The University of Western Australia

School of Electrical, Electronic and Computer Engineering

Final Year Research Project Proposal

Autonomous SAE Car

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

Chao Zhang

21435393

Supervised by Professor Dr. Thomas Bräunl

Submitted on Friday 20th April 2018

Table of Content

List of Figures and Tables

Nomenclature

UWA	The University of Western Australia
REV	The Renewable Energy Vehicle Project
TX1	NVIDIA Jetson TX1
CPU	Central Processing Unit
GPU	Graphics Processing Unit
FPS	Frame Per Second
SLAM	Simultaneous Localisation And Mapping
DNN	Deep Neural Network
CNN	Convolutional Neural Network
ORB	Oriented FAST and Rotated BRIEF
BRIEF	Binary Robust Independent Elementary Features
HOG	Histogram of Oriented Gradients
YOLO	You Only Look Once
R-CNN	Region-based Convolutional Neural Network
DBoW2	Enhanced Hierachical Bag-of-Word Library
OpenCV	Open Source Computer Vision Library
LIBSVM	Support Vector Machines Library
SVM	Support Vector Machines
NumPy	A Fundamental Package for Scientific Computing with Python

1. Introduction

1.1. Introduction and Project Goals.

The Autonomous SAE car project aims at develop an ultimate autonomous driving system for REV project's 2010 SAE Electric car. 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 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 Dan, Junho Jung, William Lai, Patrick Liddle and Chao Zhang, will focus on different topics and complete this project together. 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 to the SAE sensor fuser
- Provide cone location to the SAE mapper
- Provide lane line to the SAE path planer
- Real-time application with above functions

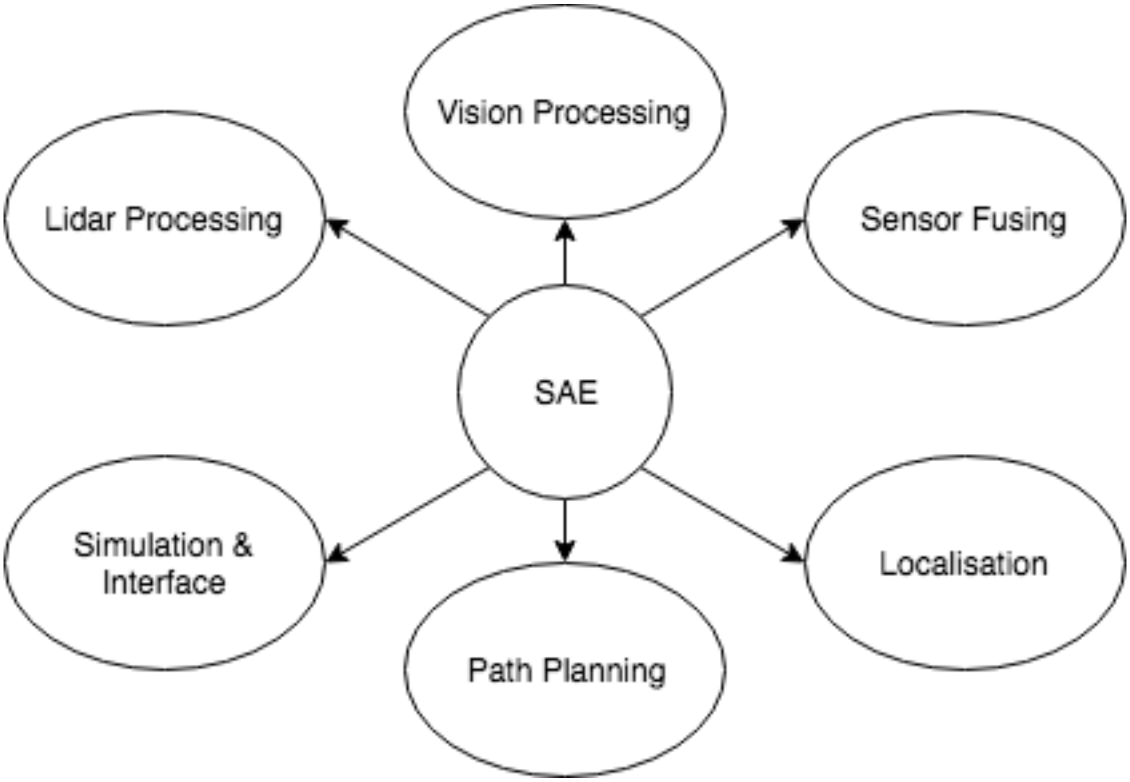


Figure 1 High Level Architecture of the SAE Autonomous Driving System

1.2. Problem identification

In order to achieve the goals of the project this year, there are three main 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 FPS on TX1.

1.2.1. Localisation of the SAE

Simultaneous Localization 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 due to the fact that SLAM is a critical component of an autonomous driving system. [1]Visual SLAM gains increasing interests as it advantage on cost compared with other sensor 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 IMU. In this project, the author will focus on extracting the estimated location of the SAE from a single monocular webcam using visual SLAM algorithm.

1.2.2. Detection of cones

Another key component in autonomous driving system is object detection, and in this case, is the cone detection specifically. In order to generate a drivable path following two sets of cones, the location of each cones have to be identified. To do this, the first thing is to successfully recognise the cones from camera. Then the actual position of the cones in horizontal plane has to be worked out.

1.2.3. Define Drivable Area

The vision has been proven as an excellent tool for detecting drivable area. For a fully autonomous driving system, the vehicle need to identify the drivable area in order to plan the future path safely. In 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.

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. [2] Initially, the research on mapping and localisation were independent. However, it was soon found that they are actually dependent with each other. A precise localisation requires an accurate map and to build an accurate map the localisation for every objects in the scenes is critical. The name of SLAM was first used in 1985. [3] 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. And when camera is used as the only sensor, the name visual SLAM is used. There are many challenges in actually application of visual SLAM, including highly dynamic environments, too high or too low intensity, occlusion of the sensor and blur caused by movement. [3] However, with the huge advantage in camera cost compared with long range lidar and the fast development in 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 visual SLAM algorithm with a runtime of only 0.03s [4]. ORB SLAM2 uses ORB feature for all modules including mapping, tracking and place recognition, which results in an excellent efficiency and runtime performance. [5]ORB feature is a fast binary descriptor based on Binary Robust Independent Elementary Features with extra feature as rotation invariant and noise resistance to some range. [3]ORB SLAM2 has three parallel thread to perform three main tasks in visual SLAM: tracking, local mapping and loop closure. [5]Inspired by the idea of using same features for all modules, in this thesis, the ORB SLAM2 will be used as the baseline system and ORB feature will be used for most of the other vision modules.

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[4]. 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. [6] The traditional method in object detection is using hand-crafted features, such as HOG [7], SIFT [8], SURF [9], BRIEF [10] and ORB [11]. These hand-crafted features provides a better runtime performance when only runs on CPU. However, as the development on GPU, parallelisation highly reduce the runtime for an adaptive filters, which is called convolutional layers. By feeding the feature vectors from convolutional layers into fully connected neural network, a deep learning structure called CNN is created. The CNN has achieved a great success in object recognition and detection. The state of art model including AlexNet [12], YOLO [13], R-CNN [14]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 [15]. And with better optimisation and enhancement for mobile application, ENet [16] achieved similar accuracy and far better runtime performance.

3. Design Architecture

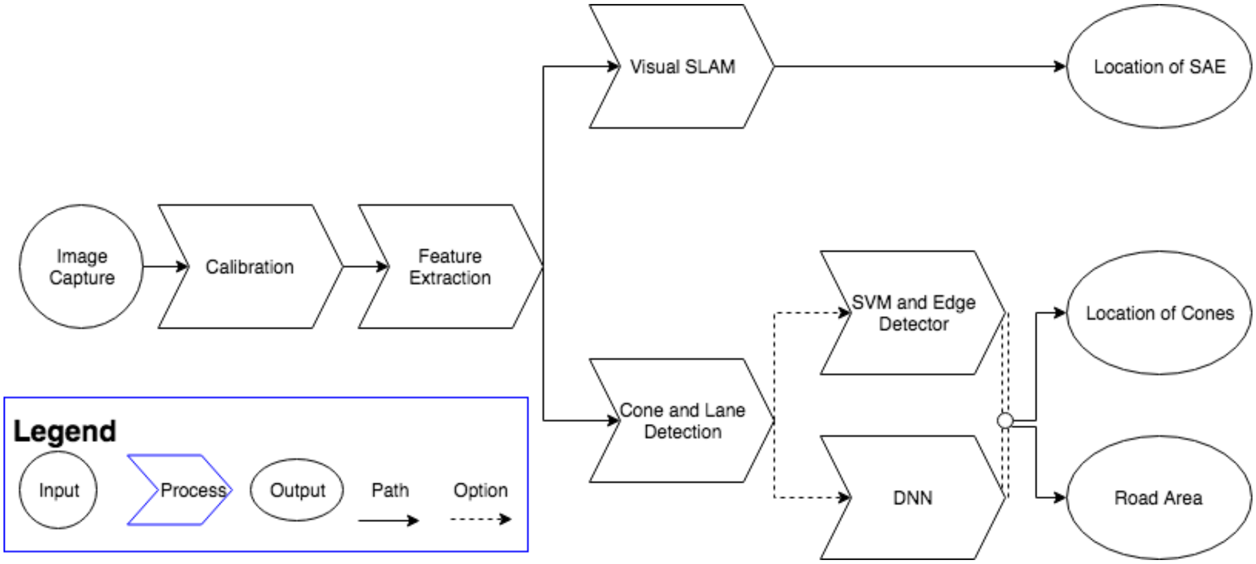


Figure 2 High Level Architecture of the integrated vision processing system

3.1. Feature Extraction

The main idea in this stage is that the same features used by the visual SLAM are used for object detection and road detection, which will increase the efficiency of the entire system and reserve as much computational capacities as possible for other modules. In this case, the feature chosen by this project is ORB features.

The ORB feature will be extracted from the entire image frame. For visual SLAM algorithm, the features from the entire image frame will be used. However, for object detection, only the patch in region of interest will be used. And for segmentation, only bottom half of the image are interested, as the road section in an image are mainly falls on the bottom half.

However, the ORB feature is only obtained at key points of an image, and as the number of key points varies from image to image, it would be very complicated and difficult to train a machine learning algorithm only based on the raw ORB features. In order to make the training more efficient and simple, DBoW2 is used here to generate bags of word from samples. By matching the ORB features from image to each bag, a histogram was created as our final feature vector which will be fed into other algorithms.

3.2. Visual SLAM

The baseline algorithm used as visual SLAM in this project is ORB SLAM2[reference]. However, in order to increase the efficiency of this algorithm for our specific cases, a new vocabularies need to be trained using the image taken by the device used in this project. By doing this, the size of image can be reduced.

As mentioned above, the jetson TX1 comes with a powerful embedded 256 cores GPU which can be used to boost image processing by parallelization. The origin ORB SLAM2 hasn't adapted GPU acceleration. And in this project, the ORB SLAM2 will be adapted to CUDA [17]and utilise the GPU on TX1.

3.3. Cone and Lane Detection

These two objectives are combined into one section here because of the potential to merge and achieve these two objectives in one algorithm. In this project, two different paths will be taken to solve these problem and the results will be compared. The most efficient algorithm will be chosen for the final solution.

3.3.1. SVM and Sobel Edge Filter

The first option is to apply SVM for cone detection and use Sobel edge filter for lane detection.

For cone detection, the histogram vector from the bags of word for each patch will be used as the input for SVM. Then for all patches which are classified as cone from SVM, threshold the hue layer of the image with orange value ,apply histogram to the thresholded image and then obtain the center of the cone as below.

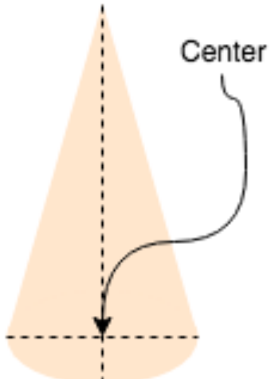


Figure 3 Center of the cone

For Lane detection, the Sobel filters are applied to extract edge information in region of interest. This region is initially defined as the bottom half of the image. After a successful detection on edge, this region is reduced to a smaller region around the lanes, with the assumption that the lane should be continuous in shape. Then by applying perspective transform to the image onto horizontal plane, the pixels with value higher than threshold value are considered as lane points. The lane points can be used as raw input for mapper module to generate drivable path, or can be used to obtain the exact curve line of the lane by applying curve fitting.

3.3.2. DNN

The second option is to adapt a pre trained DNN model for semantic image segmentation, and retrain it for our cases. There is an implementation on SegNet by last year student Yao-Tsu Lin. [18]However, due to the high latency of this network, this algorithm has not been considered to be run on SAE. Therefore, a real-time semantic segmentation algorithm ENet will be used instead. This DNN model is designed for mobile application on real-time application. On TX1, the frame per second using ENet is in average 14.6 while the frame per second using SegNet is only 0.8. [16]In this project, both the cone detection and road detection can be achieved by ENet at the same time. This requires an retrain on the pre-trained model of ENet to include an additional class as cone. The output of ENet is an image with n layers where n is the number of classes. By applying smoothing on the edge of road and cone segments, the lanes and cones can be detected and located using the same method in the first option.

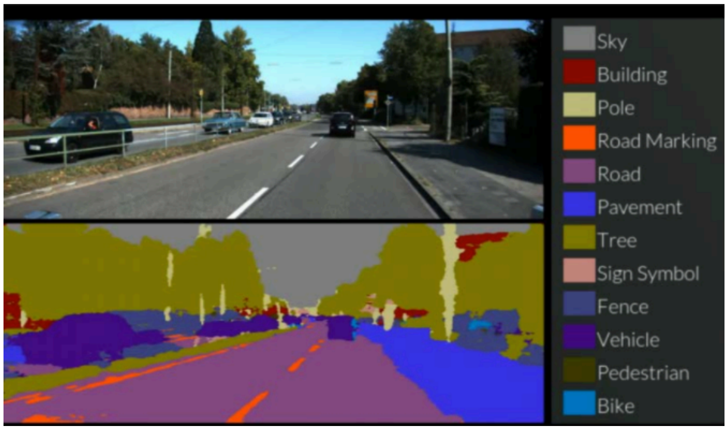


Figure 4 Implementation of SegNet by Lin [18]

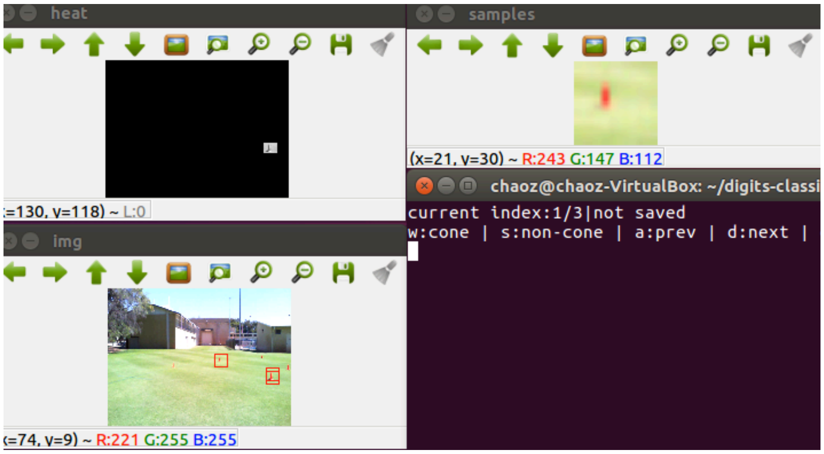
4. Preliminary Work Completed

4.1. Camera Calibration

The camera calibration has been completed by using a printed chess board and OpenCV [19] library. 45 images of the chess board are taken from different perspective and location under uniform lighting condition. By identifying the known structure of the chess board and matching the corner of the chess board, the distortion coefficient matrix and camera matrix can be obtained.

4.2. Training Data Collection

The training data used in this thesis is the driving video in the two scenarios specified by the project goals, which are driving with rows of cones and driving in the internal road. There are currently over one hour length of driving video has been taken by the team since the beginning of this thesis.


heat
samples
img
chaoz@chaoz-VirtualBox: ~/digits-classi
(x=21, y=30) ~ R:243 G:147 B:112
:=130, y=118) ~ L:0
(=74, y=9) ~ R:221 G:255 B:255

```
current index:1/3|not saved
w:cone | s:non-cone | a:prev | d:next |
```

Figure 5 Sampling and labeling software

For training a machine learning model for object detection, the image must be sampled and labeled. In order to complete this task efficiently, the author designed a software to extracting and labeling sample images from video using C++ with OpenCV [19] library. At the time this proposal is written, there are currently more than 2000 new images labeled for cone detection.

4.3. Lane Detector

The lane detector has been created and tested.

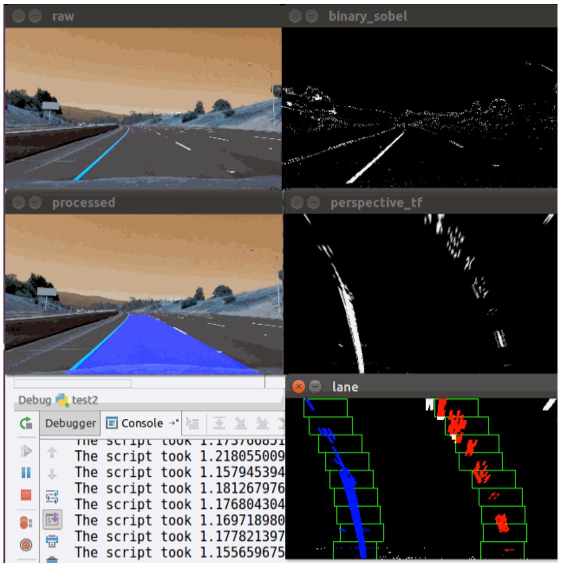
The detection following below steps:

- Undistort image
- Apply sobel filter and apply a threshold
- Do prospective transform to the processed binary image
- Get histogram for the image and find peak position on both right and left half of the image
- Use the peak position from last step as start position, slides windows from bottom to top and count the non zero pixel inside each window. The mass center is the line point we get, and also the start position for next window.
- Fit those line point with second order polyn function

The software is written in Python with OpenCV [19] and NumPy [20] package.
The average processing time for a 1280x720 image is 1.5s. The software is running under 2.9 GHz Intel Core i7 with two cores.

Current issues:

- When the gap between two dash line is too large, the fitting is not good enough.
    - Solution: add memory to previous line or record the average width of path and do estimation based on the width.
- If there are too much dark shadow area, the fitting is not good.
    - Solution: Do preprocessing before sobel operation to normalise the intensity of the image.
- Processing time is too long.
    - Solution: Optimisation on code is required. Rewrite the software using C++ with GPU acceleration enabled.



```
The script took 1.173700851
The script took 1.218055009
The script took 1.157945394
The script took 1.181267976
The script took 1.176804304
The script took 1.169718980
The script took 1.177821397
The script took 1.155659675
```

Figure 6 Lane detection

As this software is created at a very early stage of the thesis, further integration with other module is required.

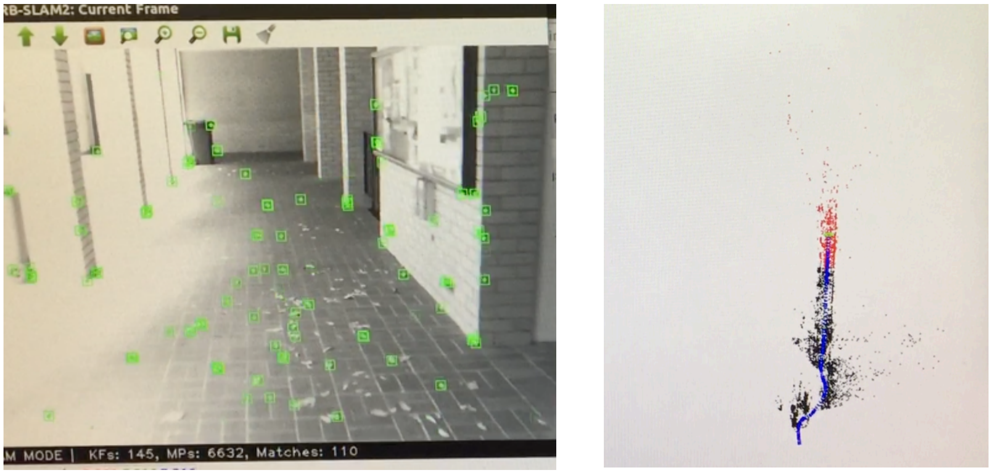
4.4. ORB SLAM2 Implementation

The implementation of ORB SLAM2 on TX1 with GPU acceleration is completed. The average FPS is 14.

Current issues:

- No official support on save/load functions
  - Solution: As ORB SLAM2 is an open source project, the author can modify and add save/load functionality into the software.
- There is a high chance for the software to lose the tracking when turning.
  - Solution: Use other localisation when turning.
- High memory usage due to the large vocabulary used for feature matching.
  - Solution: Retrain the bags of word using the collected testing data.

Also, with the save and load functions, the map can be generated beforehand, thus only localisation module will be used during the autonomous drive, which will reduce the computation caused by this software and increase the performance.



```
M MODE |  KFs: 145, MPs: 6632, Matches: 110
```

Figure 7 Implement of ORB SLAM2

4.5. HOG feature SVM Cone Detector

The implementation of cone detector using SVM is done. The training dataset used is created by previous year student Yao-Tsu Lin in last year. [18] There are three different SVM used in this software corresponding to different size of windows. Below is the indicator about the search range for different size of windows. The runtime performance of this software on TX1 without GPU acceleration is 8 FPS.

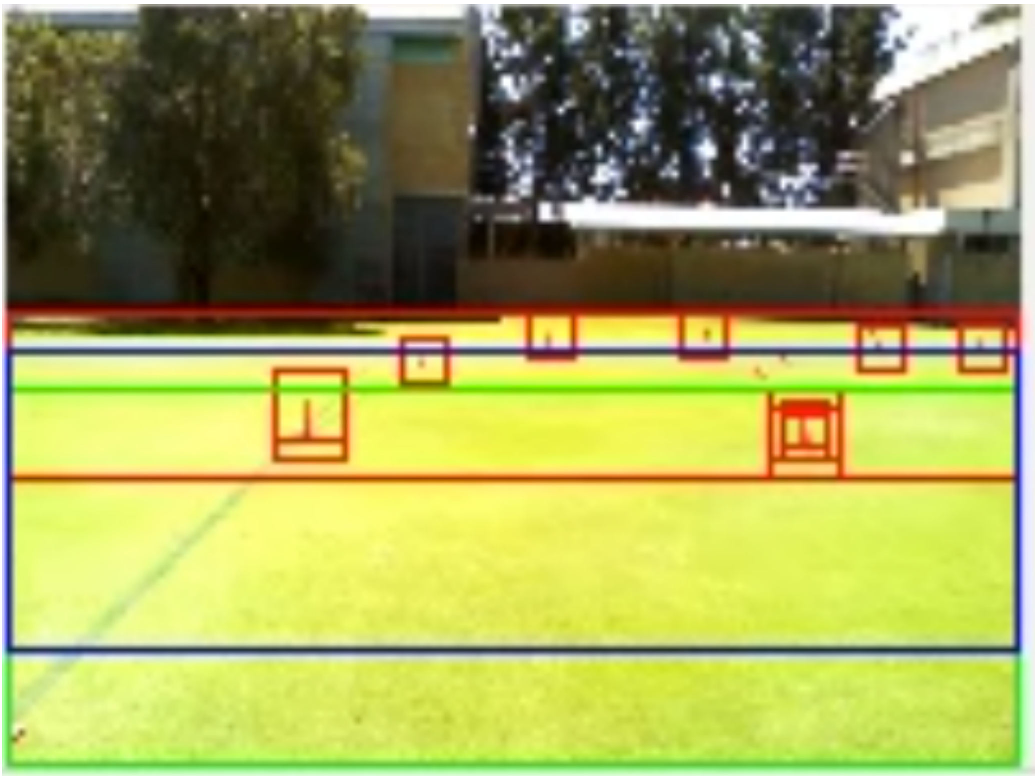


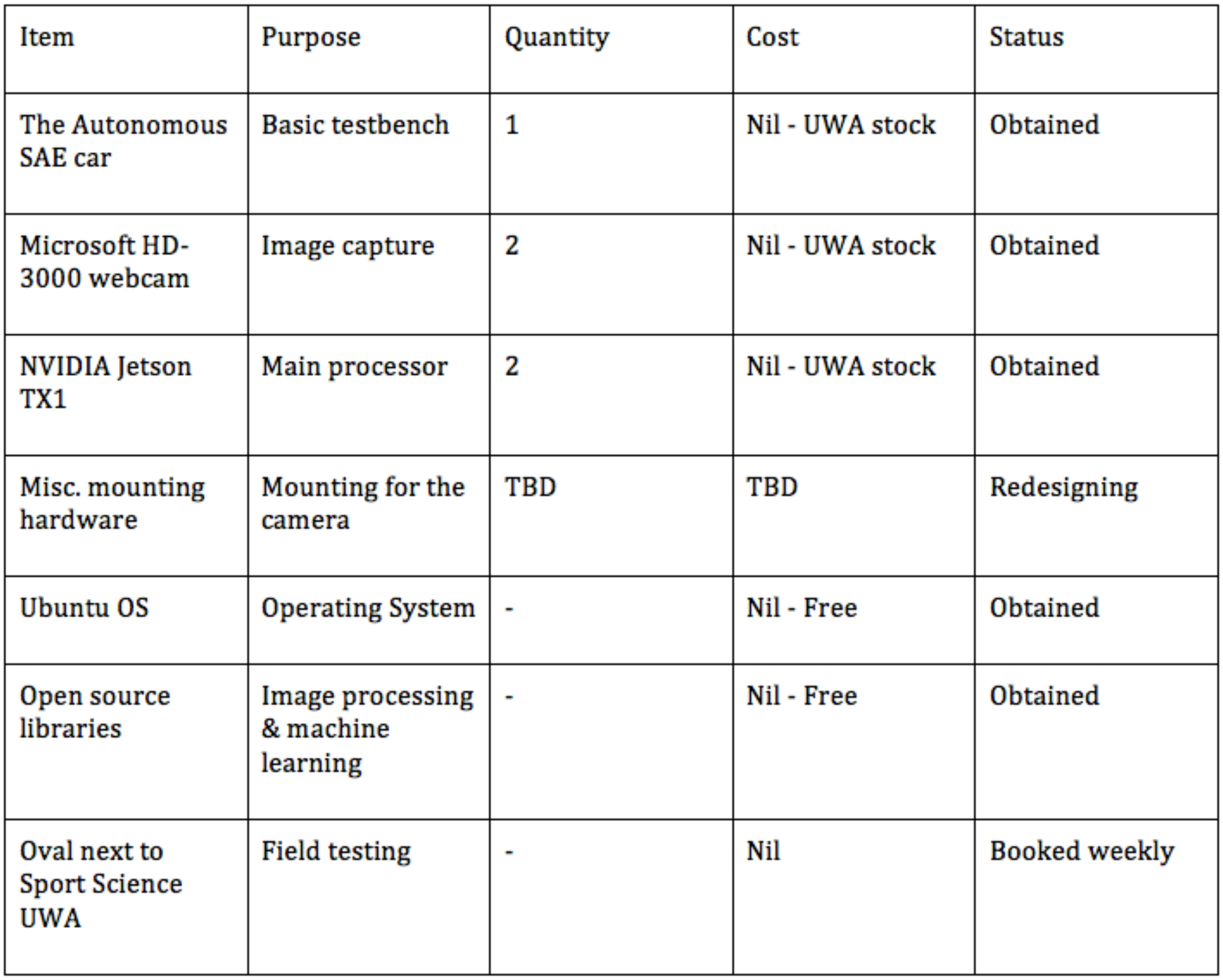
Figure 8 Cone detector. (red area for small size window, blue area for medium size window and green area for large size window)

Current issues:

- Too much false positives on collected testing videos.
  - Solution: Retrain the SVM using new image samples. Introduce more negative samples into the SVM training dataset.
- The classification under strong lighting condition is poor.
  - Solution: Do preprocessing before feature extraction to normalise the intensity of the image.
- The runtime performance is too low.
  - Solution: Change the feature used into ORB, and enable GPU acceleration with LIBSVM [21].

5. Resources

Table 1 List of resources used in this thesis

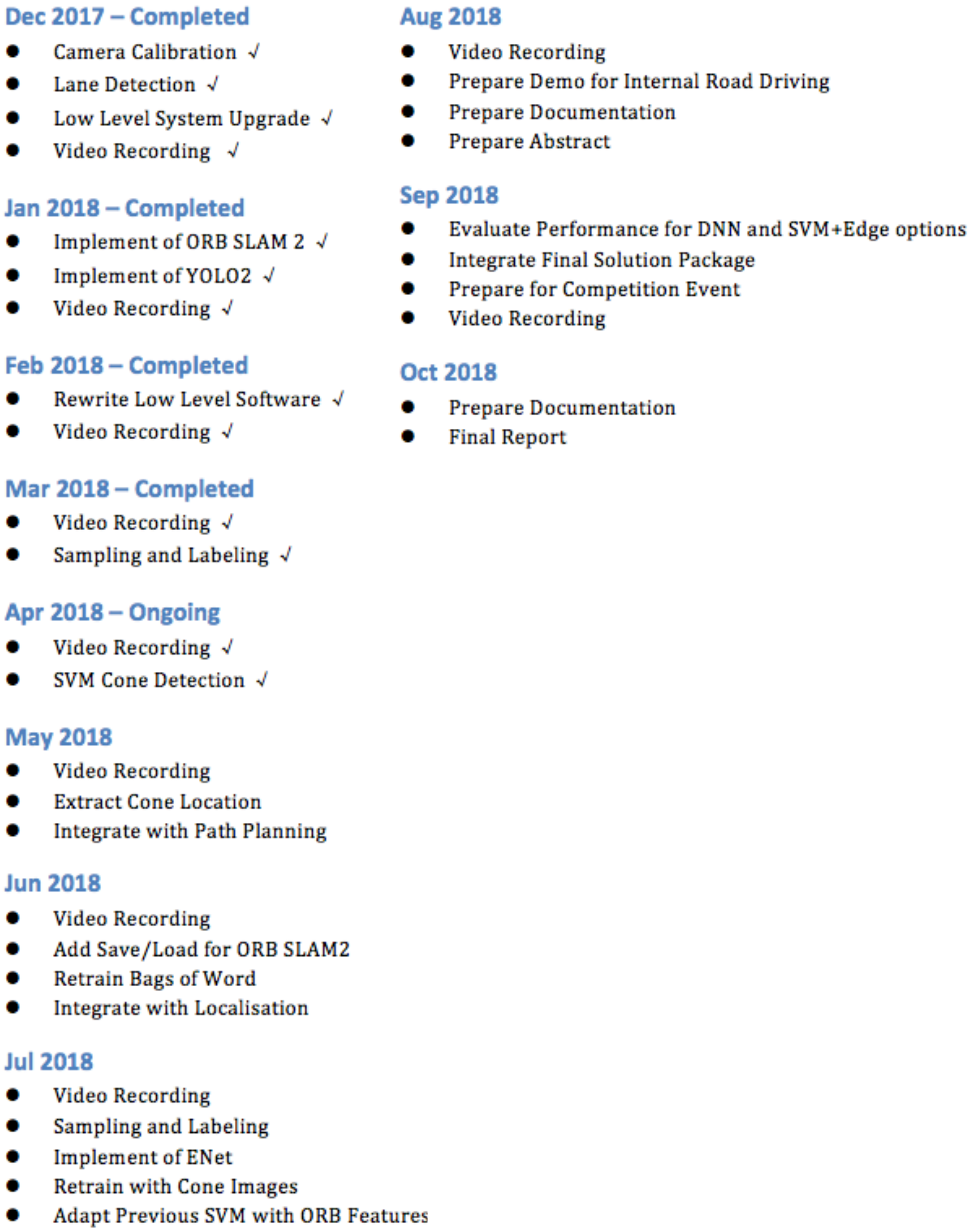


Item	Purpose	Quantity	Cost	Status
The Autonomous SAE car	Basic testbench	1	Nil - UWA stock	Obtained
Microsoft HD-3000 webcam	Image capture	2	Nil - UWA stock	Obtained
NVIDIA Jetson TX1	Main processor	2	Nil - UWA stock	Obtained
Misc. mounting hardware	Mounting for the camera	TBD	TBD	Redesigning
Ubuntu OS	Operating System	-	Nil - Free	Obtained
Open source libraries	Image processing & machine learning	-	Nil - Free	Obtained
Oval next to Sport Science UWA	Field testing	-	Nil	Booked weekly

In this thesis, as most of the hardwares are provided by the REV project team, the cost related to the hardwares is the cost for mounting parts. However, as the team is currently seeking sponsor for a better quality camera, there are potential costs for the new devices.

The operating system and software libraries used in this thesis are all open source project, thus there will be no cost associated with the use of these softwares following the corresponding open source licenses.

6. Project Timeline



Dec 2017 – Completed

- Camera Calibration ✓
- Lane Detection ✓
- Low Level System Upgrade ✓
- Video Recording ✓

Jan 2018 – Completed

- Implement of ORB SLAM 2 ✓
- Implement of YOLO2 ✓
- Video Recording ✓

Feb 2018 – Completed

- Rewrite Low Level Software ✓
- Video Recording ✓

Mar 2018 – Completed

- Video Recording ✓
- Sampling and Labeling ✓

Apr 2018 – Ongoing

- Video Recording ✓
- SVM Cone Detection ✓

May 2018

- Video Recording
- Extract Cone Location
- Integrate with Path Planning

Jun 2018

- Video Recording
- Add Save/Load for ORB SLAM2
- Retrain Bags of Word
- Integrate with Localisation

Jul 2018

- Video Recording
- Sampling and Labeling
- Implement of ENet
- Retrain with Cone Images
- Adapt Previous SVM with ORB Features

Aug 2018

- Video Recording
- Prepare Demo for Internal Road Driving
- Prepare Documentation
- Prepare Abstract

Sep 2018

- Evaluate Performance for DNN and SVM+Edge options
- Integrate Final Solution Package
- Prepare for Competition Event
- Video Recording

Oct 2018

- Prepare Documentation
- Final Report

Reference

[1] Raúl Mur-Artal and Juan 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, June 2017.

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

[3] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55-81, Jan 2015.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

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

[6] Anshika Sharma, Pradeep Kumar Singh, and Palak Khurana, "Analytical review on object segmentation and recognition," in *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, Noida, India, Jan 2016.

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

[8] David G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proceedings of the International Conference on Computer Vision*, Sep 1999, pp. 1150–1157.

[9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision*, 2006.

[10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision*, 2010.

[11] Ethan Rublee, Vincent Rabaud, and Kurt Konolige, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Barcelona, Spain, 2011.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks,".

[13] Joseph Redmon, Santosh Divvala, and Ross Girshick, "You Only Look Once: Unified, Real-Time Object Detection,".

[14] Ross Girshick, Jeff Donahue, and Trevor Darrell, "Rich feature hierarchies for accurate object detection and semantic segmentation,".

[15] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,".

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

[17] NVIDIA. Programming Guide : CUDA Toolkit Documentation. [Online]. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[18] Yao-Tsu Lin, "Autonomous SAE Car – Visual Base Road Detection," *School of Electrical, Electronic and Computer Engineering*, Oct 2017.

[19] OpenCV team. (2018) Open Source Computer Vision Library. [Online]. https://opencv.org

[20] NumPy developers. (2017) NumPy. [Online]. http://www.numpy.org/

[21] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011.