLAM, including the use of single visual cameras (mono) [7], multiple cameras (stereo) [8], lidars [9] and even radar [10].

In the automotive industry research and development of SLAM technology is incredibly valuable, sometimes attracting industry heavyweights like Volvo and Uber to share expertise and form a partnership [11]. SAE International's J3016 Standard defines a vehicle's level of automation from 0 to 5, with a higher number signifying a higher level of autonomy [12]. Level 3 autonomy is just now available in some commercial vehicles, like the Audi A8 [13] released in 2018, but Level 5 autonomy is a big step. Level 5 autonomy requires the vehicle to perform all safety critical features without requiring or expecting a driver to take control under any circumstances [12]. For this to be achieved the vehicle has to be extremely aware of its surroundings, which vehicle makers envision as a full suite of sensor [14] akin to the artist rendition in **Figure 6**.

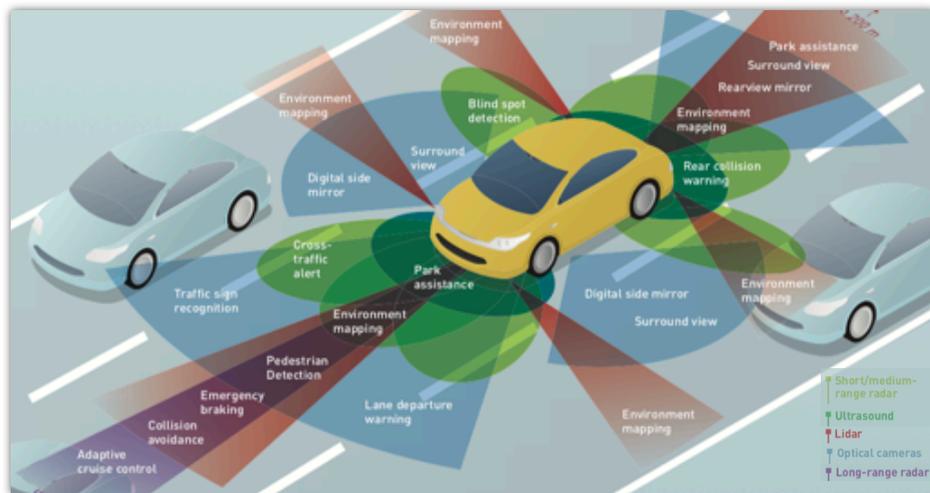


Figure 6 - Predicted sensor technologies in future self-driving cars

In other words, multiple sensors and sensor types must be fused, or at least integrated in parallel, to allow SLAM and subsequently provide a comprehensive input to a vehicle's control system.

Automotive racing is often regarded as being at the forefront of vehicular transportation technology, and up until the last five years there had not been much overlap with robotics. In 2015 the Swiss Federal Institute of Technology's Formula Electric team showcased their autonomous car, Flüela [15], at the annual Formula Student Electric Race in Germany and took out first place overall for the autonomous competition. Their greatest advantage was thanks to their algorithm which enabled the car to create its own map of the course during a slow-paced first lap. This capability was achieved through an approach of integrating vision sensors, LiDAR sensors and odometry. The resulting mapping would then be used to

pre-plan the vehicles path to be taken at competition speed the second time through. Inspired by the state-of-the-art in vehicle automation, this thesis will comparatively explore Stereo-SLAM using cameras and LiDAR with the goal of improving runtime performance, accuracy and field of view suitable for dynamic environments and rapid manoeuvring.

The primary appeal of Visual SLAM is its ability to deliver rich data (shape, colour, image position) and at a much lower cost compared to LiDAR technology. On the other hand, the drawbacks to this approach include image-capture's optical sensitivity to high and low brightness which can cause errors due to motion blur or noise [16], and the high computational complexity of continuous image processing [17].

The KITTI Vision Benchmark Suite is a body of knowledge that serves to evaluate Computer Vision Methods of the past decade, where Visual Odometry/SLAM is one of the key areas of interest [18]. ORB SLAM2 is one the fastest open-source visual SLAM algorithm with a runtime of only 0.06s under benchmark testing [19] and offers attractive additional features. ORB SLAM2 detects visual features using Oriented FAST and Rotated BRIEF (ORB), which is a rotation invariant and resistant to noise [20]. ORB is used in all its modules including mapping, tracking and place recognition, a method with two orders of magnitude lower runtimes than its predecessor SIFT [20]. ORB SLAM2 performs local mapping and camera tracking, and additionally incorporates a third parallel thread to its procedure: loop closure and re-localisation [21] as the example pictured in **Figure 7**. With the addition of Stereo and RGB-D image capabilities in 2017 [22], ORB SLAM2 offers itself as a fast, accurate and importantly versatile system to test implementation of advanced SLAM on the SAE.



Figure 7 - ORB SLAM2 Trajectory and Environment Reconstruction, with loop closures [22]

Stereo imaging produces a depth image by computing the distance to objects in an image based off the relative pixel position of a feature between images taken by two parallel cameras and the baseline distance between cameras [23]. Using a Stereo Camera for SLAM addresses the computational runtime issues usually experienced in mono Visual SLAM since minimal CPU usage is required when the camera pre-processes a planar point cloud, and offers higher map densities similar to using LiDAR technology [17]. Purpose built stereo cameras are available on the market [24], but to maximise the low cost appeal of Visual SLAM one can be built from two grayscale or RGB cameras programmed to compute the depth image [25] [26].

Building a stereo depth image is most easily achieved using two images from cameras capturing the same scene with (ideally) the same exposure and focal settings from slightly varying perspectives, where the focal length f and baseline length b are known variables [23].

Camera calibration is an important step to correct for lens distortion and allow the 2D image coordinate system to correspond to the real-world 3D coordinate system [27]. Open source software packages are available [26] [28] to compute the intrinsic and extrinsic parameters of mono and stereo cameras to map image coordinates to real-world coordinates. Stereo Disparity computation is the next step to achieving Stereo Imaging, where points on a left and right image are identified and the disparity (i.e. distance between left and right position) is calculated [23]. This is exemplified for two simple points on the left and right images in **Figure 8**. The disparity parameter d is related to focal length, baseline and depth by:

$$d \propto \frac{f b}{z} \quad \text{as described in Figure 9.}$$

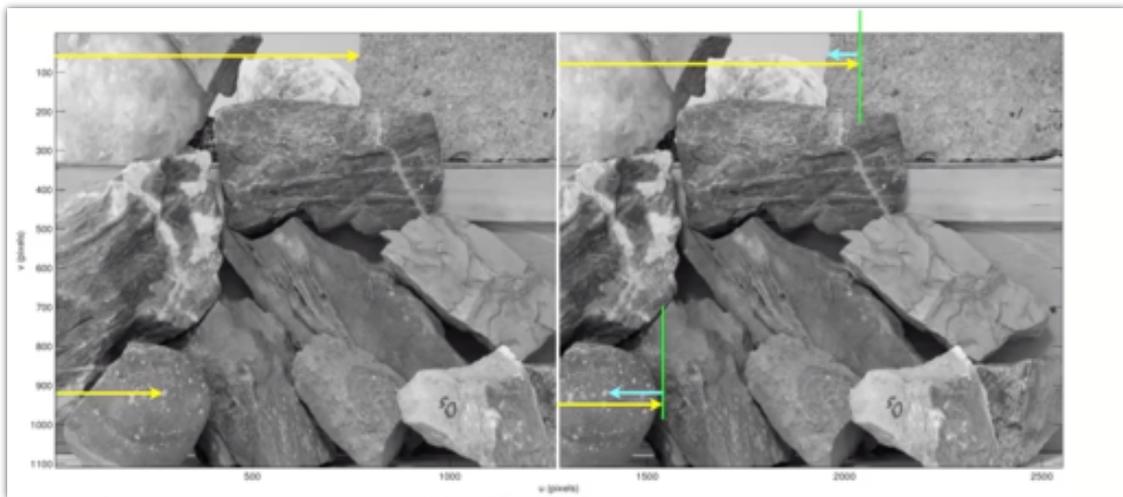


Figure 8 - Left & Right Image Disparity (blue) [29]

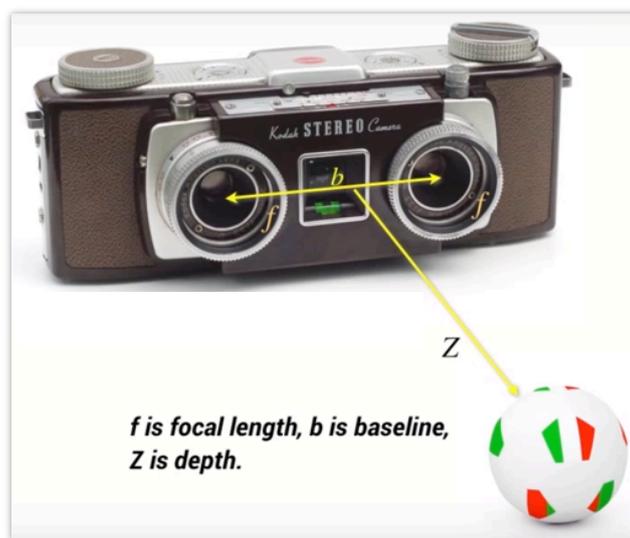


Figure 9 - Stereo Camera Parameters [23]

Semi-Global Block Matching (SGBM) algorithm [30] is one algorithm to compute disparity and produce a depth image. This method uses information from each pixel and its neighbours as a compromise between disparity accuracy and computational time. Open-Source libraries based on OpenCV [31] are available to run SGBM for depth imaging.

3. DESIGN ARCHITECTURE & RESOURCES

A high-level overview of the design architecture for each element of the project is provided in this section along with a table summarising the resources used for the project

3.1 SAE Overview

This year the SAE's *high-level system* has been fitted with new and upgraded hardware to pursue the project goals outlined in **Section 1.1**. New additions include a Velodyne Puck™ LiDAR, a Netgear GS108 8 Port Gigabit Switch and a 9-Terminal Bus Bar. The high-level system hardware is summarised in **Figure 10**.

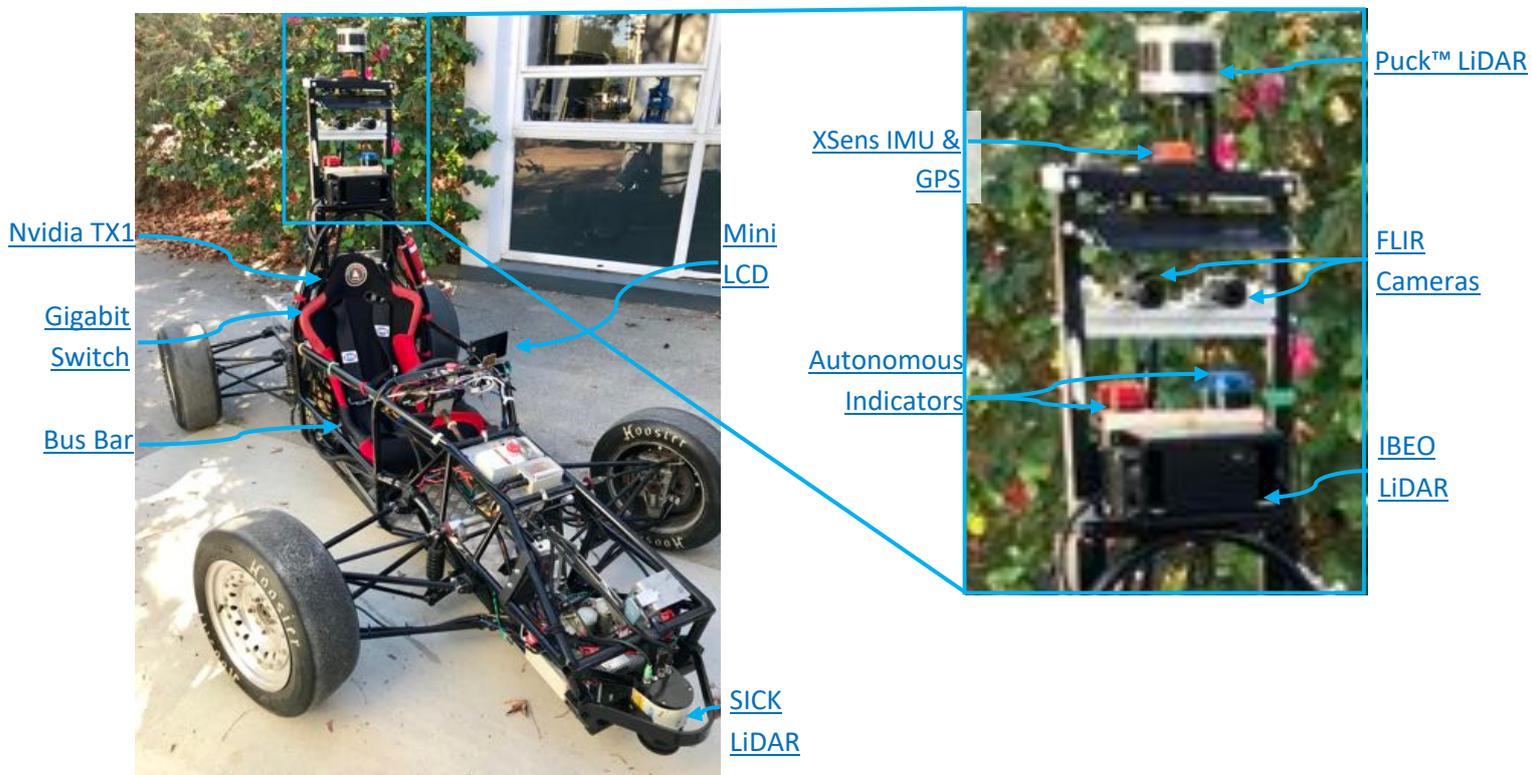


Figure 10 - High-Level System: Sensors, Communications and Power

3.1.1 On-Board Computer

The Nvidia Jetson TX1 (Jetson) is an ARM-board with 64-bit architecture and acts as the vehicle's on-board computer, which is used as the interface to the SAE's high-level control system and as a data storage device. The Jetson runs Ubuntu 16.04, a Linux based Operating System (OS), which is interfaceable by connecting a keyboard, mouse and either the on-board Mini-LCD or an external monitor. It has similar capabilities to a conventional computer, such as file storage/navigation, internet browsing and peripheral connectivity, but requires less power and is more compact. The Jetson is the central piece of equipment on the SAE, as all other devices must physically interface with it (directly or indirectly). An Nvidia Graphics card is also installed on the device for dedicated graphics processing.

3.1.2 Software Platform - ROS

In 2018 the SAE's high-level control system (running on the Jetson) was redesigned to improve flexibility, resilience and modularity. The Robot Operating System (ROS) [32] was chosen as the platform for high-level control for its ability to highly decouple software nodes, where interactions between node input and output topics need only conform to the defined message type. In essence, ROS allows nodes (programs) to subscribe or publish to topics (inputs/outputs) to create a wholistic, structured, software-based architecture for high-level control of the SAE. Version *ROS Kinetic* is installed on the Jetson for compatibility with Ubuntu 16.04. Details on how the design covered in this thesis integrates with ROS are explained in the following sections.

3.2 Stereo-Visual SLAM Architecture

The design requirements for *Advanced Visual SLAM*, as set in **Section 1**, determined that a Stereo-Visual SLAM system be designed for the SAE to improve accuracy, map density, computational runtime, and to integrate with the ROS control architecture. The two high-resolution FLIR Blackfly Cameras already mounted on the central frame of the vehicle are used as the sensors for this system. The cameras interface with the Jetson over Gigabit Ethernet cable via the Gigabit Switch.

Based on learnings from the literature in **Section 2**, the final output required from this system is a *ROS topic* of type *point cloud* which maps the vehicle's environment for the Path Planning node to use for navigation. **Figure 11** introduces the high-level design architecture for the new Stereo-Visual SLAM system. The numbers one to four in the figure correspond to the stages of architecture as described in **Section 3.2.1**.

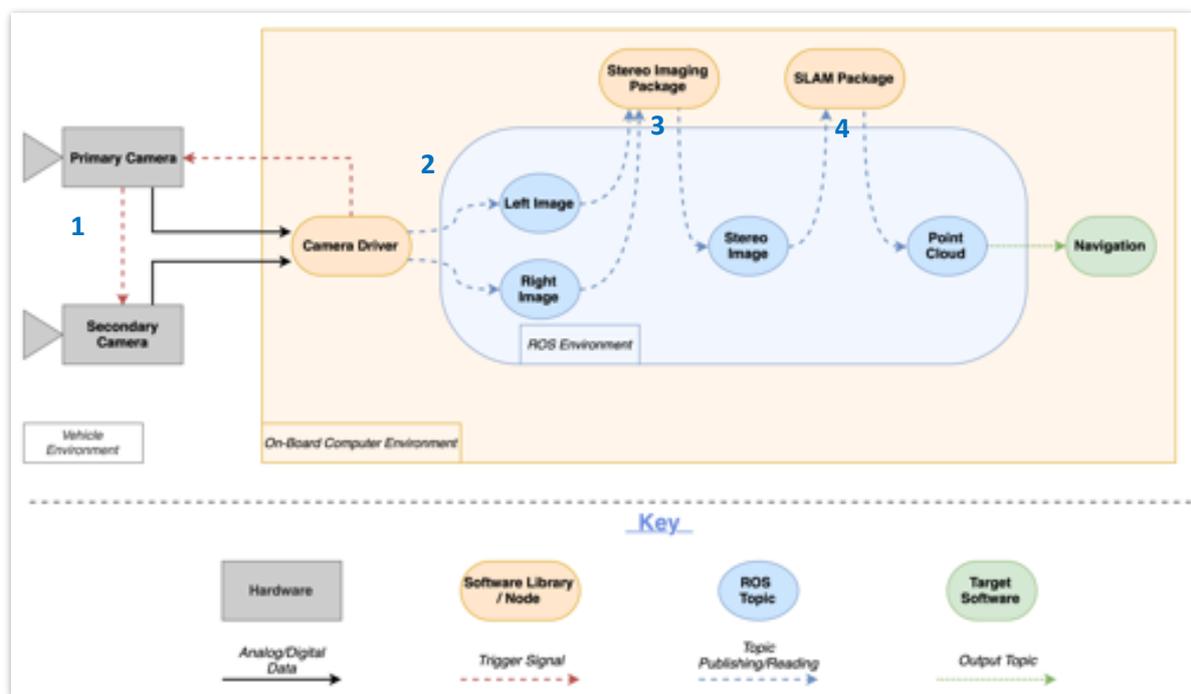


Figure 11 - Stereo-Visual SLAM System Architecture

3.2.1 Stages of Stereo-Visual SLAM Architecture:

1. Synchronised Image Capture must be achieved with the left and right cameras, as the left and right images must be of the same scene and at (virtually) the same time to construct a meaningful depth image. This will be achieved through a *hardware trigger signal* sent from one camera (the primary camera) to the other (the secondary camera) upon exposure as the primary begins to take an image. A camera driver currently exists on the ROS system which calls upon the cameras to individually begin capturing footage and publish images to the *left_image* and *right_image* ROS topics, but this will have to be redesigned to run both cameras configured to capture synchronously.
2. The cameras must be calibrated to remove lens distortion and enable accurate mapping of real-world coordinate points from 2D image coordinates. An open source library (e.g. [33], [26], [28]) for calibration is an efficient way to solve this. Calibration should only have to be done once for each camera, as the transformation matrices produce can be used for any image taken once calibration is done. However, this only applies for a specific focal configuration and will need to be performed again if the configuration of image size changes.
3. The synchronised left and right images published must be processed using a stereo imaging method to produce a depth image. This method must integrate with ROS to subscribe to the images, perform a stereo disparity algorithm using the camera parameters (f , b), and publish the new resulting *depth_image* topic. The method must have a fast enough computational runtime to produce depth images for use in real time driving (5 FPS or better desired). An open source library (e.g. [30], [31]) using an algorithm like SGBM can be modified to integrate with the ROS architecture. **Figure 12** depicts an example of the SGBM method used to build a depth image in a traffic environment.



Figure 12 - Example of SGBM generated stereo disparity map [30]

- Once depth images are obtained, they can be processed with a Stereo SLAM package to build a map and trajectory. ORB SLAM2 has been identified as an open source SLAM package that suits the requirements for autonomous driving on the SAE due to its Stereo image support, fast performance, robustness, and loop closure capabilities. Additionally, ORB SLAM2 features ROS support. The final SLAM outputs consist of a point cloud mapping the environment, and a pose following the cameras trajectory. Finally, these can be published to ROS as topics *visual map* and *visual_odometry*.

3.3 Multi-LiDAR SLAM

Similar to the Stereo-Visual SLAM system, the design requirements set in **Section 1** determined that a Multi-LiDAR SLAM system designed for the SAE should feature a more comprehensive field of view (including immediate obstacles near the ground and objects further away) with high point density, whilst integrating with the ROS architecture and not dramatically affecting runtime. For this design the existing front-mounted SICK LiDAR and the newly installed Velodyne PUCK[®] LiDAR (mounted atop the central frame) can be used as input sensors. The LiDARs interface with the Jetson over Gigabit Ethernet cable via the Gigabit Switch.

Like in **Section 3.2**, the final output required from this system is a *point cloud* ROS topic for use with navigation nodes. **Figure 13** introduces the high-level design architecture for the new Multi-LiDAR SLAM system. The numbers one to three in the figure correspond to the stages of architecture as described in **Section 3.2.1**.

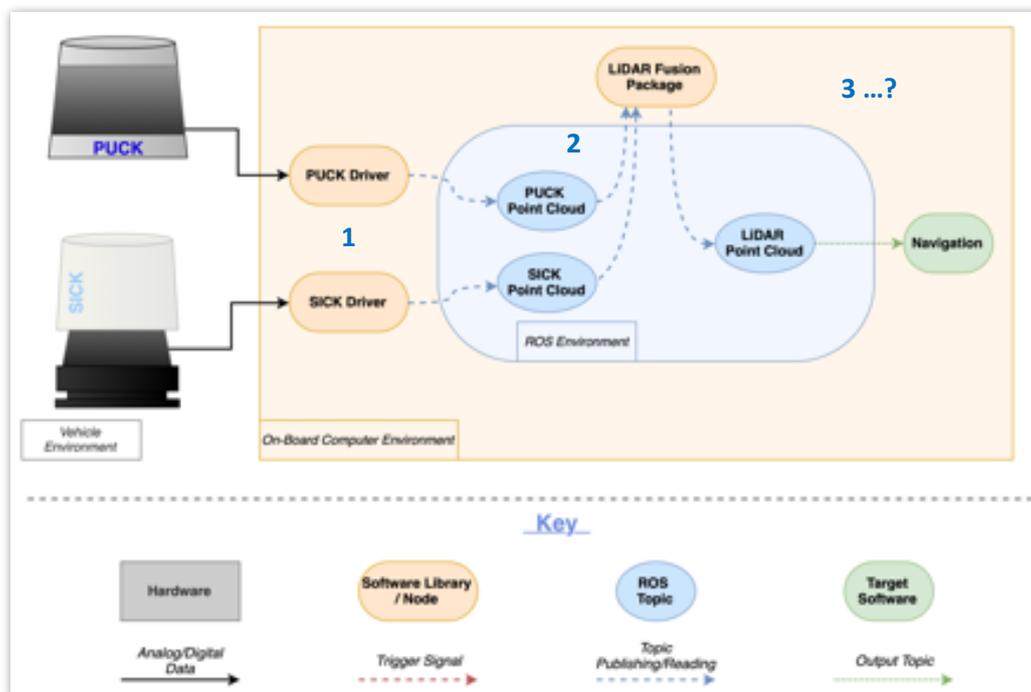


Figure 13 - Multi-LiDAR SLAM System Architecture

3.3.1 Stages of Multi-LiDAR SLAM Architecture:

1. A driver already exists for the SICK LiDAR, which is used for the cone avoidance demo on the SAE. This will have to be tuned to process points at a range that best suits the driving needs once the other LiDAR has been integrated. A ROS integrated driver must be constructed for the PUCK LiDAR to initialise the device, begin taking measurements and publishing the resulting point cloud. To address the synchronisation issues for point cloud fusion a timestamp should also be published along with each LiDAR point cloud. The output topic names can respectively be called *sick_cloud* and *puck_cloud*.
2. The next step in this design is to fuse the point clouds from the two LiDARs to form a cohesive output. As the two devices are installed at a fixed position relative to each other in space, a transformation matrix can be derived to accurately fit the point clouds together in correspondence to reality. To calculate this a distinct object that can be adequately resolved by both LiDARs (e.g. a cube) can be placed in front of the vehicle, and the point clouds transformed to fit the edges of the object together. An additional step would transform the resulting point cloud to set the origin of the coordinate system at the expected eye-level of the driver
3. This point is featured with a question mark in Figure 13 because the requirements for cone-track driving are still being defined and as such exists outside of the scope of this research project. Currently the cone driving demo uses the point cloud from the front-mounted SICK LiDAR to create a “local map” of the cones detected within the processing range. A global map has not yet been explored for the purpose of mapping an entire track with loop closure. For this final step the fused point cloud can be used as input to an open source LiDAR SLAM package like Google’s Cartographer, which supports ROS integration [34]. An example of Cartographer’s Mapping and Localisation capabilities is included in **Figure 14**.

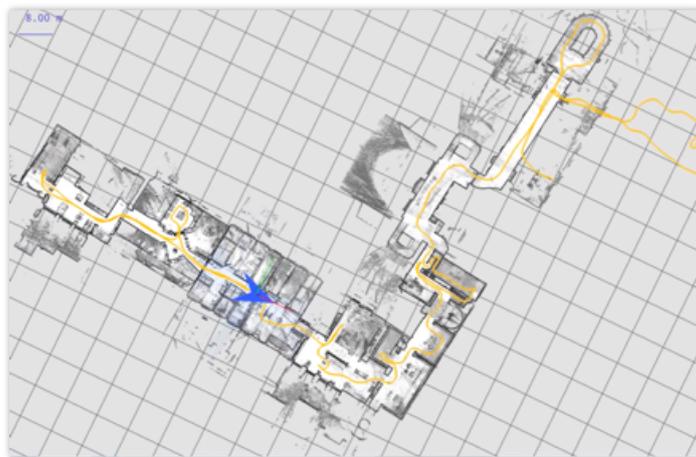


Figure 14 - Indoor LiDAR Mapping example using Cartographer

3.4 Mapping

Currently the ROS Architecture allows users to save the output topics from a testing session in the format of a *rosvbag*. In essence this can be used to save the trajectories and point cloud maps that are output ROS topics of the SLAM systems. With integration of the reactive cone driving demo a map of a cone-based racetrack could be built, saved as a *rosvbag*, and used to pre-process driving instructions for the next lap around the course using an advanced navigation node. However, navigation corrections using sensors would still be required for accurate navigation of the course during a second lap however. It should be noted that this high-level procedure is theoretical and lies outside the scope of this paper.

Other considerations for developing this function's architecture further are discussed below:

1. *rosvbag* files can be very large in size, occupying several gigabytes of memory even for a short two-minute drive. A high memory capacity will be needed on the on-board computer to store data from a series of test drives or tracks, even so the user will need to be selective with the maps to store on the hard drive.
2. Automatically recognising a track from a library of tracks is a difficult loop closure task given the vast amount of data and processing required. Specifying a starting point on a specific track is one strategy that could be considered to simplify loop closure. Fusion with the GPS sensor could also improve speed and accuracy of track recognition.

3.5 Resources: Hardware & Software

This section summarises the hardware and software resources used in this design and build research project.

Table 1 - Table of project resources

Resource	Purpose	Source/Cost
HARDWARE		
The SAE Car	Testing ground & build environment	Already in stock
Nvidia Jetson TX1	On-board-computer and control mainframe	Already in stock
(2x) FLIR Cameras	Visual Image Recording and Capture	Already in stock
SICK LiDAR	Immediate vicinity object and kerb detection	Already in stock
PUCK LiDAR	360° medium range object detection	Lent from project sponsor
(4x) Gigabit Ethernet cables	Data communication between Jetson and sensors	(2x) Already in stock (2x) - \$3.95 [35]
(2x) Hirose 6 Pin GPIO Cables	Power and General Purpose Input / Output for the FLIR cameras	Already in stock
470 Ohm Resistor	Pull-Up resistor for camera synchronisation signal	\$0.50 [36]
9-Terminal Bus Bar	Improved power connections for on-board electronics, with additional terminals for expansion	\$54.95 [37]
8-Port Gigabit Switch	Old switch damaged, additional for new hardware and future expansion	\$77.00 [38]
(3x) 2-pin Deutsch Connectors	Improving power connections for crucial sensors & equipment (sturdy, weatherproof)	(3x) - \$7.95 [39]
(2x) Hoosier 20.5" \varnothing – 6.5" W tyres and rims	Rear replacement wheels	Already in stock
SOFTWARE		
<i>Spinnaker SDK</i>	"New Generation" Camera API for programming cameras	Open Source [40]
<i>FlyCapture SDK</i>	"Previous Generation" Camera API for programming cameras	Open Source [41]
<i>Vision Stereo Calibration</i>	Camera Stereo Calibration package	Open Source [33]
<i>Open CV Stereo</i>	SGBM Depth Imaging package using OpenCV Library	Open Source [31]
<i>ORB SLAM2</i>	Mono, Stereo & RGB-D Visual SLAM Package	Open Source [42]
<i>VeloView</i>	Velodyne LiDAR Point Cloud Viewer	Open Source [43]
<i>Cartographer ROS</i>	LiDAR SLAM Package (ROS integrated)	Open Source [44]

4. ENABLING WORKS

This section briefly covers works completed by the author and throughout this research project that enabled the work within the main scope of this paper.

4.1 SAE Autonomous Test Drives

From the beginning of this project the new REV team undertook weekly test drives of the SAE to test the to-date capabilities of the SAE vehicle in order to scope the works that need to be done and comprehend the system's nuances. Testing generally consists running "demonstration" mode on the SAE, taking data required for projects and testing new capabilities. An example of autonomous testing, taking place on UWA's Riley Oval is pictured in **Figure 15**.



Figure 15 – SAE Autonomous Test Drive

For the thesis project in this paper, test drives were used to capture footage on cone tracks and UWA internal roads for testing with the SLAM system and subsystem designs.

4.2 Hardware Installation

As outlined in **Section 3**, several new pieces of hardware had to be installed onto the SAE vehicle to begin development of new features.

The first task undertaken was to redesign and organise the wiring of the 12V power system on the SAE. A DC-DC converter provides regulated 12V Power (from the main 50V battery) for the majority of electronic devices installed on the SAE, but the converter only features four terminals for connecting devices to power and ground. The existing solution was to "piggyback" connections to each other through connectors and soldered wires, which caused reliability issues early into this project and complicated troubleshooting. To address this a bus bar was installed behind the driver's seat as an extension to

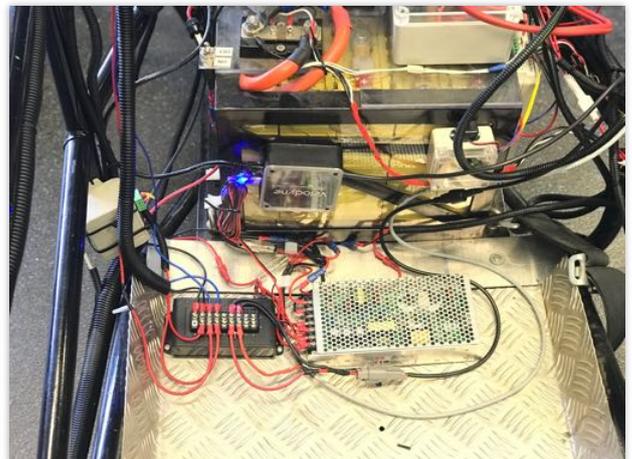


Figure 16 - Redesigned 12V Wiring

the DC-DC, featuring terminals for 9 devices and built-in fuses for short circuit protection. The results (depicted in Figure 16) improved connection reliability, organisation and wiring aesthetics.

Finally, the new hardware, including the Velodyne Puck™ (mounted by Jia Yu) and the 8-Port Gigabit Switch could be installed and wired.

4.3 Wheel Replacements

At the UWA Open Day on the 4th August the REV team showcased the SAE's Autonomous Demo, running the vehicle around a track and allowing visitors to ride aboard. By the end of the day the vehicle was performing erratically and emitting a squeaking noise as it drove. Further inspections showed that the rear wheel rims had deformed over time and with a small clearance to the mounting, had begun scraping against the mounting bolts. To address this the rear wheels were removed, the bolts sent to the UWA Mechanical Workshop to be machined down by two millimetres, and the rims and tyres replaced with a spare pair in stock at the REV workshop (**Figure 17**). After this the SAE was fit for test driving once again and the REV team's research projects could continue.



Figure 17 - SAE Rear Wheel replacement

4.4 Mono-Visual SLAM Preliminary Results

Preliminary testing was done with ORB SLAM2 to test its localisation and mapping capabilities with mono-visual inputs, providing a performance benchmark for *Stereo Visual SLAM*. Using the graphics-friendly workstation tower in the 3.13 Robotics Lab, rosbags of typical drives on UWA internal roads and cone track were played to publish image topics. ORB SLAM2 was configured and run in mono mode, publishing point cloud and pose data as it computed trajectory and environment from the images.

It was found that ORB SLAM2 could not identify sufficient features for cone driving, as footage of the vast oval contained too few features and cones were neglected due to their position in the frames. Fortunately, UWA internal roads provided better results, with a discernible map being constructed of the internal road located behind the physics building, as demonstrated in **Figure 18**.

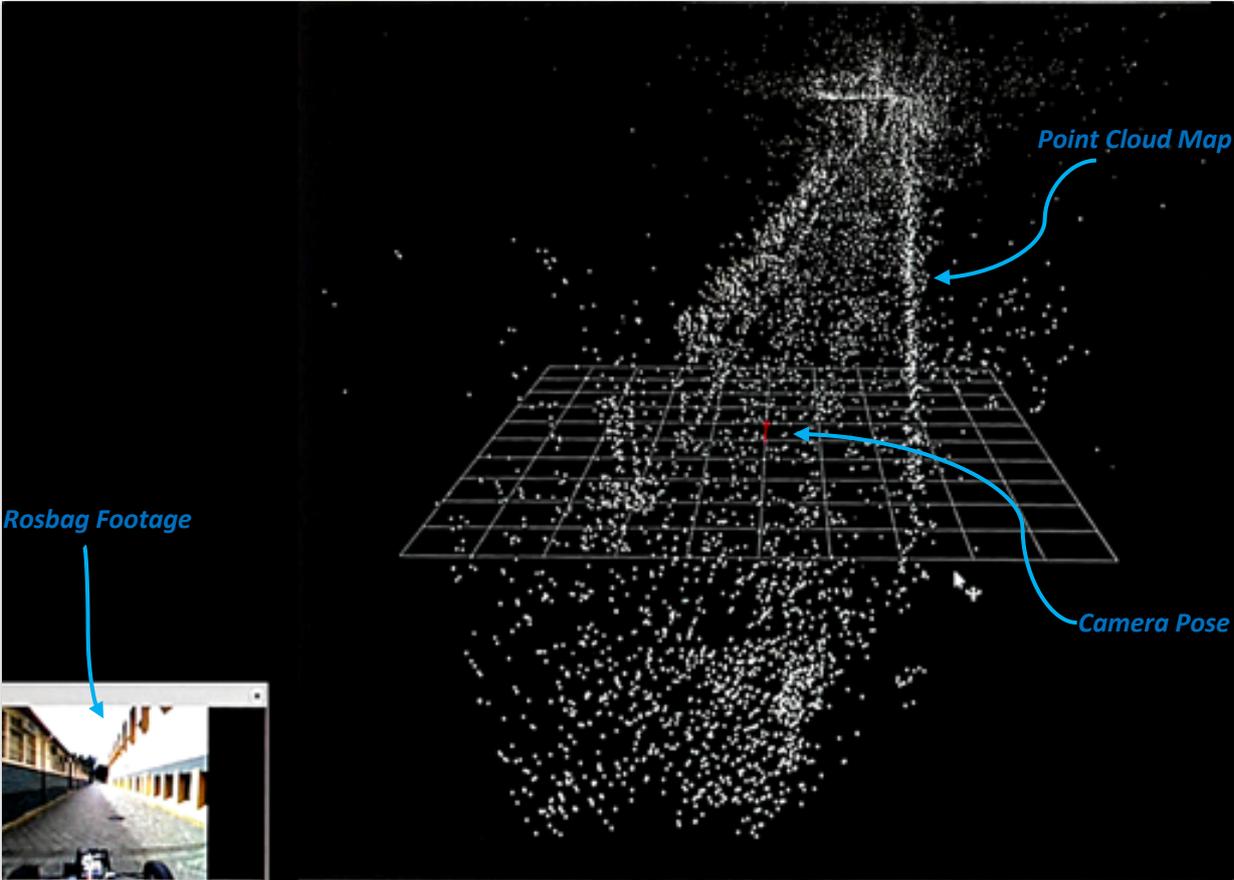


Figure 18 - Mono ORB SLAM2 Testing

5. DESIGN IMPLEMENTATION

This section covers the building and implementation phase of this research project. Here progress made on implementing each element of the project designs is discussed, including successes, issues encountered and alternative methods attempted.

As expected at the beginning of this project, implementation of the design posed several challenges along the way to achieving the project goals. Here, the progress and findings of implementation are documented step by step.

5.1 Stereo-Visual SLAM

5.1.1 Synchronised Capture

As covered in **Section 3.2**, synchronised capture with the left and right cameras is the first step to stereo imaging for real-time use. This is important because the SAE is designed to operate in dynamic environments, and a fraction of a second delay between left and right images captured could make a dramatic difference in the distances computed in the depth image.

As FLIR cameras are designed for machine vision uses, they are highly configurable and programmable and can perform tasks like triggered capture. The design to enable synchronised capture is outlined as follows:

1. Setting one camera as the **primary** and the other as the **secondary**, power is supplied through the 12V and ground pins of the GPIO cables. The GPIO Input pin of the secondary camera is connected to 3.3V through a pull-up resistor, and to the GPIO Output pin of the primary camera. Camera grounds are also connected to be common between the two cameras. See **Figure 19**.
2. The primary camera is configured to capture freely at a desired rate of 30 FPS, and pass an active-low signal (a *strobe*) for the duration of its exposure.
3. The secondary camera is configured to await a *trigger* signal from the primary camera before beginning to capture a frame
4. Both cameras are activated and set to run, where the secondary camera will capture frames in synchronisation with the primary camera's frames

FLIR offers two Software Development Kits (SDK) for controlling and programming the cameras: *FlyCapture2* and *Spinnaker* [45]. Spinnaker is the newer of the two SDKs with more updates and expansion planned in the future, and as such was originally chosen as the SDK with which to build the *camera_driver* module. However, Spinnaker claims to have limited support (i.e. no guarantees) when running on ARM boards, such as the Jetson. While simple image capture was possible, bugs would frequently occur using

the API and the desired results couldn't be fully met. As a result, *FlyCapture2* SDK was explored instead to implement the software.

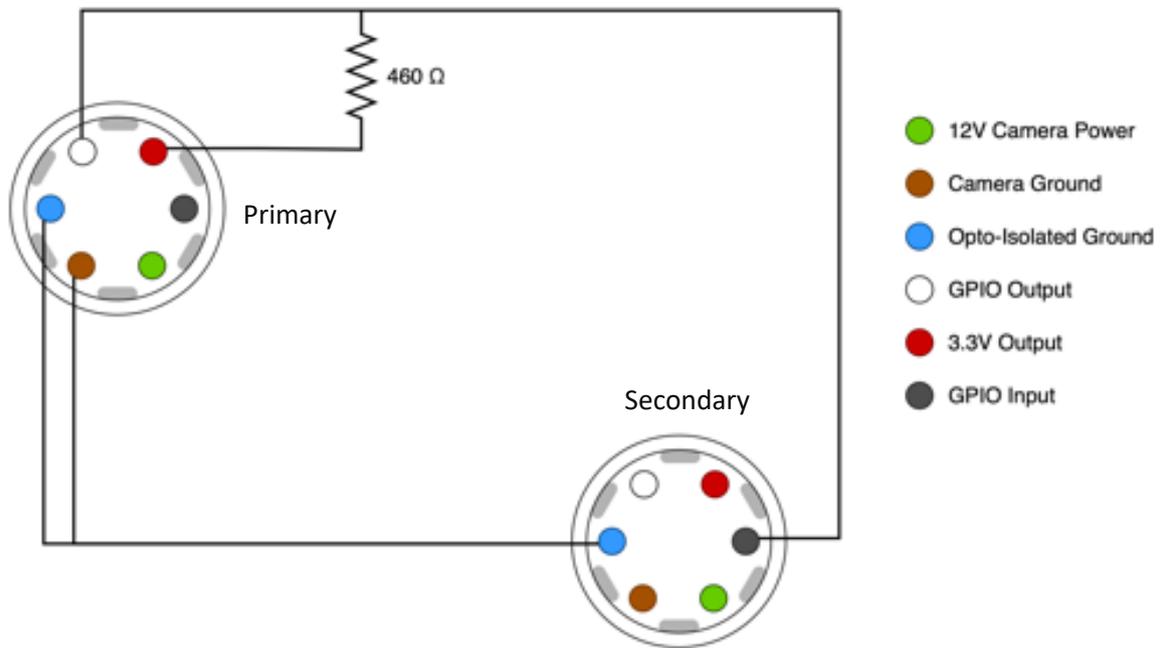


Figure 19 - Primary to Secondary Camera GPIO Connections

The first experiment performed to test the modules functionality involved measuring the strobe signal from the primary camera on an oscilloscope and is discussed further in **Section 6.1**.

One finding from this implementation was that the cameras could not properly process footage at full resolution, and to return results the height and width of each image had to be halved. Another finding was that the cameras could run freely by themselves and return image frames at the desired rate of 30 FPS, but when the synchronisation method was implemented, the framerate would drastically suffer often processing at 5 FPS. The exact cause is unknown, but the bitrate and buffer rate will have to be explored in the future. FLIR has been contacted for support with these issues for, but these issues have not yet been resolved at the time of writing this paper.

Once a method was configured to enable synchronised capture, a test was run to compare frames side-by-side using motion of a falling object. This experiment is discussed further in **Section 6.2**.

5.1.2 Camera Calibration

Once synchronised left and right images could be taken the next step in this process was to calibrate the cameras to correct for lens distortion and find the transformation matrix between the two camera's lines of sight.

The first method tested was using an open source package from *Justin Blaber Computer Vision* [25], which uses a python supported version of *Spinnaker API* to capture images from multiple FLIR cameras and calibrate them using a checkerboard pattern like in **Figure 20**. Unfortunately, some of the pre-requisite packages were not compatible with the ARM architecture of the Jetson, and a different method was attempted.

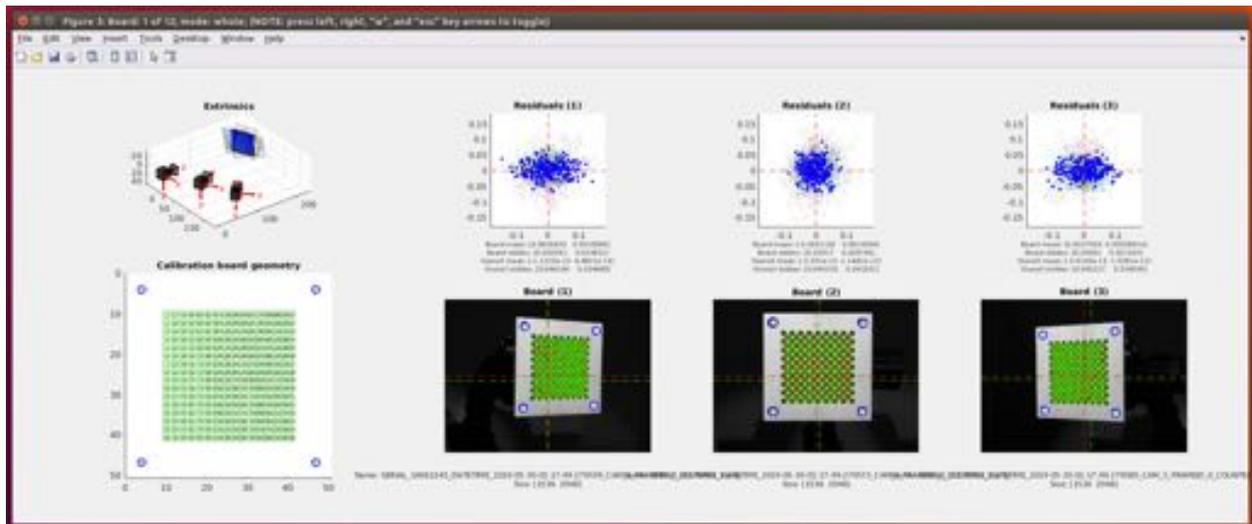


Figure 20 - Example of Checkerboard Calibration

A second open source package from GitHub user *super148666* [33] was trialed for camera calibration. This method ran into a different issue, where the OpenCV packages on the Jetson appear to have been corrupted, and efforts to try and fix them and reinstall them were unsuccessful. Attempts were made to run this package from an Ubuntu virtual machine from a Macintosh Laptop, but issues were occurring with the virtual machine and attempts to at the time of writing this paper have been unsuccessful.

5.1.3 Depth Imaging

With an uncalibrated camera depth image results would not yet be useful for autonomous driving through SLAM, but nevertheless a depth imaging package was trialed using the uncalibrated cameras.

Left and right images were taken from outside the REV workshop to capture a simple scene that the SAE might typically encounter (

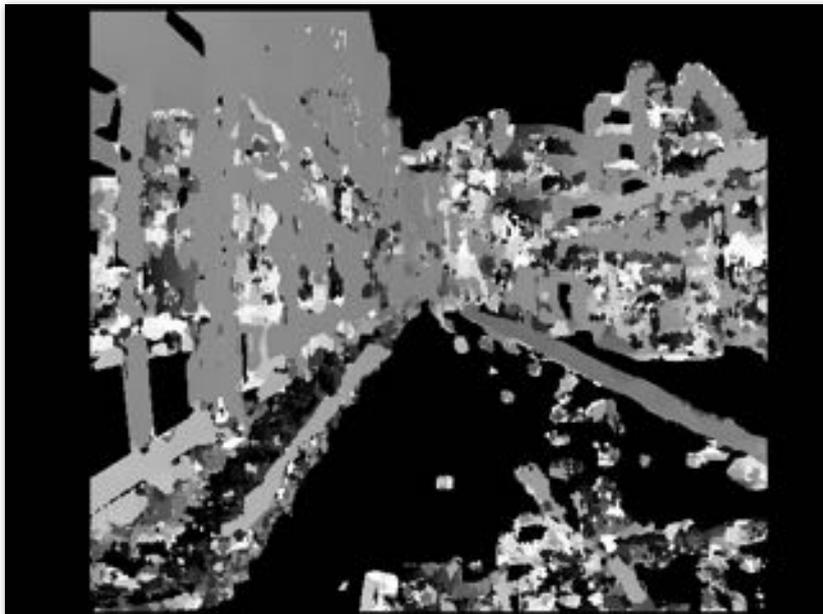
Figure 21). The focus was adjusted to best fit the scene, and baseline distance parameter measured at 8cm



Figure 21 - Left and Right Test Images

Using an Open Source SGBM python method [31] a depth image was computed as pictured in

Figure 22. Even without calibration the algorithm appears to succeed in detecting features on the image, where road markings, buildings and windows are visible. Depth imaging contains several dead spots, but with improved image and camera better depth are expected.



synchronised
performance
calibration
image results

Figure 22 - Depth Image

5.2 Multi-LiDAR SLAM

5.2.1 Velodyne Point Cloud Testing

With the help of Jia Yu, preliminary tests of the Puck's™ LiDAR imaging abilities were able to be tested in the REV workshop. **Figure 23** shows the results obtained from the newly installed LiDAR as it creates a point cloud map of the REV workshop. Due to issues with OpenCV on the Jetson, integration with ROS has not yet been possible at the time of writing this paper.

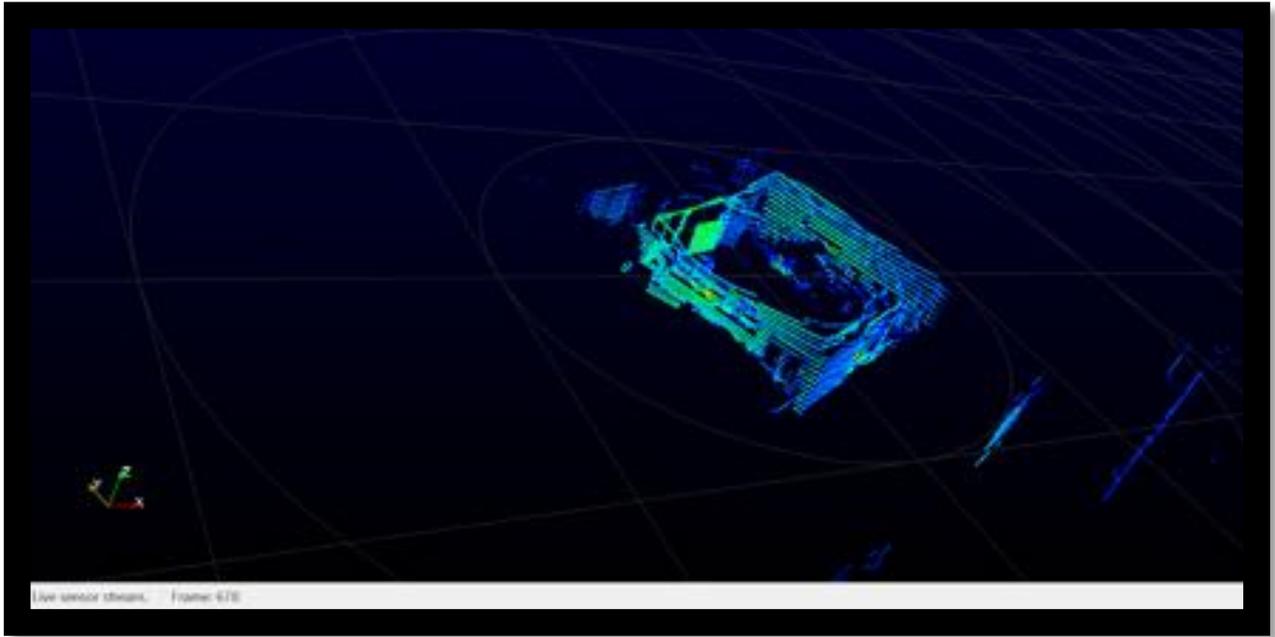


Figure 23 - Puck LiDAR Imaging using VeloView

6. EXPERIMENTS

6.1 Strobe Signal Testing

Using a digital oscilloscope, the strobe signals programmed to transmit from the primary camera are visualised and measured. Free running the camera at 30 FPS results in a rectangular pulse signal as seen in **Figure 24**, with the expected frequency of 30Hz. As this is an active-low signal, the strobe will usually pass a high signal, and only pass low during exposure while capturing an image. Qualitative tests showed that the pulse-width of the low signal increased as the exposure time was increased for the primary camera, as expected.

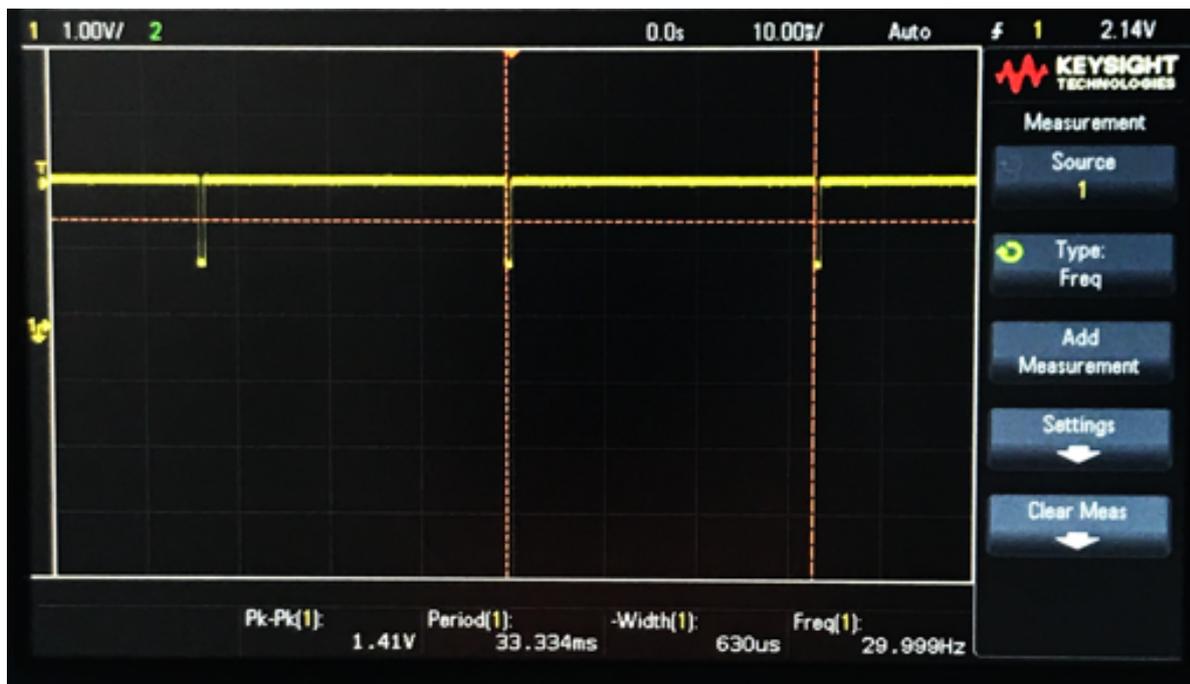


Figure 24 - Strobe Signal as visualised on oscilloscope

6.2 Frame Comparison Drop Test

Despite the low framerate found during the build phase of this project, the functionality of the synchronised capture was tested by dropping a bright coloured ball in front of a checkerboard-ruler to measure the disparity of the balls position between left and right frames, as seen in **Figure 25**.

An example of the side by side comparisons made is demonstrated in. Running this experiment a number of times it was found that the average ball position disparity between the left and right frames was 1.1cm, but this experiment had a high margin of error due to the camera resolution, frame rate, and the noticeable motion blur. The standard deviation of disparity was significantly high, at 7.7cm, and as such further investigation is recommended once previously mentioned issues have been addressed.

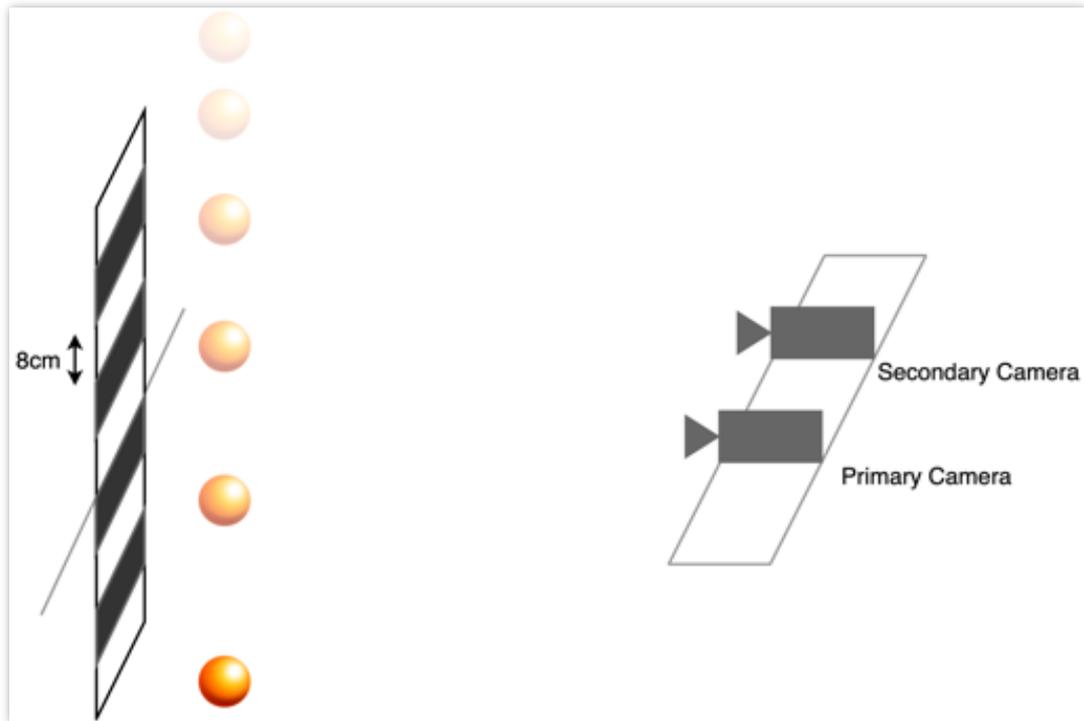


Figure 25 - Drop Test Setup

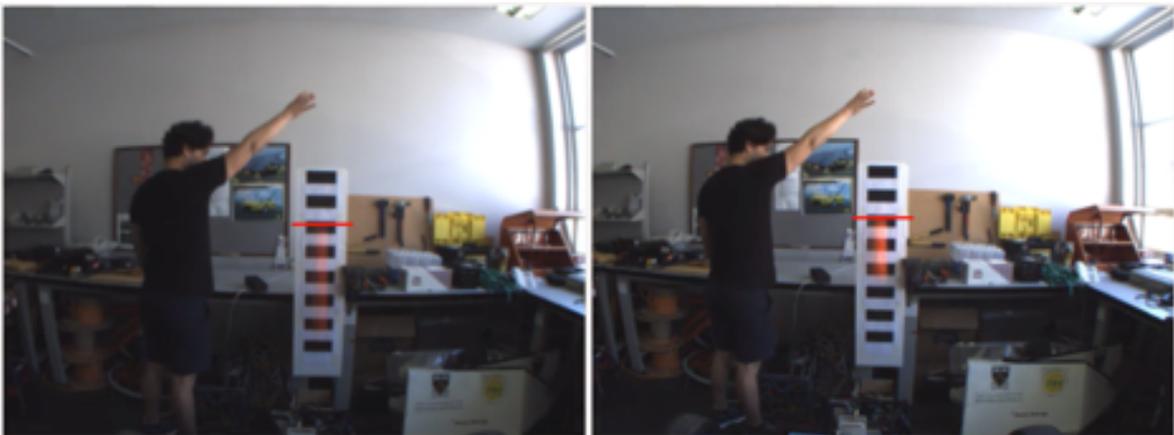


Figure 26 - Drop Test Example Frames and disparity

7. CONCLUSIONS

This paper has presented a detailed design incorporating hardware and software elements for both an advanced Visual SLAM system and an advanced LiDAR SLAM system for autonomous driving on the REV SAE Vehicle. Hardware needed for implementation of these designs has been fully installed, most software elements and frameworks of the design have been implemented and tested to progress the project to meet the REV team project goals. Adjustments need to be made to some elements of the design architecture, and fine tuning will still be required for the design to reach its final real-time-driving implementation.

The REV team is in the process of upgrading the on-board-computer with an Nvidia AGX, which is expected to address software and runtime issues. The existing designs outlined in this paper should be tested and continued with the upgraded computer, and the latest compatible versions of Ubuntu and ROS.

360-degree LiDAR imaging is ready to use, and integration with ROS will be possible as soon as new on-board-computer, the AGX, is mounted, integrated and configured from a clean slate with the packages required including OpenCV.

Synchronised capture has been implemented with the left and right cameras, albeit with a less-than-ideal framerate, and image resolution. This will likely be improved when implemented on the new on-board-computer and upgraded graphics card.

Finally, a robust architecture and technical framework is documented as a result of this paper, allowing future generations of the REV team to tune, expand and improve on the advanced SLAM system designs.

7.1 Future Work and Recommendations

More specifically, recommendation for future generations working on SLAM and computer vision on the SAE include:

1. Having a conservative project scope and putting a heavy focus on implementation of design. For example, focusing on Multi-LiDAR SLAM and studying the ideal LiDAR placements and configurations for racing and performance style driving
2. Integration of mapping with path planning to construct a demo for “explorative mapping” where a racecourse is mapped during a slow lap and performed at competition speeds during subsequent laps.
3. There is evidence to suggest that the cameras may not be working ideally. Firmware should be reinstalled, FLIR should be contacted for further support in case of a faulty camera, and if these don't fix the issues a next generation camera like the USB Blackfly S is recommended
4. Exploration of different depth image algorithms & methods to dedicate computational use of the GPU for this task.

8. REFERENCES

- [1] T. Kelliher, "Real Time Obstacle and Road Edge Detection for an Autonomous Formula SAE-Electric Race Car," University of Western Australia, Perth, 2018.
- [2] C. Zhang, "Integration of Cone Detection, Visual SLAM and Lane Detection for Real-time Autonomous Drive," University of Western Australia, Perth, 2018.
- [3] Quanergy Systems Inc., "Mapping," 2016. [Online]. Available: <https://quanergy.com/mapping/>. [Accessed October 2019].
- [4] C. Stachniss, "Robotic Mapping and Exploration," *Springer Tracts in Advanced Robotics*, vol. 55, pp. 93-115, 2009.
- [5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99-110, 2006.
- [6] J. Fuentes-Pacheco and J. Ruiz-Ascencio, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55-81, 2015.
- [7] R. Mur-Artal, J. D. Tardos, J. M. Montiel and D. Galvez-Lopez, "ORB-SLAM2," 13 January 2017. [Online]. Available: https://github.com/raulmur/ORB_SLAM2. [Accessed 11 April 2019].
- [8] J. Engel, J. Stückler and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015.
- [9] M. Bezemek, "It Began With a Race...16 Years of Velodyne LiDAR," 2 January 2017. [Online]. Available: <https://velodynelidar.com/newsroom/it-began-with-a-race/>. [Accessed 11 April 2019].
- [10] F. Schuster, C. G. Keller, M. Rapp, M. Haueis and C. Curio, "Landmark based radar SLAM using graph optimization," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, 2016.
- [11] S. Birch, "Volvo, Uber share the ride to safety with autonomy," *SAE International - Automotive Engineering*, pp. 8-10, October 2016.
- [12] SAE, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (J3016)," SAE International, 2018.
- [13] B. WASEF, "2019 Audi A8 L Review | Brilliant engineering in an unassuming wrapper," 16 October 2018. [Online]. Available: <https://www.autoblog.com/2018/10/16/2019-audi-a8-l-review-first-drive/>. [Accessed 10 April 2019].
- [14] J. Hecht, "Lidar For Self-Driving Cars," *Optical & Photonics News*, pp. 28-33, January 2018.

- [15] L. Laurensen, "The Tech That Won the First Formula Student Driverless Race," *IEEE Spectrum*, 16 August 2017. [Online]. Available: <https://spectrum.ieee.org/cars-that-think/transportation/self-driving/the-tech-that-won-the-first-formula-student-driverless-race>. [Accessed 11 April 2019].
- [16] M. Levoy, E. Talvala, K. Dektar and A. Adams, "Variables that affect exposure," 29 February 2012. [Online]. Available: <https://graphics.stanford.edu/courses/cs178/applets/exposure.html>. [Accessed 12 October 2019].
- [17] T. Taketomi, H. Uchiyama and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSI Transactions on Computer Vision and Applications*, vol. 9, no. 16, 2017.
- [18] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Visual Odometry / SLAM Evaluation 2012," 2012. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_odometry.php. [Accessed October 2019].
- [19] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, 2012.
- [20] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, Barcelona, 2011.
- [21] J. D. Tardós, R. Mur-Artal and J. M. Montiel, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147 - 1163, 2015.
- [22] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255 - 1262, 2017.
- [23] P. Corke, "Principles of Stereo Vision," Queensland University of Technology, 2017. [Online]. Available: <https://robotacademy.net.au/lesson/principles-of-stereo-vision/>. [Accessed October 2019].
- [24] FLIR® Systems Inc, "Bumblebee®2 FireWire," 2019. [Online]. Available: <https://www.flir.com/products/bumblebee2-firewire/?model=BB2-08S2C-60>. [Accessed October 2019].
- [25] J. Blaber, "Multiple Camera Calibration," 12 July 2018. [Online]. Available: <http://justinblaber.org/camera-calibration-application/>. [Accessed June 2019].
- [26] J. Blaber, "Camera calibration toolbox," 2019. [Online]. Available: https://github.com/justinblaber/camera_calib. [Accessed June 2019].
- [27] MathWorks®, "What Is Camera Calibration?," MathWorks®, 2019. [Online]. Available: <https://au.mathworks.com/help/vision/ug/camera-calibration.html>. [Accessed August 2019].
- [28] MathWorks®, "Stereo Camera Calibrator," 2019. [Online]. Available: <https://au.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html>. [Accessed August 2019].

- [29] H. Hirschmüller and D. Scharstein, "Evaluation of Cost Functions for Stereo Matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, 2007.
- [30] MathWorks®, "FPGA Implementation of Stereo Disparity using Semi-Global Block Matching," MathWorks®, 2019. [Online]. Available: <https://au.mathworks.com/help/visionhdl/examples/stereoscopic-disparity.html>. [Accessed October 2019].
- [31] T. Samart, "Open CV Stereo – Depth image generation and filtering with python 3+, ximgproc and OpenCV 3+," 9 July 2017. [Online]. Available: http://timosam.com/python_opencv_depthimage. [Accessed September 2019].
- [32] Open Source Robotics Foundation, "About ROS," Open Source Robotics Foundation, [Online]. Available: <https://www.ros.org/about-ros/>. [Accessed February 2019].
- [33] C. Zhang, "super148666/vision," 2018. [Online]. Available: <https://github.com/super148666/vision>. [Accessed August 2019].
- [34] The Cartographer Authors, "Cartographer ROS Integration," 2018. [Online]. Available: <https://google-cartographer-ros.readthedocs.io/en/latest/>. [Accessed July 2019].
- [35] Altronics, "Blue 0.5m Cat6a SSTP Ethernet Patch Cable," 2019. [Online]. Available: <https://www.altronics.com.au/p/p1630a-blue-0.5m-cat6a-sstp-ethernet-patch-cable/>. [Accessed June 2019].
- [36] Altronics, "4R7 0.25W 5% Carbon Film Resistor PK 10," 2019. [Online]. Available: <https://www.altronics.com.au/p/r7018-4r7-0.25w-carbon-film-resistor-pk-10/>. [Accessed June 2019].
- [37] Altronics, "9 Way Terminal Block Bus Bar," 2019. [Online]. Available: <https://www.altronics.com.au/p/s9757-cop-security-9-way-terminal-block-bus-bar/>. [Accessed June 2019].
- [38] Harvey Norman, "Netgear GS108 8 Port Gigabit Switch," 2019. [Online]. Available: <https://www.harveynorman.com.au/netgear-gs108-8-port-gigabit-switch.html>. [Accessed June 2019].
- [39] Altronics, "2-pin 13A IP67 Weatherproof Connector," 2019. [Online]. Available: <https://www.altronics.com.au/p/p7882-deutsch-2-pin-13a-ip67-weatherproof-connector/>. [Accessed June 2019].
- [40] FLIR® Systems, "Spinnaker SDK," 2019. [Online]. Available: <https://www.flir.com.au/products/spinnaker-sdk/>. [Accessed July 2019].
- [41] FLIR® Systems, "FlyCapture SDK," 2019. [Online]. Available: <https://www.flir.com.au/products/flycapture-sdk/>. [Accessed July 2019].

- [42] R. Mur-Artal, J. Tardós, J. M. Martínez Montiel and D. Galvez-Lopez, "ORB-SLAM2 Repository," October 2017. [Online]. Available: https://github.com/raulmur/ORB_SLAM2. [Accessed March 2019].
- [43] Velodyne Lidar, 2019. [Online]. Available: <https://www.paraview.org/VeloView>. [Accessed July 2019].
- [44] Google Cartographer, "Cartographer ROS Repository," August 2018. [Online]. Available: https://github.com/googlecartographer/cartographer_ros. [Accessed August 2019].
- [45] FLIR® Systems, "Blackfly GigE," 2019. [Online]. Available: <https://www.flir.com.au/products/blackfly-gige/>. [Accessed March 2019].
- [46] J. Blaber, "Multiple Camera Calibration," 12 July 2018. [Online]. Available: <http://justinblaber.org/camera-calibration-application/>. [Accessed July 2019].