THE UNIVERSITY OF WESTERN AUSTRALIA

The University of Western Australia
School of Electrical, Electronic and Computer Engineering

Integration of Cone Detection, Visual SLAM and Lane Detection for Real-time Autonomous Drive

Chao Zhang

21435393

Final Year Project Thesis submitted for the degree of Master of Professional Engineering.
Submitted 21th Oct 2018

Supervisor: Professor Dr. Thomas Bräunl

Word Count: 6273

Table of Content

Index of Tables

Index of Figures

Abstract

Image processing is one of the critical components of the software frameworks for autonomous vehicles. Compared with other sensors such as LiDARs, GPS, and IMU, cameras generally provide richer information with a lower unit price. However, the processing of the data is much more complicated and has higher requirements on computation power for real-time application. In this paper, the author presents a high-level integration of visual software solution that combines cone detection, visual SLAM and lane detection for real-time autonomous driving application, with optimisation on architecture for improving runtime performance. This image processing stream is part of the REV SAE autonomous program. The framework is implemented as a node which communicates with other modules in the system under the Robot Operating System (ROS). The testing results are collected from both the simulation system and field driving.

1 Introduction

1.1 Introduction and Project Goals

The Autonomous SAE vehicle project aims to develop an ultimate autonomous driving system for REV project's 2010 SAE Electric car. The SAE vehicle was converted into a fully electric vehicle in 2010. Also, the full drive-by-wire system was constructed in 2013. The first attempt to have the SAE navigating without human inference is the waypoint navigation using Global Positioning System (GPS) in 2015. In 2017, the focus of the team was shifted toward autonomous driving with various streams on research topics such as Light Detection and Ranging (LiDAR) processing, vision processing, sensor fusion, and localisation. However, most of the research outcomes were not fully tested and integrated with the SAE control system. In other to avoid the same issue this year, the research topic for this project focuses more on the integration with the entire software and hardware system.

Moreover, the software of the Autonomous SAE car is being completely rewritten in this year and is concentrating on two use-case scenarios. One is the Formula-SAE Autonomous competition, which uses a race track set by two rows of cones for the vehicle to drive through. The other is the automatically detect and drive the internal roads of the University of Western Australia (UWA).

This project will be done by the Autonomous SAE team in this year. As shown by the graph below, the high-level architecture of the SAE Autonomous Driving System consists of six core topics. The team members of the Autonomous SAE team, including Craig Brogle, Timothy Kelliher, William Lai, Patrick Liddle, and Chao Zhang, will focus on different topics and complete this project together.

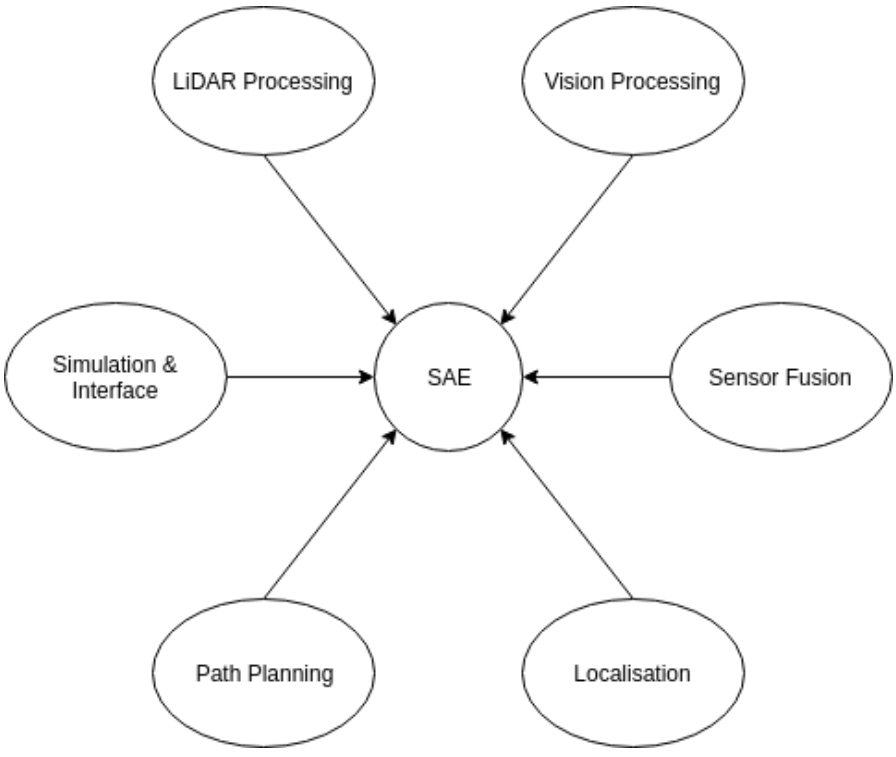

Figure 1Research topics in 2018

The thesis proposed by Chao Zhang in this proposal aims to design a real-time vision solution toward these two use-case scenarios with following goals:

- Provide localisation of the vehicle to sensor fusion
- Provide cone location to the SAE path planner
- Provide lane line to the SAE path planner
- Real-time performance with above functions

1.2 Problem Identification

In order to achieve the goals of the project this year, there are three primary outcomes required from this project, which are the location of SAE, the location of cones and drivable area. Also, at the same time, the solution must be able to run at real-time with a minimum requirement of 10 Frame Per Second (FPS) on Nvidia Jetson TX1 [1] (TX1).

1.2.1 Localisation of SAE

Simultaneous Localisation and Mapping (SLAM) has been a hot research topic in the last two decades in the Computer Vision and Robotics communities and has recently attracted the attention of high-technological companies since SLAM is a critical component of an autonomous driving system. [2] Visual SLAM gains increasing interests as it advantages on cost compared with other sensory modalities. In order to acquire accurate localisation for the SAE, visual odometry is required to be fused with other sensors such as wheel encoder, LiDAR and Inertial Measurement Unit (IMU). In this project, the author focuses on extracting the estimated location of the SAE from a single monocular camera using visual SLAM algorithm.

1.2.2 Detection of Cones

Another critical component in an autonomous driving system is object detection, and in this case, the cone detection specifically. In order to generate a drivable path following two sets of cones, the location of each cone has to be identified correctly. The first thing is to recognise the cones from the camera successfully. Then the position of the cones in the image needs to be transformed into a real-world position related to the vehicle itself.

1.2.3 Detection of Lane

The vision has been proven as an excellent tool for detecting drivable area. For a fully autonomous driving system, the vehicle needs to identify the drivable area to plan the future path safely. In a well-constructed area, there are always some lane markings to guide the driver. A vision system should be able to detect the lane markings and provide the drivable area for path planner.

1.3 Structure of the Document

The section “1 Introduction” is the introduction of the entire project including project goals, project background and problem identification.

The section “2 Literature Review” is an overview about the related work presented by other academic researcher in the field.

The section “3 System Overview” provides a summary to the entire system on SAE vehicle in this project.

The section “4 Camera Setup” presents the current sensor setting for vision application, including calibration and data collection.

The section “5 Localisation of SAE” describes the method used in this project to perform SLAM using visual sensor data.

The section “6 Cone Detection” shows the methods used in this project for cone detection, and the comparison between the traditional approach and Convolutional Neural Network (CNN) approach.

The section “7 Lane Detection” demonstrates the procedure for lane detection and the improvement since last year.

The section “8 Distance Extraction” describes the approach to extract distance in real world from image frame.

The section “9 Experiments” presents the experiments from both field test and simulation for the visual system discussed in this document.

The section “10 Conclusion” concludes the results and findings demonstrated in this document, and lists several future research directions.

2 Literature Review

In order to achieve autonomous driving, a self-navigation is essential. The navigation can be divided into three key topics: mapping, localisation and path planning. [3] Initially, the research on mapping and localisation were independent. However, it was soon found that they are actually highly dependent with each other. A precise localisation requires an accurate map and to build an accurate map the localisation for every object in the scenes is critical. The name of SLAM was first used in 1985. [4] SLAM is the process for an entity to construct a global map for the visited surrounding and at the same time correct its own position in the map concurrently. Also, when the camera is used as the only sensor, the name visual SLAM is used. There are many challenges in the actual application of visual SLAM, including highly dynamic environments, too high or too low intensity, occlusion of the sensor and blur caused by movement. [4] However, with the considerable advantage in camera cost compared with long-range LiDAR and the fast development in the graphics processing unit, visual SLAM gains increasing interests due to its rich information and affordability.

Based on the KITTI Vision Benchmark Suite, ORB SLAM2 is the fastest open-source visual SLAM algorithm with a runtime of only 0.03s. [5] ORB SLAM2 uses Oriented FAST and Rotated BRIEF (ORB) feature for all modules including mapping, tracking and place recognition, which results in excellent efficiency and runtime performance. [6] ORB feature is a fast binary descriptor based on Binary Robust Independent Elementary Features (BRIEF) with an extra feature as rotation invariant and noise resistance to some range. [4] ORB SLAM2 has three parallel thread to perform three main tasks in visual SLAM: tracking, local mapping and loop closure. [6]

The study of object detection aims to recognise the target objects with the theories and methods of image processing and pattern recognition, locate the specific objects in the images. [5] The performance of object detection is affected by many factors, such as the complexity of background, disturbance from noise and movement, occlusion, low-resolution, scale and rotation changes. [7] The traditional method in object detection is using hand-crafted features, such as Histogram of Oriented Gradient (HOG) [8], Scale Invariant Feature Transform (SIFT) [9], Speeded Up Robust Feature (SURF) [10], BRIEF [11] and ORB [12] . These hand-crafted features provide a better runtime performance when only runs on Central Processing Unit (CPU). However, as the development on Graphics Processing Unit (GPU) progresses, parallelisation highly reduces the runtime for training a customised filter, which is called convolutional filter. By feeding the feature vectors from convolutional layers into a fully connected neural network, a classic deep learning structure called Convolutional Neural Network (CNN) is created. The CNN has achieved great success in object recognition and detection. The state of art model including AlexNet [13], You Only Look Once (YOLO) [14], Regional Convoluational Neural Network (R-CNN) [15] achieved much higher accuracy and better and better runtime performance with GPU acceleration. Also, the research in segmentation using CNN has been done by Cambridge as SegNet [16]. And with better optimisation and enhancement

for mobile application, ENet [17] achieved similar accuracy and far better runtime performance.

3 System Overview

This section introduces our current software framework for the SAE race car. A flexible and modular software architecture was used due to the requirements on resiliency, flexibility, extensibility and simple integrations. This allows for highly decoupled software to be developed, with each component or series of components needing only to conform to the expected message type on the input and output topics. In the early this year, the team decided to migrate the existing old software stack into Robot Operating System (ROS) [18] framework. [19]

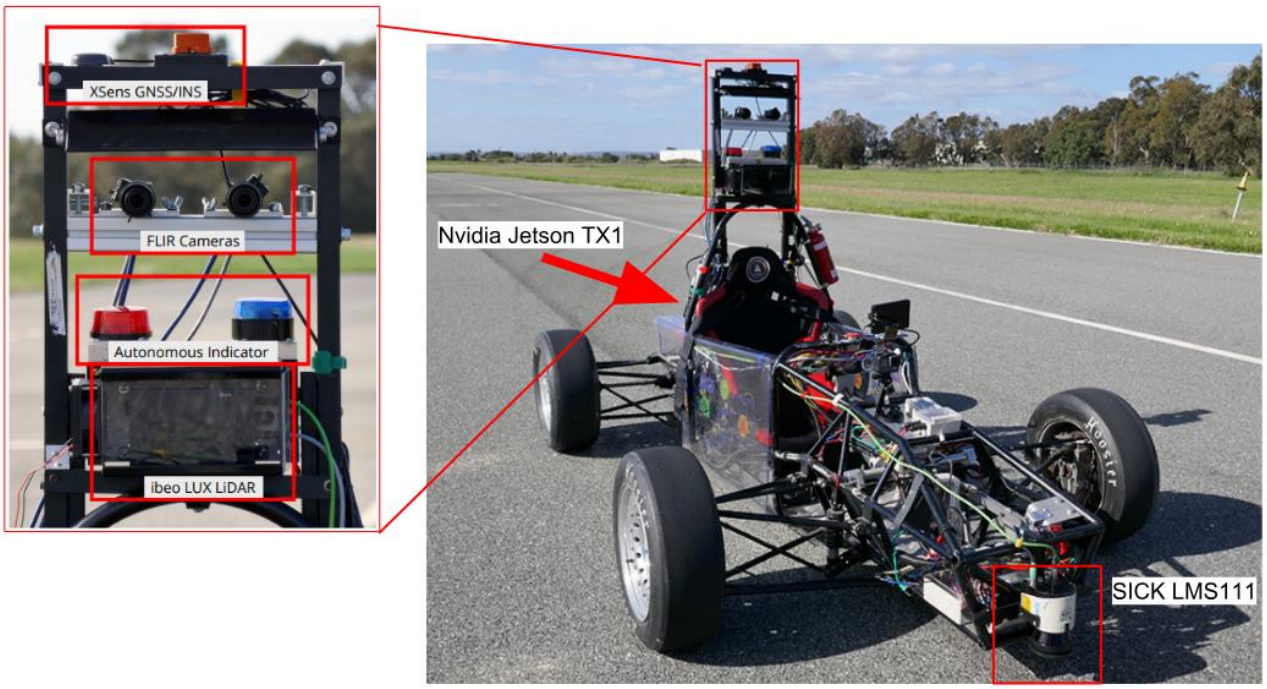

Figure 2Hardwares on the SAE vehicle

3.1 System Architecture

The system has three critical controllers, which are the low-level controller, high-level controller and safety supervisor respectively.

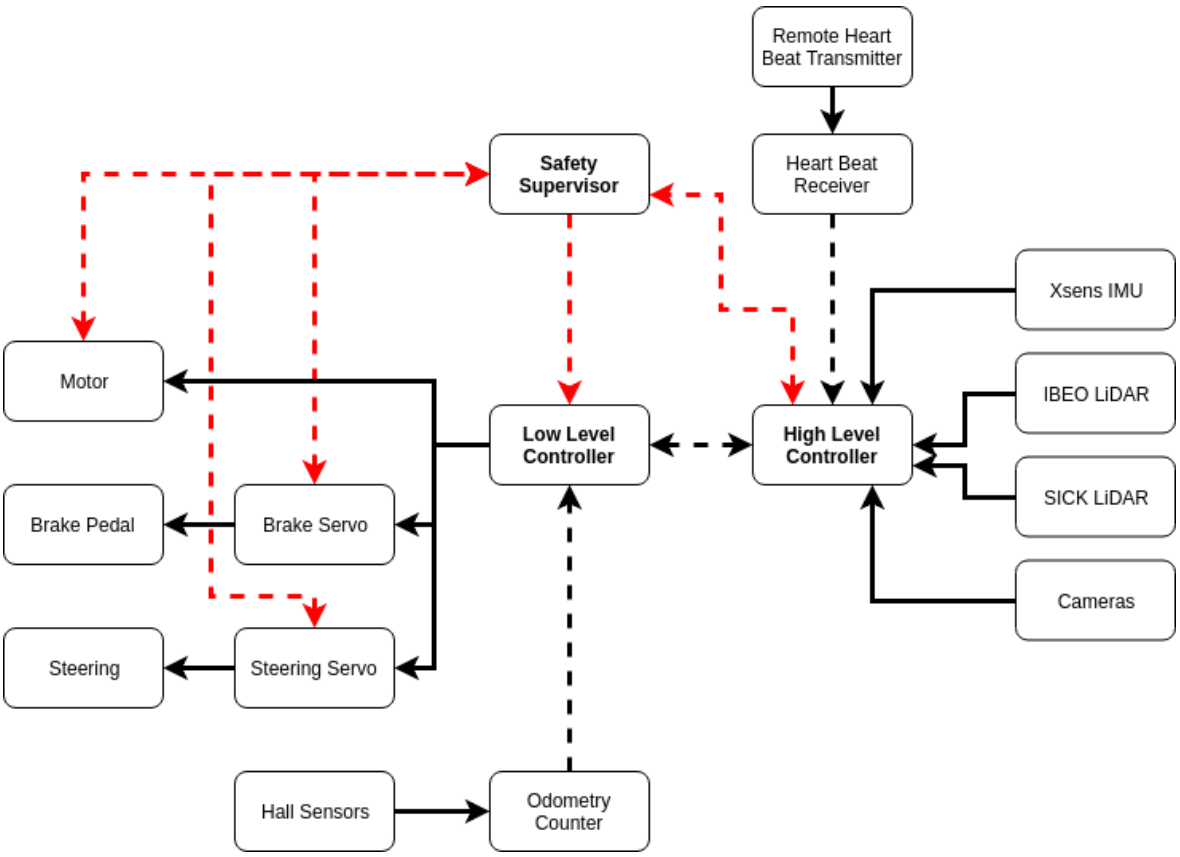

Figure 3System Architecture

The low-level controller is implemented on ARDUINO MEGA [20]to handle all drive-by-wire system. Acceleration, steering, and brake are controlled by the low-level controller with Proportional-Integral-Derivative (PID) control. Serial communication is set up between the low-level controller and high-level controller for communication of control command, the status of actuators and wheel odometry.

The high-level controller is running on the onboard computer TX1 mounted at the back of the seat. All high-level modules are running on this system. The core control node collects status information from the low-level controller and publishes this information as topics for other modules. Also, it subscribes to control topics of the SAE vehicles and passes these speed command into the low-level controller.

Safety supervisor is an independent system regarding the low level and high-level controllers. It monitors the status of the actuators directly at the lowest level. Any unexpected signal from these actuators triggers emergency brake on both high level and low-level systems. Moreover, the safety supervisor cut off the power supply toward the motor when emergency brake engaged.

3.2 ROS

ROS provides a robust communication infrastructure for inter-process communication as a middleware. Message passing is handled anonymously by master core node in a publish/subscribe manner. Also, it allows the user to record and playback the messages freely without the dependency on the original environment. ROS also provides easy to use distributed parameter system for configuration of the program, which provides a more flexible method to introduce configurable parameters without the use of complicated command line parameter passing.

The core of ROS is licensed under the standard three-clause Berkeley Software Distribution (BSD) license [21], which is a very permissive open license. It allows reuse in commercial and closed source products, and is compatible with most of the open licenses.

Due to its popularity in the open-source community, ROS has been adapted into many open-source robotics research projects. Therefore, there are plenty of the open-source packages compatible with ROS framework available, covering most common toolset for robotics researching, including 3D visualisation of data, common sensor drivers, and library for the coordinate transform. These resources were identified as one of the key benefits for this software migration.

The team migrated the existing code base of the high-level software system to use the ROS framework in early 2018 to reduce the development complexity of the software system. ROS provides low-level device control, implementation of commonly used tools, message passing between processes, and package management [22]. The use of ROS allows each research topic to have their own independent program. These programs then communicate with other modules through message topics handled by the master broker, making the integration of the entire system much more manageable. Using ROS, therefore, simplifies the integration process for each individual module in the system. By defining the topic information for messages to communicate, the individual nodes can work together without too much effort in integration. In general, ROS is a robust and flexible framework which is suitable for autonomous vehicle development. More specifically, existing modules in our system presented in [23], including modules for logging, web server and serial communication were replaced with their equivalent ROS packages, as they are often more stable and better supported. All existing messages from the system are converted into ROS messages. The testing for the individual modules therefore only requires minimal changes to the core software. The ROS version used on the SAE vehicle is currently ROS Kinetic Kame running on Ubuntu 16.04 Long Term Support (LTS) which is long-term supported until April 2021.

3.3 OpenCV

Open Source Computer Vision (OpenCV) [24] is a popular open-source library for image processing. It provides efficient implementation of various image processing operation used in this project, including camera calibration, video recording, image pixel level manipulation, image transform and feature detection. This library is included in ROS as an

internal package. The implementation of almost all algorithm used in this project requires dependency on OpenCV package.

OpenCV is licensed under standard three-clause BSD license, which is identical to the license for ROS.

4 Camera Setup

The camera used in this project was upgraded from monocular web camera to industrial grade cameras with stereoscopic setup.

A pair of FLIR Blackfly GigE [25] cameras are mounted on the vehicle's frame to perform tasks related to visual navigation, such as semantic segmentation and visual odometry. These cameras are fitted with Fujinon f/1.2, 2.8–8 mm wideaperture varifocal lenses [26], and are individually capable of capturing a wide field of view. To suit our application, these cameras use 1.3MP 1/3" global shutter Charge-Coupled Device (CCD) image sensors that will not be affected by any distortions caused by the rolling shutter effect [27]. The cameras are connected to a Gigabit Ethernet switch that connects to the TX1, interfacing them through customised camera driver ROS node.

The new cameras were installed in the second half of this project, while most of the current outcome was designed for a monocular camera. Moreover, the experiment on stereoscopic operation with current placement was not ideal. Therefore, this project focuses mainly on monocular camera application. Also, the stereoscopic application needs further improvement including better mechanical mounting in the future.

4.1 Interfacing

The use of a web camera requires no other driver but a Universal Serial Bus (USB) driver, which is always a standard component of a modern operating system. However, the use of FLIR Blackfly GigE cameras requires a specific library to handle the Gigabyte Ethernet protocol used by the cameras. The official driver provided by the manufacturer is FlyCapture2 Software Developer's Kit (SDK) [28]. The FlyCapture2 SDK was used to configure the cameras and capture the raw image data through an Ethernet interface. The image size is binned by a factor of 2 on both the width and height of the image to allow the use of both cameras at the same time, due to the limitation of Ethernet bandwidth.

The raw image data is then converted into ROS sensor message standard data structure for the image, and publish into similar ROS topics.

4.2 Calibration

The image captured from the camera presents the higher dimension realistic into 2-dimension representation, which may result in distortion caused by the lens of the camera. Furthermore, in order to extract the relationship between image frame and real-world reference global frame, the camera matrix, containing the information of focal length, camera intrinsic and extrinsic parameters, is required.

This process is done by OpenCV camera calibration module. By identifying the known features and their relationship in the image frame, the relationship as camera matrix can be approximated. The structure used in this project is a 9x7 black and white chessboard. Example of calibration is shown below.

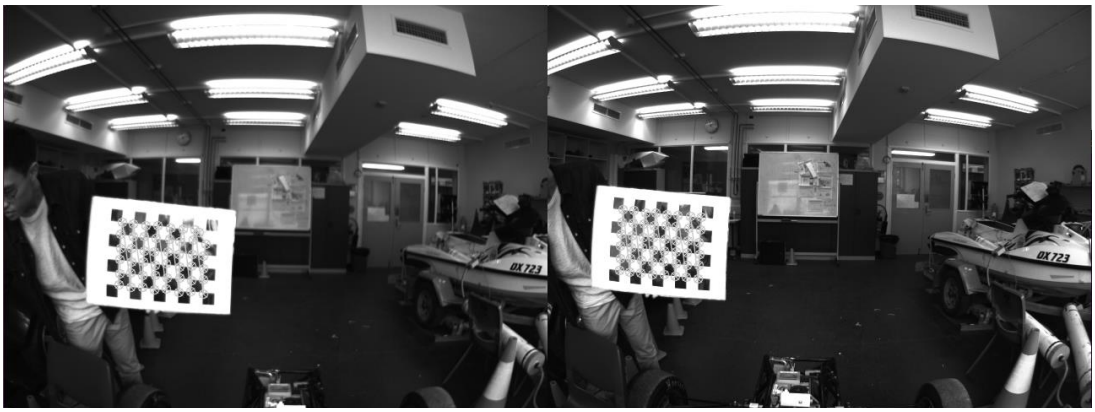

Figure 4Camera Calibration Examples for both Cameras

The results of calibration are shown below.

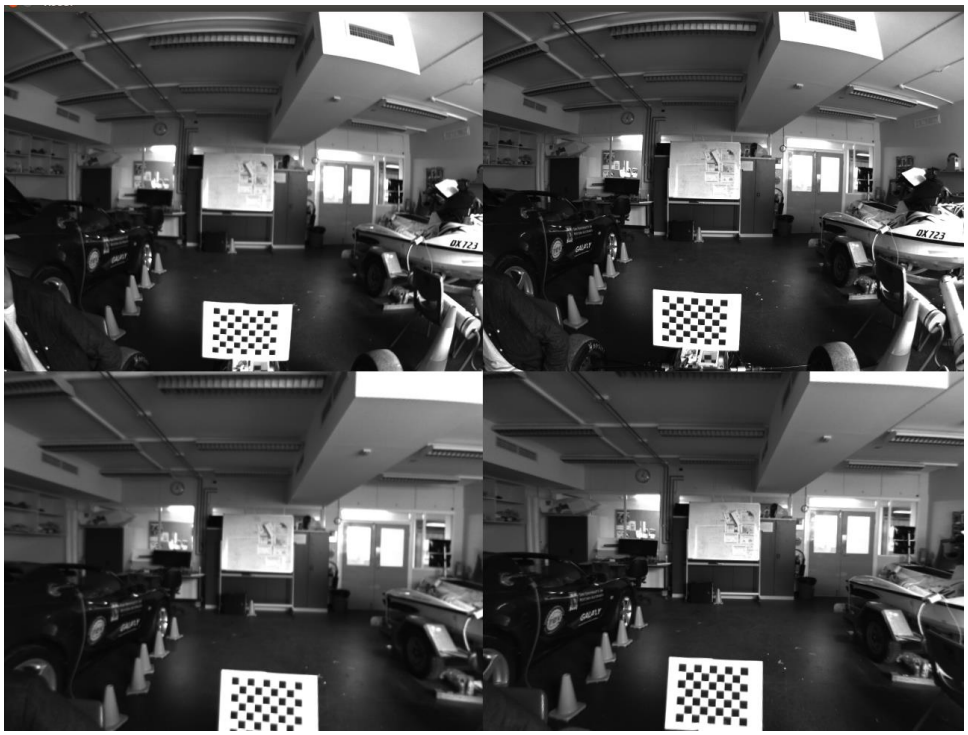

Figure 5Top: Raw Images for both Cameras; Bottom: Calibrated Images from both Cameras

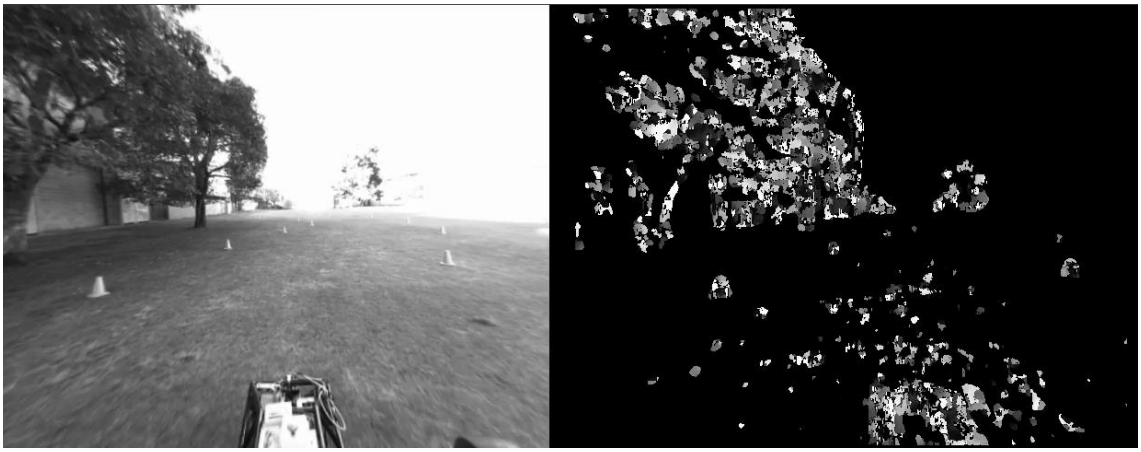

Figure 6Depth Map generated with stereoscopic cameras

As mentioned in the previous section, the stereoscopic setting of the current cameras is not suitable for autonomous driving application due to the low variance in the left and right image in the middle and long distance. The example of depth map generated using the current setting is also shown below.

4.3 Video Recording Collection

In order to lower the minimum requirement on development, the team collected sensory data in the format of rosbag in the weekly test drive. The pre-recorded rosbag allows the author to extract image and store as the video format for sample data source in the development process. With this data, the author was able to replay the data frame by frame to investigate the behaviour of each algorithm, which is constructive for the verifying algorithm or identifying the error.

At the time this paper is writing, there are over 5 hours recording in total for this project.

4.4 Image Sample Collection

Machine learning algorithms are used in cone detection module, requiring a considerable amount of training data with the correct label. Similar to video recording collection, the author replays the rosbag and captures sample data from the image frame by frame. The selected patches are cropped and stored as images in a folder with label name.

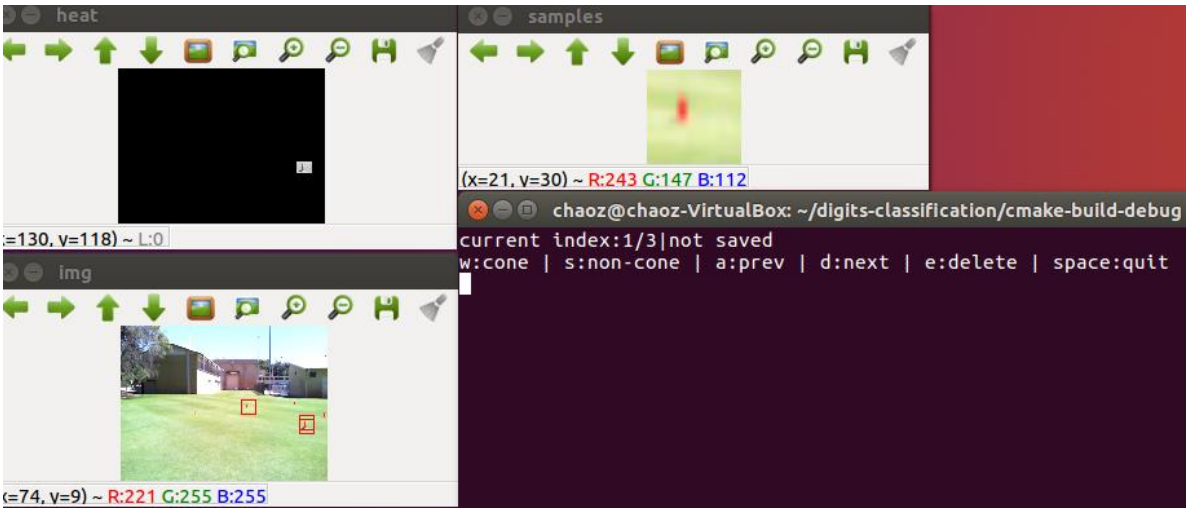

Figure 7Sampling Program Created in this project

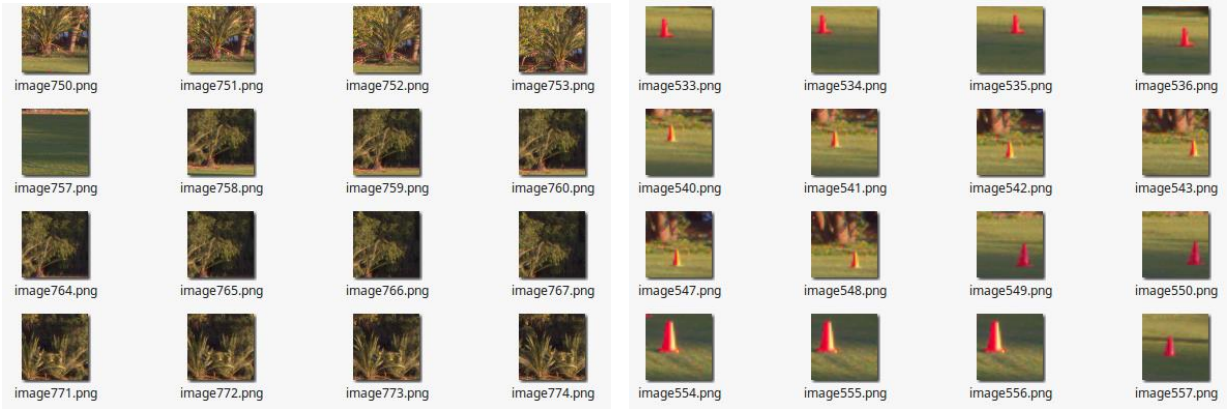

Figure 8Training Samples for Non-Cones (Left) and Cones (Right)

At the time this paper is writing, there are 1743 positive samples and 5945 negative samples in total for this project.

5 Localisation of SAE

Self-navigation is essential in order to achieve autonomous driving, which can be divided into three key areas — mapping, localisation and path planning. Simultaneous localisation and mapping (SLAM) encompass the former two areas, but there are many challenges in a practical application of visual SLAM. [4] These include highly dynamic environments, extreme variations in light intensity, sensor occlusions and motion blurs. However, due to the enormous advantage in camera cost as compared to long-range LiDARs and the rapid developments in GPU technologies, visual SLAM is gaining increasing interest due to its rich information output and increased affordability.

Visual SLAM is similar to visual odometry, and is differentiated by visual SLAM's ability to perform loop closures. Based on the KITTI Vision Benchmark Suite, ORB-SLAM2 [2] is the fastest visual SLAM algorithm with a runtime of only 0.03 s at the time of its publication. ORB-SLAM2 uses ORB features for all modules including mapping, tracking and place recognition, which results in an excellent efficiency and runtime performance. ORB [12] features use a fast binary descriptor based on Binary Robust Independent Elementary Features (BRIEF) [11] with an extra feature of introducing rotation invariance and noise resistance to some range. ORB-SLAM2 uses three parallel threads to perform three main tasks in visual SLAM — tracking, local mapping and loop closure. It is used as our baseline algorithm for visual odometry on the vehicle.

As previously mentioned, the TX1 comes with a powerful 256-core embedded GPU which can be used to improve image processing through parallelisation. The original ORB-SLAM2 was not adapted for this acceleration.

5.1 ORB SLAM2

ORB SLAM2 utilises fast corner detector in OpenCV for feature detection, and ORB feature extractor in OpenCV for feature extraction. The key frame is then created with camera pose, camera calibration parameters and extracted features. A keyframe-based graph optimisation procedure is then taken for updating the entire graph, including the camera pose for any related keyframe. In general, ORB SLAM2 have three parallel streams including Localisation, Place Recognition and Mapping. [29] An overview of the architecture used by ORB SLAM2 is shown below.

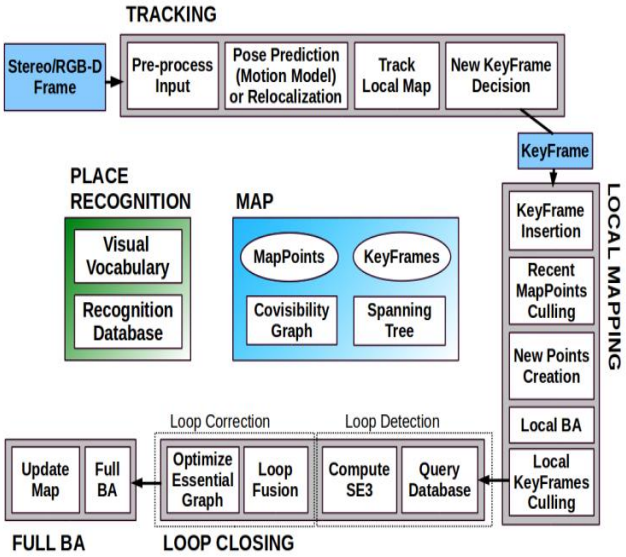

Figure 9System Overview of ORB SLAM2 [2]

The runtime performance of ORB SLAM2 is outstanding compared with other visual based SLAM algorithm based on the benchmark provided by KITTI. [5] Moreover, by disabling the mapping thread, the ORB SLAM2 can have much higher runtime performance with only about half of the processing time required for each frame. However, this will require the application environment must be mapped in advance.

6 Cone Detection

The research of object detection has been progressed dramatically in the last decade due to the development in CNN. The traditional computer vision approach using hand-crafted features has been overwhelmed by deep learning algorithms. However, in our case, for the simple binary classification task, the hand-crafted features remain its advantage in terms of simplicity, while the difference in accuracy is small. In this project, both approaches are implemented and compared. Eventually, the traditional approach is selected as the final solution. The discussion is given in the section below.

6.1 HOG with SVM

The traditional feature extraction used in this project is the Histogram of Oriented Gradient (HOG). The HOG descriptor first evaluates the gradient orientation and magnitude. Then by binning orientation into cells with the same size, sums up all the magnitude of gradients which is falling into that orientation range. After normalisation, a feature vector is obtained by flattening the two-dimension vector into one dimension.

The linear classifier used in this approach is Support Vector Machine (SVM). The SVM maps all sample feature vectors into a high dimension space and extracts an optimal linear classifier in that space which separates the support vectors with the largest margin distance. In order to train SVM, all samples are first converted into feature vectors using HOG descriptor. Then these feature vectors along with the correct label for each vector are then passed into SVM for training. The OpenCV provides an efficient implementation of SVM using LibSVM library. [30]

To identify all cones in an image, HOG descriptor is applied to the entire image. This is done by sliding a 64 x 64 window through the whole image with a stride of 4 and then passing the cropped image in each window into HOG descriptor. The obtained feature vectors are then fed into the optimised linear classifier evaluated from SVM training, which outputting binary classification results for all windows. Then, for all regions which are positively classified as a cone from the SVM, we then threshold the hue layer of the image with an orange value, as we benchmark our system using orange cones. We finally apply a histogram to the thresholded image and then obtain the position of the cone within the image frame. However, in order to fuse the classification result with other sensors, the detected cones must be presented in the global reference frame. This is done by applying a perspective transform to the image, and with the assumption that the vehicle is driving on a flat plane, the position of the cones in the global frame can then be obtained by projecting these cones into a horizontal ground plane. The detailed approach and procedure for extracting the location of cones are discussed in section 8.

6.2 CNN

In order to reduce the effect from variations of brightness caused by the different sunlight angles, more samples must be included in the training process of the SVM. This significantly degrades the runtime performance of the entire system while offering only a minor accuracy improvement. In which case, we designed a convolutional neural network

(CNN) to provide a flexible feature extraction method to adapt this variation in the environment. This network has two CNN layers and two fully-connected layers, which are used for detecting cones. In order to run the visual cone detection process smoothly along with all other modules in the system, we designed the network to be straightforward with a small footprint. Using this CNN yielded increased detection accuracies as compared to the SVM approach. However, without GPU acceleration, the performance using this network is not competitive compared with the SVM approach in terms of introduced latency.

7 Lane Detection

According to Jason Lin in 2017 [31], Canny edge detector was used, which uses a set of high non-linear algorithms for approximation, and therefore, requires long processing time. In this paper, Sobel filter is used instead. Sobel filter uses a linear algorithm for finding the edge, which is basically two kernels for approximating the gradient of a pixel in regards with its neighbour pixels.

The Sobel operator takes a single channel image as input. In order to further reduce the runtime performance, a grayscale image is used as input here.

The Sobel operator approximate the gradient in two directions, along with the axis x and axis y. However, in order to obtain actual edge information regardless of the direction of the edge, the magnitude of the gradient is used for the final mask.

The mask is obtained by thresholding the magnitude of gradient on each pixel.

In order to search for lane incrementally from bottom to top of the image, a perspective transform is performed to convert the image into a linear scale in terms of real-world distance. This operation assumes the ground is levelled. The lanes in the image are now parallel to each other after the transform.

To identify the starting point of each lane, a histogram of the mask on the y-axis is used. The peaks in left and right half of the image represent the start point for each lane. This operation is computationally expensive. In order to optimise the runtime performance, this operation is only used when the last frame of the image does not provide valid lane marking. Otherwise, the previous value is used.

Lane points are then located by sliding a rectangle window from bottom to top following the mass centre of the mask inside the window. The mass centre is identified as the point where the sum of all previous pixel in the window weight over the half of the total weight.

The lane points are then published as marker array message in ROS and is received by path planner module for further processing.

The complete steps for lane detection are shown below.

1. Undistort and Crop image
2. Apply Sobel filter and apply a threshold.
3. Do prospective transform into the processed binary image.
4. Get histogram for the image and find peak position on both right and left half of the image.
5. Use the peak position from the last step as the start position, slides windows from bottom to top and count the non-zero pixel inside each window. The mass centre is the line point we get, and the start position for next window.
6. Fit those line point with second-order function (optional).

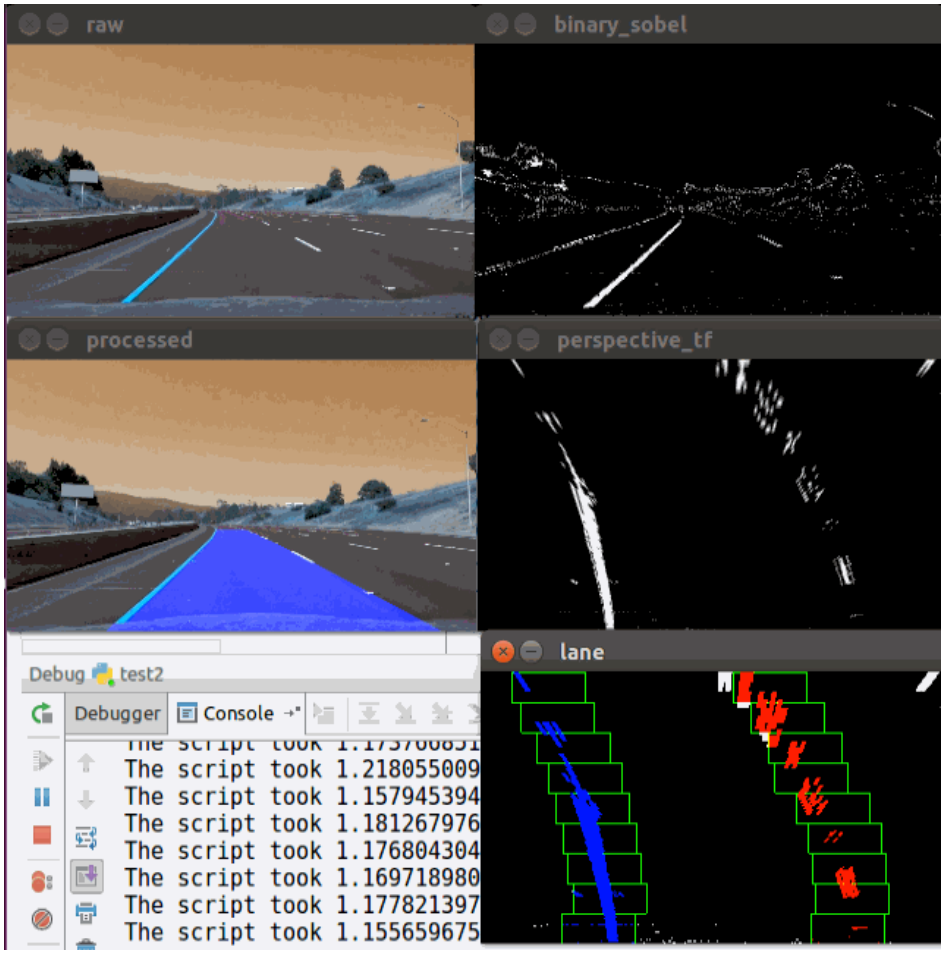

Figure 10Process breakdown of lane detection

8 Distance Extraction

The result from cone detection is only the location in the image. However, the desired outcome is the actual location in the global frame, with real-world distance in respect to SAE vehicle. In order to extract distance of cones, three additional steps are performed.

Firstly, for each cone detected, count the number of pixels with the desired colour range. In this case, the orange colour is used. If the number of pixels with correct colour does not reach a threshold, this classification result is then treated as false detection and does not proceed to the next step. For all cones passed the previous colour filter, histograms of the desired colour pixels on both x and y-axis are evaluated. Then the cone centre is identified as the point (peak position in the x-axis, peak position in the y-axis).

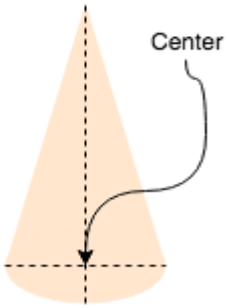

Figure 11Centre of the Cones

Secondly, the perspective transform is required to convert the image from an image frame into the horizontal reference frame where the cones sit. In order to do this, lists of image points with its global reference points are required. These points are then used for approximating the transform matrix.

This operation is handled by OpenCV function getPerspectiveTransform(). Then by applying this matrix to the region of interest in the image, a transformed image in bird view is obtained. This operation is handled by OpenCV function perspectiveTransform().

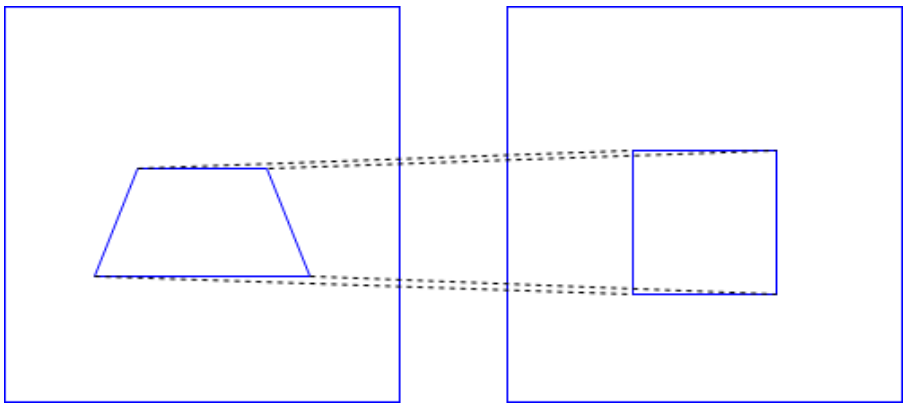

Figure 12Perspective Transform from Image Frame (left) to Real World Frame (right)

Lastly, by manually identify and record a reference point in the transformed image, with pre-measured distance scaling per pixel, the actual distance on both x-axis and y-axis in respect to the SAE vehicle can be identified.

9 Experiments

The experiments were done both in the field test and in the simulation. The field test is carried out weekly. The location of the field test is mainly in UWA at the oval next to sports science. The test bench used in the field test is the electric SAE vehicle. The location is mainly in UWA Crawley campus. During the field test, all low-level system and sensors are turned on, to simulate the fully operating condition. The program allows the driver to manually drive while collecting data and is controlled by path planner module while driving autonomously.

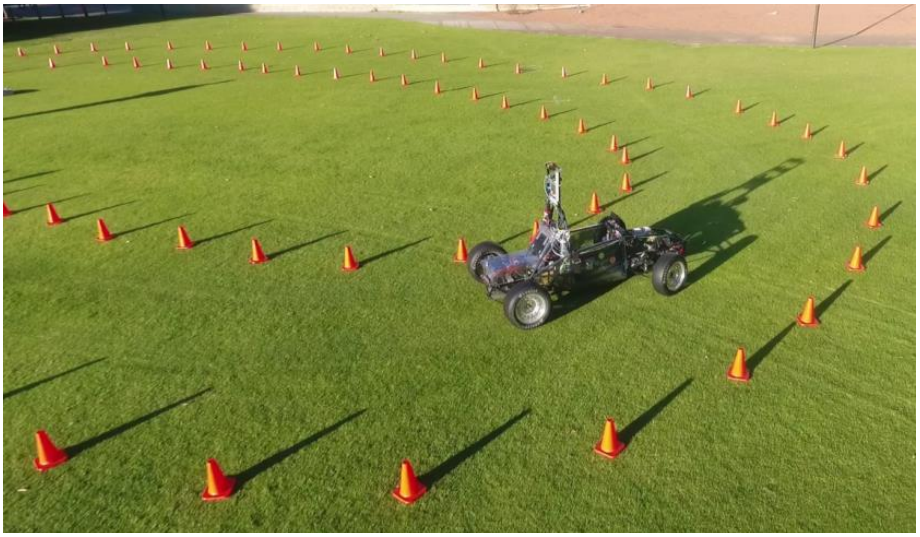

Figure 13Field Test Setup

Figure 14Simulation Setup

The simulation is carried out in UWA Electrical & Electronic Engineering building lab 3.13, in a hardware in the loop simulation system setup by Craig Brogle. [19] The simulation used in this project is a modification from open-source simulation system CARLA. [19] Unreal Engine is used by CARLA as core physic graphics simulating engine. The simulation is running on an external independent computer, while the tested software is running on identical platform Nvidia Jetson TX1 as on the SAE vehicle. At current stage, the calibration for camera in simulation is still infeasible, therefore, only task which does not require calibration is tested in simulation.

9.1.1 ORB SLAM2

The map created in ORB SLAM2 is compared with Google Map shown below.

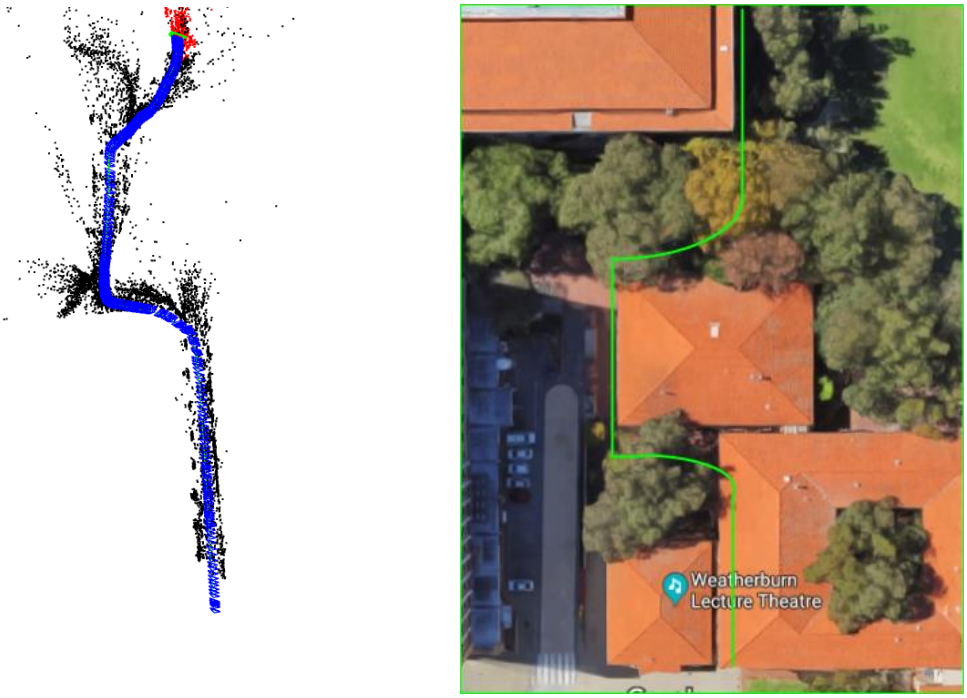

Figure 16Map. (left: ORB SLAM2; right: GOOGLE MAP)

Figure 15 Feature Extraction in ORB SLAM2. (green marker is the ORB feature matched with current key frame)

The map generated from the ORB SLAM2 shows a satisfying result with most of the well-constructed structure presented correctly compared with the inaccurate wheel odometry. Furthermore, the ability to localise its own position within the mapped area is excellent. And the time required for localisation only is much less than full SLAM mode, which will be discussed in section 9.2 Runtime Performance.

However, due to the fact that there are no other reliable localisation modules available on the SAE vehicle at the time the paper is written, the translational and rotational errors introduced by ORB SLAM2 cannot be measured accurately. Further calibration about this algorithm requires localisation results from either other SLAM algorithm or sensor fusion with IMU.

9.1.2 Cone Detection

The cone used in this project has dimension as 228 x 228 x 325 (l x w x h) mm and is in orange colour. The test drive is generally in the afternoon of a day, with a lower sunshine angle. As a result, the test drive is affected by the different sunshine angle toward the cones while driving in a loop, which can have cones been washed out by the sunlight.

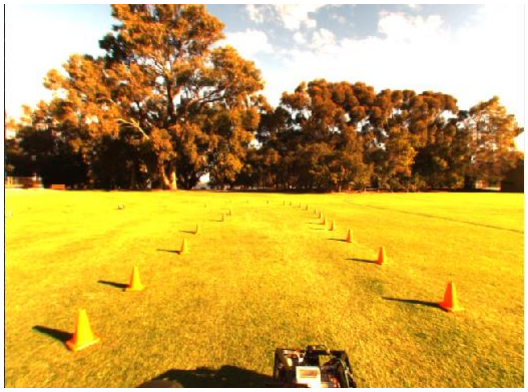

Figure 17 Illumination Variation on Field Test

The accuracy of cone detection is measured in two approaches. First is by running the detection overall image data samples including the training set and validation set. Second is by running the detection over video recording, and randomly select frames for accuracy evaluation over the entire image.

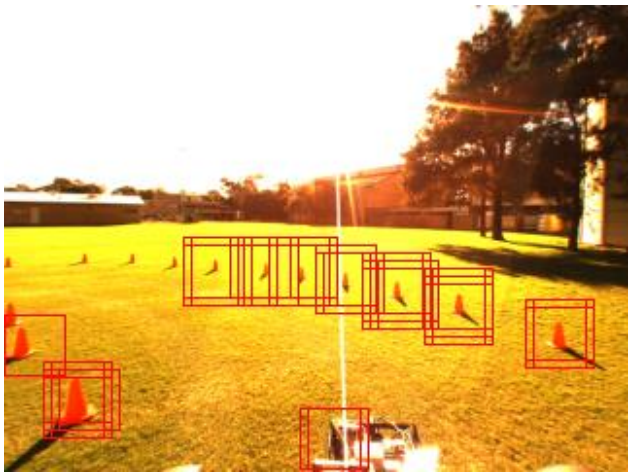

Figure 19Cone detection on field test

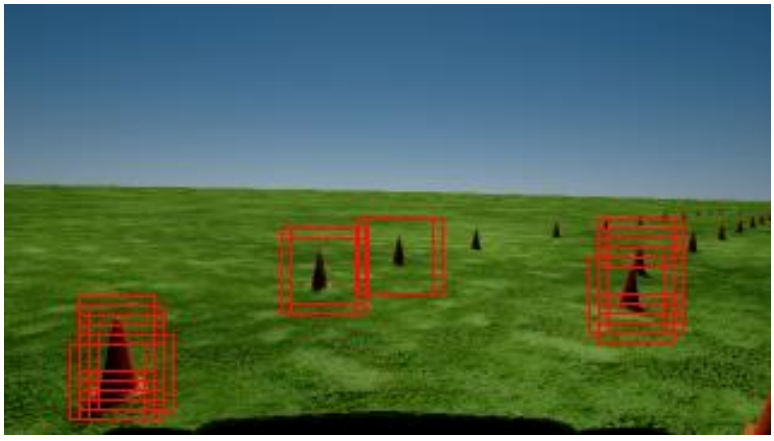

Figure 18Cone detection on Simulation

The result is shown below.

Table 1Accuracy

	Training Set	Test Set
Accuracy	96.3%	92.1%

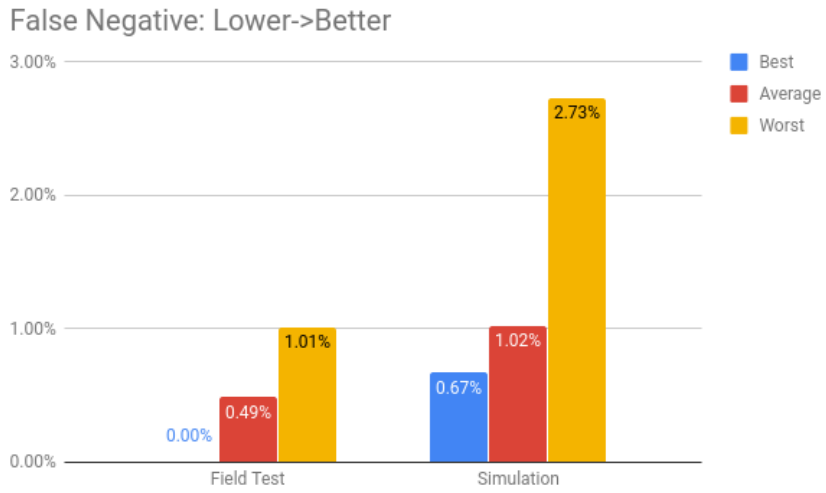

Figure 21 False Negative for detection results

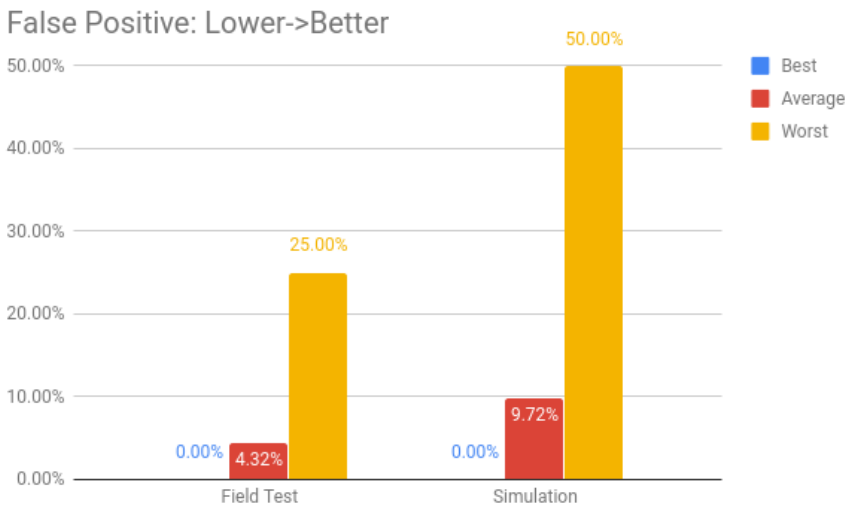

Figure 20False Positive for detection results.

The experiment presents satisfying results in terms of accuracy for both field test and simulation. In general, the simulation environment yields a slightly worse result compared with the result from the field test. This is caused by the fact that the training data is collected from recording using identical camera used in the field test, while the image from the simulation is completely “new” for our classifier. This performance can be improved by training classifier with data from the simulation. Also, due to the difference in camera mounting angle and position, the cones are presented with various size. With a fixed

window size, the effective range of detection is reduced in simulation for a slightly larger cone.

9.1.3 Lane Detection

The experiment on lane detection was carried out at RAC race track. There are various lane markings with different width.

Figure 23 Lane detection on RAC race track. (top: lane detected in region of interest; bottom: raw image captured on field)

The lanes are detected and converted into a real-world position regarding the SAE vehicle.

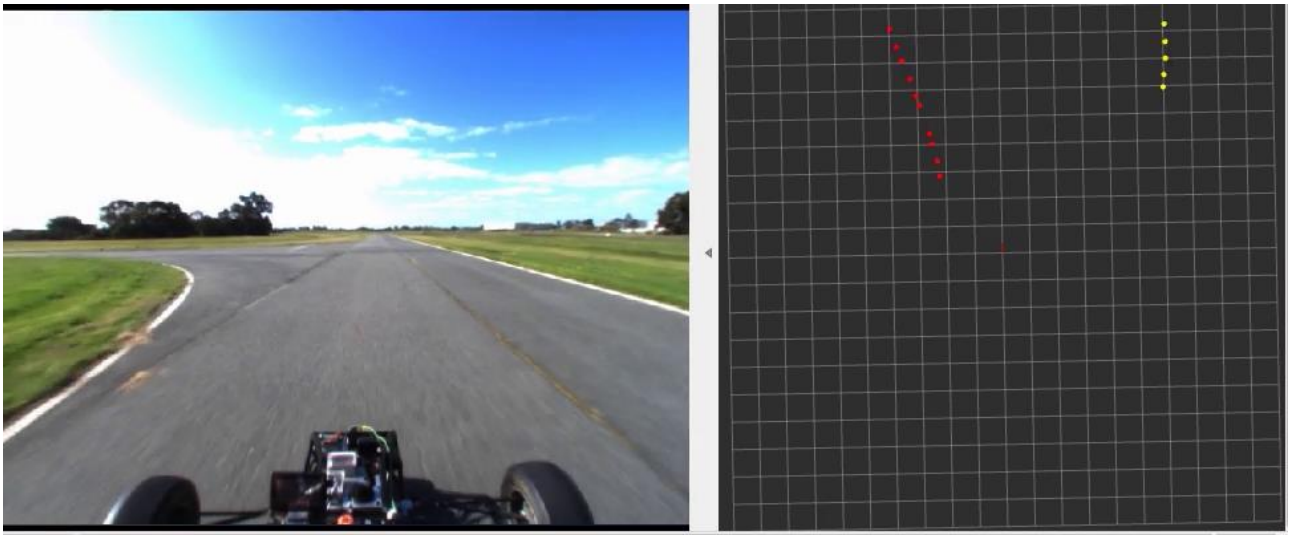

Figure 22 left: raw image; right: lane points passed to path planner

The reason to publish the lane as points is for the compatibility with path planner. The current existing path planner module accepts object points for its path generation algorithm. In order to integrate the lane detection module with the path planner without massive rewrite in other modules, the author decides to publish the lane in the same format as cones with a small interval along the lane.

9.2 Runtime Performance

In order to enable autonomous driving, the vehicle must be able to react to the environment fast enough to avoid potential crashes. The target performance for this project is set to minimum as 10 FPS.

The runtime performance is collected without GPU acceleration.

The runtime performance is also collected from all modules running in TX1 without any other non-critical system process running.

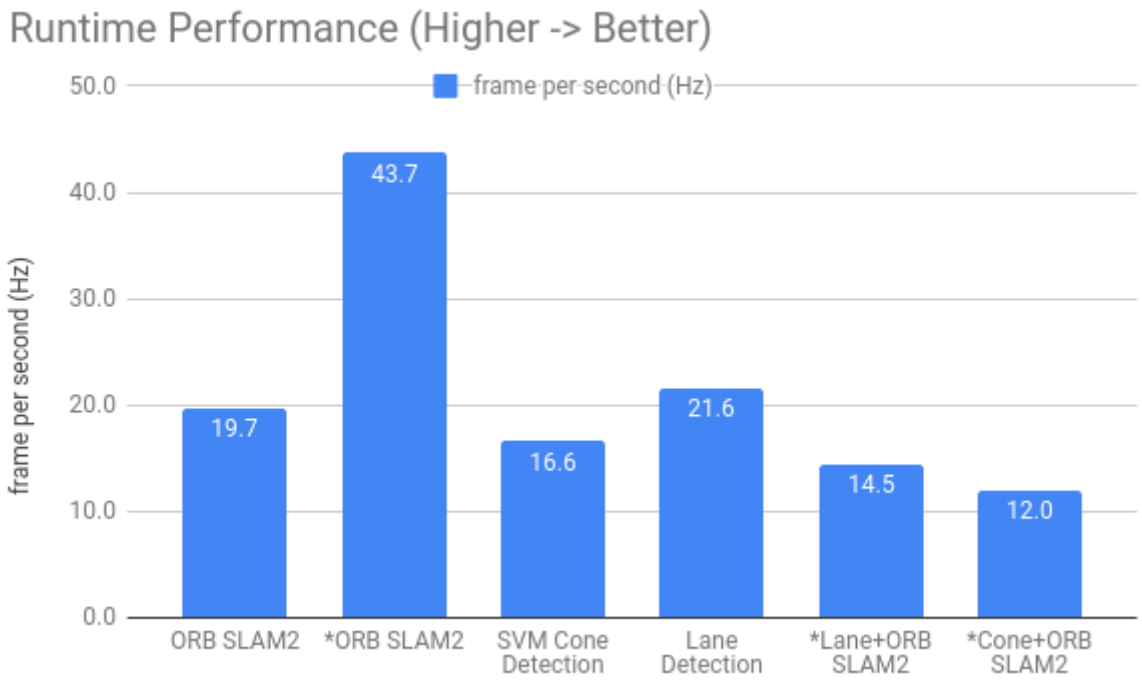

Figure 24 Runtime Performance on SAE vehicle.

The runtime performance for ORB SLAM2 is excellent, especially for localisation mode, which is presented as "*ORB SLAM2" in Figure 24. The cone detection and lane detection modules also satisfied the requirement for a real-time application (minimum 10 FPS). Moreover, the full functional combinations of lane detection and cone detection with localisation are able to run at 14.5 FPS and 12.0 FPS respectively.

10 Conclusion

The author has presented an integrated software framework for a visual system that is designed for REV SAE autonomous driving vehicle. Compared with the visual modules in 2017, localisation is added, the detection algorithm is optimised to allow real-time operating, and the visual modules are tested and integrated with other modules such as path planner. Camera setup is updated with new equipment and new data sets, which allows machine learning based detection algorithm to reach 94% accuracy in the field test.

Most importantly, the runtime performance for all modules is improved with optimisation in both implementation and algorithm. The runtime performance for lane detection is improved to 21.6 fps from about 1 FPS. The full lane detection with localisation is now able to run at 14.5 FPS. The full cone detection with localisation is now able to run at 12 FPS.

For cone detection, since a cone is a well-structured object and there are only two classes in the classification, the traditional approach handles this task very well, given a competitive accuracy performance and even better runtime performance compared with CNN approach. However, there is a potential improvement in runtime performance with GPU acceleration given there are powerful GPU cores available in TX1.

Overall, the visual modules introduced in this paper are robust and low in latency, providing detection and localisation results in real-time to help the decision-making process in path planning and backup the detection when other sensors are failed to provide reliable results.

10.1 Future Work

To obtain better depth information from the camera, a better configuration on stereoscopic cameras is beneficial. This can be achieved by replacing the current lens with a more suitable one or adjust the width between two cameras. The first option requires extra cost related to the hardware replacement, and the second option requires additional mechanical modification to the sensor rack on the SAE vehicle. Due to the limitation on time, this improvement is recommended as future work.

To further improve the performance of the current cone detection algorithm, ORB feature can be introduced to filter the patches of the image fed into the classifier, which can effectively reduce the amount of processing required for cone detection. The current number of patches processed in each frame is 804, while the number of features identified in ORB SLAM2 is only around 100 or even less when driving on the oval or race track. Also, as shown in the experiment result in section 9, there are some overlapping in the cone detection. The overlapped windows can be removed to reduce processing time further less as these operations are redundant.

The Nvidia Jetson Series equips with dedicated GPU, which allows hardware acceleration to tasks that can be parallelised, such as applying local filter toward the entire image, linear classifier evaluation and graph optimisation in ORB-SLAM2. Therefore, optimise the current visual system with CUDA C can further lower the latency.

Also, the enable of GPU acceleration may make DNN detection approach feasible, such as MobileNetV2 SSD [32]. However, the processing time required for running critical modules for the SAE vehicle must be taken into consideration when applying computationally intensive algorithms.

As the depth information from the camera always more fluctuates than the depth from LiDAR, it is even more beneficial to apply an adaptive filter for each detection. Extended Kalman Filter (EKF) can be a good candidate for this task. It will not just stabilise the detection result, but also helps to filter out some false detection over time, which can result in better accuracy performance.

ORB SLAM2 is a popular visual SLAM toolset in the open-source community. There is also some potential for improvements. Firstly, the saving and loading function for the generated map has not been implemented. Secondly, the Dictionary Bag of Words (DBoW2) [33] dictionary used in ORB SLAM2 can be optimised for our use case, by regenerate the dictionary using our own data samples. Lastly, despite the fact that the ORB extraction is fast, it is still a hand-crafted feature. Apply deep learning algorithm to train an optimised feature extractor may be beneficial for the overall performance.

In higher level, powered by the boost from GPU performance, end to end network becomes applicable in many use cases. End to end network takes raw image input and outputs the control command directly without any unnecessary middle steps, which results in high efficiency in general.

11 Reference

12

[1] NVIDIA, "Jetson TX1 Module," 21 10 2018. [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx1.

[2] 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, 6 2017.

[3] C. Stachniss, "Robotic Mapping and Exploration," *Springer Tracts in Advanced Robotics,* vol. 55, p. 198, 2009.

[4] J. Fuentes-Pacheco, J. Ruiz-Ascencio and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review,* vol. 43, no. 1, pp. 55-81, 1 2015.

[5] 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*, 2012.

[6] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics,* vol. 31, no. 5, pp. 1147-1163, 10 2015.

[7] A. Sharma, P. K. Singh and P. Khurana, "Analytical review on object segmentation and recognition," in *Cloud System and Big Data Engineering (Confluence) 6th International Conference*, 2016.

[8] W. T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition," in *IEEE Intl. Wkshp. on Automatic Face and Gesture Recognition*, Zurich, 1995.

[9] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proceedings of the International Conference on Computer Vision*, 1999.

[10] H. Bay, T. Tuytelaars and L. V. Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision*, 2006.

[11] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision*, 2010.

[12] E. Rublee, V. Rabaud and K. Konolige, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference*, Barcelona, 2011.

[13] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Network," 2012.

[14] J. Redmon, S. Divvala and R. Girshick, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[15] R. Girshick, J. Donahue and T. Darrell, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013.

[16] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," 2015.

[17] A. Paszke, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation," 2016.

[18] "ROS.org | Powering the world's robots," [Online]. Available: http://www.ros.org. [Accessed 21 10 2018].

[19] K. L. Lim, T. Drage, C. Zhang, C. Brogle, W. W. L. Lai, T. Kelliher, M. Adina-Zada and T. Braunl, "Evolution of a Reliable and Extensible High-Level Control System for an Autonomous Car," 2018.

[20] Arduino, "Arduino Mega 2560 Rev3," Arduino, [Online]. Available:

https://store.arduino.cc/usa/arduino-mega-2560-rev3. [Accessed 21 10 2018].

[21] O. S. Initiative, "The 3-Clause BSD License," Open Source Initiative, [Online]. Available: https://opensource.org/licenses/BSD-3-Clause. [Accessed 21 10 2018].

[22] O. S. R. Foundation, "ROS/Introduction - ROS Wiki," [Online]. Available: http://wiki.ros.org/ROS/Introduction.

[23] K. L. Lim, T. Drage, R. Podolski, G. Meyer-Lee, S. Evans-Thompson, J. Y.-T. Lin, G. Channon, M. Poole and T. Braunl, "A Modular Software Framework for Autonomous Vehicles," in *29th IEEE Intelligent Vehicles Symposium*, Changshu, 2018.

[24] "OpenCV library," OpenCV team, [Online]. Available: https://opencv.org/. [Accessed 21 10 2018].

[25] FLIR, "Blackfly 1.3 MP Color GigE PoE (Sony ICX445)," [Online]. Available: https://www.ptgrey.com/blackfly-13-mp-color-gige-vision-poe-sony-icx445-camera.

[26] "Fujinon YV2.82.8sa-2, 2.8mm-8mm, 1/3”, CS mount Lens," [Online]. Available: https://www.ptgrey.com/fujinon-yv28x28sa-2-hd-vari-focal-lens-3.

[27] C. Liang, L. Chang and H. H. Chen, "Analysis and Compensation of Rolling Shutter Effect," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1323-1330, 2008.

[28] FLIR, "FlyCapture SDK," FLIR, [Online]. Available: https://www.ptgrey.com/flycapture-sdk. [Accessed 21 10 2018].

[29] C. Kahlefendt, "Implementation and Evaluation of Monocular SLAM for an Underwater Robot," 2017.

[30] C.-C. Chang and C.-J. Lin, "LIBSVM : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011.

[31] J. Y.-T. Lin, "Autonomous SAE Car – Visual Base Road Detection," *Final Year Project Thesis*, 2017.

[32] M. Sandler and A. Howard, "MobileNetV2: The Next Generation of On-Device Computer Vision Networks," Google, 3 4 2018. [Online]. Available: https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html. [Accessed 21 10 2018].

[33] D. Galvez-Lopez and J. D. Tardos, "Bags of Binary Words for Fast Place Recognition in Image Sequences," *IEEE TRANSACTIONS ON ROBOTICS*, vol. 28, no. 5, pp. 1188-1197, 2012.