Towards Self Driving: Sensing and Learning Interim report

Michael McDonnell*

UNSW Canberra at ADFA.

Is an abstract required for the interim report? What is the wording style for it? The aim of this project is to investigate localising navigation data from a GPS feed to the observed road via a vehicle mounted camera. A subordinate aim as part of this is to develop a simulation that will provide sensor data to an external program for processing. The early focus for this project was developing core competencies in Image Processing and Computer Vision techniques which feed in to the design of the localisation system architecture. OpenCV in Python will be used for the sensor analysis which includes GPS position data, navigation route information and video feed from a simulated vehicle. The simulation design and tool analysis was conducted with the key technical risks addressed. The simulation is being developed in Unity and requires inter process communication between the Unity process and the external Python process. This has been developed using the ZeroMQ library and the simulation uses dynamic time scaling to ensure that sensor data processing occurs at a desired rate. The simulation development has now entered an agile design approach with iterative functionality improvements to be delivered. The navigation localisation problem has key technical risks in curved lane and intersection detection and correlation of the detected features with the navigation data. Several candidate approaches to address these risks have been identified and work in these areas will be progressing in the short term.

*CAPT, School of Engineering and Information Technology, ZEIT4901

Contents

I Introduction **3**

A Terminology 4

B Project aim 4

C Scope and Deliverables 5

II Project management **5**

A Project Methodology 5

III Current work **6**

A Computer Vision 6

1 Open CV 7

2 Language choice 7

B Simulation system 8

1 Simulation engine 8

2 Interprocess Communication (IPC) 8

3 Simulation time dilation 9

IV Future work **9**

A Computer Vision 10

1 Lane detection 10

2 Intersection detection 10

B Navigation localisation 11

C Simulation 11

1 Vehicle controller 11

2 Road network 11

3 GPS sensor 12

4 GPS navigation 12

5 Possible Extensions 12

V Conclusions **12**

A Gantt Chart for Research Project **14**

I. Introduction

TODO: REFERENCES DO REFERENCING THE FULL SECTION - SHOULD BE III-B-2, NOT JUST 2

The idea of a future where personal transportation is handled by autonomous vehicles is increasingly in the public consciousness however there is a range of challenges that need to be addressed from legal, security and ethical (GM, 2018) to maturity concerns that out to the 2030s, 'autonomous vehicles will be expensive novelties' (Litman, 2019). Autonomous driving is not a binary capability however, rather a scale with increasing levels of autonomy. The automation levels identified by the US National Highway Traffic Safety Administration (NHTSA, 2017) outlined in table 1. While true full automation may be many years off, there is an undeniable increase in the cognitive assistance and partial automation technologies in consumer vehicles. As an example the 2019 Kia Sorento includes active lane keeping assist (lane detection and steering) and adaptive cruise control (autonomous acceleration and braking based off radar distance to leading vehicles) (KIA, 2019).

Table 1. automation levels identified by the US National Highway Traffic Safety Administration (NHTSA, 2017)

Level	Classification	Detail
0	No Automation	Zero autonomy; the driver performs all driving tasks.
1	Driver Assistance	Vehicle is controlled by the driver, but some driving assist features may be included in the vehicle design
2	Partial Automation	Vehicle has combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all times.
3	Conditional Automation	Driver is a necessity, but is not required to monitor the environment. The driver must be ready to take control of the vehicle at all times with notice.
4	High Automation	The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.
5	Full Automation	The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.

In order for an autonomous vehicle to navigate effectively there are a few key challenges. The vehicle must have a mechanism to sense the local environment for example lane and road detection as well as transient aspects such as other vehicles and on road obstacles. There are many Computer Vision techniques that can assist in providing an understanding of the environment. Direct techniques such as edge and line detection are widely used however Deep Convolutional Neural Networks have also been shown to be effective for road detection (Oliveira et al., 2016).

In addition to the local area, the vehicle must also have the ability to reconcile navigation data with the current location. A supporting concept to this is that of map matching which calculates out the vehicle location by using the geographical information from sensors such as GPS position and inertial data and the map information from the mapping service (Zhao et al., 2018). Current high end self driving vehicles require high fidelity 3D maps to operate effectively. Despite this, position localisation improvements have been achieved using a data fusion of GPS and inertial navigation system data (Qingmei Yang and Jianmin Sun, 2007) correlating a back lane mark registry supported with CV, GPS and inertial data with map data (Vivacqua et al., 2017) and through use of Kalman filters and LIDAR in more complex environments (Bosse and Zlot, 2008).

This process of combining data from several sources into a single unified description of a situation is known as data fusion (Qingmei Yang and Jianmin Sun, 2007). Self driving vehicles rely on sensor and data fusion to achieve the four capabilities required of autonomous driving; car navigation system, path planning, environment perception and car control (Zhao et al., 2018). This project touches on elements of the first three capabilities in order to localise navigation information. The localisation of navigation data will be achieved by the use of GPS position data and computer vision techniques to correlate data from preloaded maps.

The ability to use computer vision to align GPS positioning with mapping service data will allow greater cognitive assistance in driver included tasks without the requirement for significant prior mapping. As augmented reality technology increases this will also allow driver aides such as navigation routes overlayed onto the visible road. In addition to the cognitive assistance, the ability to localise a navigation route allows the automation of certain tasks such as military field logistics supply lines. A true complete solution will rely on sensor fusion from a suit of complimentary sensors supporting each other and providing redundancy. The idea of local navigation goals has been explored with the use of Open Street Map data and LIDAR for road mapping with promising success (Ort et al., 2018) however the ability for a single camera system to facilitate navigation data localisation assists in both system redundancy and lowering the financial and technical barriers to implementation.

A. Terminology

The following abbreviations and definitions are used throughout this report:

- **Computer Vision (CV)**. Techniques to allow machines to 'see'; processing visual images of the world and deriving understanding.
- **Navigation localisation**. Definition definition
- **WORD (abbrev)**. Definition definition
- **WORD (abbrev)**. Definition definition
- **WORD (abbrev)**. Definition definition
- **WORD (abbrev)**. Definition definition
- **WORD (abbrev)**. Definition definition
- **TODO: DIP, External Program, Simulation 'tick', Open Street Map, Polyline**

B. Project aim

The aim of this project is to investigate localising navigation data from a GPS feed to the observed road via a vehicle mounted camera. A subordinate aim as part of this is to develop a simulation that will provide sensor data to an external program for processing.

GPS route data is held as encoded polylines which represents a road as a series of connected points (nodes) (Google, 2018). The approach for the positional localisation is to use the vehicle GPS position as an approximate input location and detected road features to determine an accurate position of the vehicle and identify the GPS route on forward facing video feed. This sets the conditions for autonomous control of the vehicle based on a programmed navigation route.

A subordinate aim to the project is the development of a simulation which provides a sensor data, such as GPS location and video feeds from a simulated vehicle. This simulation will allow both the generation of a large and varied data set for testing as well as allow the ability to provide control feedback for the simulated vehicle based off the processing of the sensor data.

The data flow is anticipated to be as follows:

- Simulated sensor data passed to external program
- External program processes data:
  - Road/Lane detection
  - Curve/Map matching of current GPS area
  - Reprojection of desired direction onto road image
- External program returns control data (as applicable) to simulation for next 'tick'

C. Scope and Deliverables

The scope of the project is deliberately kept constrained initially. This is to focus on the specific problem of localising a navigation route without losing development effort to supporting elements. While the scope is initially narrowly defined it can be extended towards the end of the project as time and opportunity allow.

The scope and deliverables have been identified as follows:

- **Limitation of road complexity.** There is a requirement for road/lane detection as part of this project (to marry up with the GPS polyline data) however it is not the main focus of the project. Further as the project purely uses the output data of lane detection, it can be considered a 'black box' and implementations can be swapped out as more advanced options are identified. The initial limitations on scope of road detection includes:
  - Limit road detection to easily detectable road surface
  - Limit roads to single lane
- **TODO: OTHER SCOPE?!??!**
- **Simulation deliverable requirements.** A core supporting element of this project is the development of the simulation. In addition to providing data for this project the intent is for the simulation to be held as an asset within SEIT for use in subsequent student projects in this area. The basic requirements for the simulation are:
  - Ability to provide 3D video feed of simulated driving to external program
  - Support simulated GPS tracking data
  - Support simulation of GPS route guidance

Based on the levels outlined in table 1 the intended level is to level 2 or 3 for cognitive assistance and in automated remote logistics applications such as autonomously traversing known routes in a military field setting to level 4.

II. Project management

A. Project Methodology

The initial state of the project was the field of autonomous vehicles as the general area of focus. As a result the preliminary phase of the project was the identification of the 'problem area' and narrowing of scope. A broad reading of relevant research and industry articles identified the ability to navigate in arbitrary areas as a candidate problem area and the scope was refined as outlined in section I.

This project is being undertaken in an area of study that is a new field for the author. As a result the early focus was on developing base competencies in digital image processing and general computer vision. This included both an understanding of the theory and mathematics behind DIP and CV tools and familiarity with implementation options. This extended into more specific areas such as straight lane detection.

In parallel with the CV competency development, research and experimentation on key technical risk elements of the simulation was conducted. The core technical risk was the ability for simulation code to communicate with sensor processing code. The aim was to keep these two code bases separate to allow other individuals to use the simulation for relevant purposes without being tied in to the aims of this project.

Once the core competencies have been developed and the supporting tool options have been analysed the focus will split between agile development of the simulation and the development of the external processing program. The simulation development will consist of 'sprint' periods designed around providing incremental functionality, prioritised as needs arise. The development of the external processing forms the specific primary aim and will include the following core milestones:

- Curved lane detection
- Intersection detection and identification of discrete roads
- Mapped road position estimation based off the GPS position and nearby road map nodes (accounting for GPS inaccuracy)
- Map matching, specifically some form of curve or spline matching to correlate the estimated map position with the identified road features

- Consolidation of return data. This may include:
  - Overlaying the navigation route directly to the video feed
  - Providing control inputs to the vehicle for autonomous driving of the route

Additional considerations which will be addressed as part of the simulation development include the data structure representation of the GPS maps and navigation routes. These will both be in line with the OSM data structures but may be simplified to specifically the required data for ease of processing in testing. Simplification of data will only occur where it is feasible that the OSM (or similar) data format can be transformed into the used data format with sufficient preprocessing. **TODO: ie. Using only information that is present in the OSM or similar data TODO: Gantt Chart discussion and update (Is this a subsection or just another para?)**

III. Current work

The bulk of the initial effort has been developing technical competency within informing disciplines. This includes both the CV technical competency development and addressing the simulation technical risks. Additionally to these aspects, familiarity projects were conducted on neural networks to consolidate the basic theory for potential future use. The projects consisted of developing neuroevolution based solutions for both a physics based ball balancing challenge over the duration of 5 minutes and custom cloned version of the mobile game 'flappy bird'. Both solutions used a simple feedforward neural network coded in C# without the use of any existing libraries with the networks evolving using a custom built genetic algorithm. The CV and simulation aspects are discussed in more detail in the following subsections.

A. Computer Vision

Before any effective computer vision approaches can be implemented a robust base of understanding of digital image processing is required. In order to develop a robust understanding of DIP concepts study was undertaken using a range of publicly available resources. The main resource used for structured learning was the Spring 2015 offering of ECSE-4540 at Rensselaer Polytechnic Institute by Rich Radke which has the full set of recorded lectures freely available online. Basic algorithms were implemented from first principles in MATLAB which reinforced the internalisation of the concepts. More advanced computer vision concepts were subsequently researched and tested within Python using OpenCV functionality which is discussed further in this section (subsections 2 and 1).

Simple lane detection approaches for straight lanes using the Hough Transform were implemented with varying levels of success. It was noted that the simple approaches suffered when the dashed lane line was faint in comparison with other linear features and when the road was cast in shadow. The final experiment to improve on detection involved lane detection on a video feed using a rolling average of the lower area of interest of the most recent 5 frames of video as input to the lane detection for each frame. The lane detection approach involved identifying straight lines using the Hough Transform of the Canny edge detected area of interest after a low pass filter had been applied. The Hough lines were then chosen based on strength with the strongest line for each angle (side of the lane) chosen. The output consisted of the averaged area of interest with the detected lane lines in red overlayed onto the original frame. An example frame from the video output is included as figure 1.

THE FLOW OF CV IN LANE DETECTION involves correcting the image for camera lenses before inverse perspective mapping to get a 'birds eye view' of the road. From this top down perspective, road/lane analysis can be conducted properly. An inverse perspective mapping calibration was developed using the interprocess communication implementation discussed in section 2. This calibration consists of still image of a checkerboard pattern on the horizontal plane within the simulation taken from the perspective of the camera. Corners of the checkerboard are manually identified in pixel space and the in-

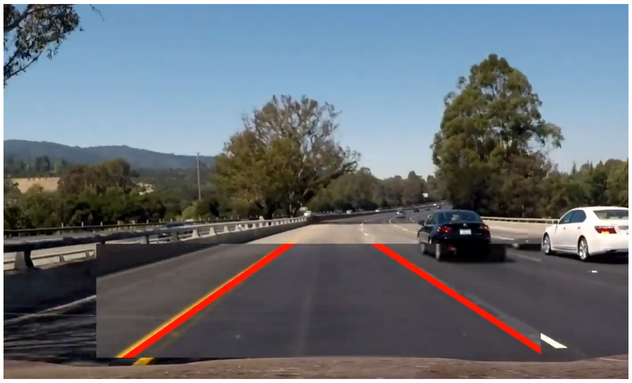


Figure 1. Still from video feed lane detection experiment

verse perspective matrix is calculated based on the current old (perspective) and desired new (inverse perspective) positions. Automated identification of checkerboard corners is possible however the simulation camera rotation with reference to the horizontal plane is fixed therefore only a single calibration is required. The input and output of the inverse perspective mapping calibration undertaken is included as figure 2.

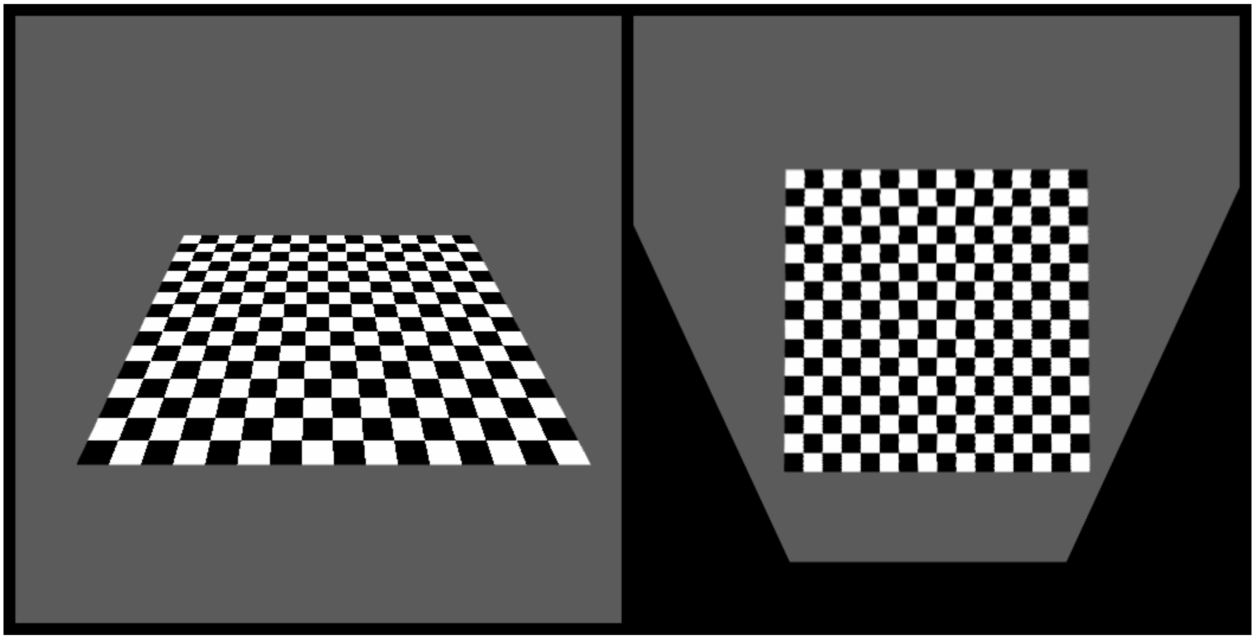


Figure 2. Inverse perspective mapping calibration. Original camera feed *(left)* and inverse perspective mapped image *(right)*

Key considerations within the Computer Vision implementation decision process are in the following subsections.

1. Open CV

The DIP theory that was undertaken opened up the possibility to implement the CV functionality from first principles, indeed basic tools such as filtering had already been implemented in MATLAB experiments. The alternative option considered was to use an open source computer vision library called OpenCV. OpenCV has C++, Python, Java and MATLAB interfaces and is widely used in commercial applications in companies such as Google, IBM and Honda (OpenCV).

While implementing all functions manually has merit for further reinforcement of the basic principles, the decision was made to use the OpenCV library. Rationale for this includes the following:

- Efficient high performance vectorised implementations
- Allows a greater proportion of time to be spent on solving the problem as opposed to implementing known algorithms.
- Increase familiarity with available tools for future use

2. Language choice

Given the choice of OpenCV for the CV framework the options considered for the core language were C++ or Python. C++ has the advantage of being 'lower level' however the OpenCV Python interface compiles down to the same C++ code so the only performance gains would be in the general program code as opposed to the CV implementations. Simulation engine considerations also played a part, as discussed in section 1, Unreal Engine uses C++ which would allow simulation code to talk directly to the OpenCV interface.

Despite the above advantages, C++ comes with an increased complexity and any speed gains require a robust understanding (and implementation) of memory management. By contrast, Python presents a lower barrier to entry while maintaining the OpenCV C++ vectorised optimisation. In addition, libraries such as numpy within Python present fast vectorised operations for arrays. The decision to use Unity for the simulation engine as discussed in section 1 removed the major benefit of the C++ interface thus Python was determined to be the preferred language.

B. Simulation system

A significant element of the project deliverable is the development of the simulation system to provide sensor outputs. The intent of the simulation is to provide a pipeline of sensor feeds and potentially an interface to control a simulated vehicle based off the processed sensor data. The scope of the simulation is discussed in section C. Key elements of the simulation system are discussed in the following subsections.

1. Simulation engine

Two courses of action were available for the implementation of the simulation engine; a full custom engine or use an existing game engine for the base. For the existing game engine option, Unreal and Unity were both considered. A full custom engine allows fine grain control over the implementation the specific requirements of the simulation however has significant overheads including the implementation of 3D visuals including rendering, texturing and importing of 3D models. It was determined that this represented a very large time commitment for negligible benefit.

Both Unity and Unreal are professional 3D engine used in commercial games and computer graphics applications. They are well established and have a robust set of supporting tools. Specific considerations for each engine are as follows:

- Unity
  - Personal experience with workflow; very comfortable
  - Personal experience with C# which is used for scripting within the engine
  - No C# interface with OpenCV therefore interprocess communication will need to be developed between the OpenCV processing implementation and the simulation engine.
- Unreal
  - New, unfamiliar workflow
  - Lack of familiarity and comfort with C++ which is used for scripting within the engine. ‘Blueprint’ system does allow a visual programming approach although that represents an additional competency to be developed
  - C++ code allows direct interface with OpenCV

When considering the above factors and the OpenCV considerations discussed in section 1, Unity was chosen due to the ability to ‘hit the ground running’ based on workflow and language familiarity. It was noted that this accepts a large technical risk of IPC implementation however it was deemed to be the preferred option.

2. Interprocess Communication (IPC)

One of the main issues identified with using Python Open CV and the Unity game engine is the ability for simulation data, for example video feeds, to be transferred from Unity to the Python process. This was a significant technical challenge (REF GANTT CHART? HOW LONG WAS BUDGETED). The motivation for fast IPC was to allow real time processing of the simulation data in order to allow for potential control feedback to the simulation. Transferring image data represents a large amount of memory thus has a significant time complexity per simulation ‘tick’.

The initial research approach involved investigating the ability to use a static memory buffer that could be written to by Unity and read by Python. Shared memory or memory mapped files do offer this however they add another level of complexity and implementation details are largely for lower level languages and could not be identified in particular for Python. Named pipes were also investigated and, while they represented an improved option, still had many of the technical and implementation complexity issues of the memory mapped approach.

The final solution identified was using TCP via the ZeroMQ middleware. This has implementations for both C# and Python and is well optimised for speed on localhost (**TODO: What is localhost**) machines. The speed of this approach is highly dependant on the size of the transferred data **TODO: GRAPH OF TIME PERFORMANCE OF 64, 128, 256 AND 512....**. The use of this approach requires slower than real time simulation processing which was achieved by dynamic simulation time dilation, discussed in section 3. This approach allows easy and reliable two way communication which is required for effective feedback between the processes. This is something that would be complicated significantly by memory mapping, requiring multiple memory locations with varying roles. The current solution also offers the ability to expand off a single local machine, for example a hosted simulation with remote processing. While this is well out of scope of this thesis it presents additional flexibility for future development.

Once the IPC approach was confirmed, the data handling pipeline between the Unity (C#) and Python processes was developed. The final simulation output data structure for video feed is a flattened 3D byte array, consisting of the x and y pixel coordinates and the colour value. The colour is represented by a four element vector of bytes for the Red, Blue, Green and Alpha channels.

The Python process casts the byte stream as a byte array and reshapes the data to a 3D array. The fourth element of each colour value is removed, resulting in a final data structure consisting of a 2D array of 3 element vectors of bytes which is the required data structure for OpenCV processing operations.

A simple calibration was established using three coloured cubes (Red, Green and Blue) in Unity to confirm colour channel and axis orientation correctness. A still from the colour image IPC calibration undertaken is included as figure 3.

3. Simulation time dilation

While the initial design aimed for realtime simulation and processing it was quickly determined that the simulation had a requirement to dynamically scale simulation time. This scaling is based on the external processing in order to ensure the simulation effectiveness regardless of the data processing time requirement and allows both the simulation to be effective on lower end computers and the number of sensors to scale without affecting simulation quality.

The implementation of time dilation involved a simulation controller that runs the simulation for a set time based off the desired sensor processing rate. After this time has elapsed, the sensor data is collated and sent to the external processing program via the IPC process outlined above. On completion of the data processing, the external program sends an acknowledgement in return which triggers the next simulation tick. This has been implemented and tested by sleeping the processing thread for several seconds each tick.

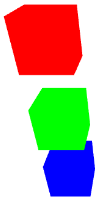


Figure 3. Still from calibration of colour image IPC pipeline between Unity and Python

The time dilation was tested using an initial prototype of the simulation in the same manner as the colour calibration discussed previously. Two stills from the output video showing the raw feed and the canny edges is included as figure

IV. Future work

The work completed thus far has solved some key technical challenges required, in particular on the simulation side. The core areas of work in the remainder of the project, with the associated assessed technical risk, are discussed in the coming subsections.

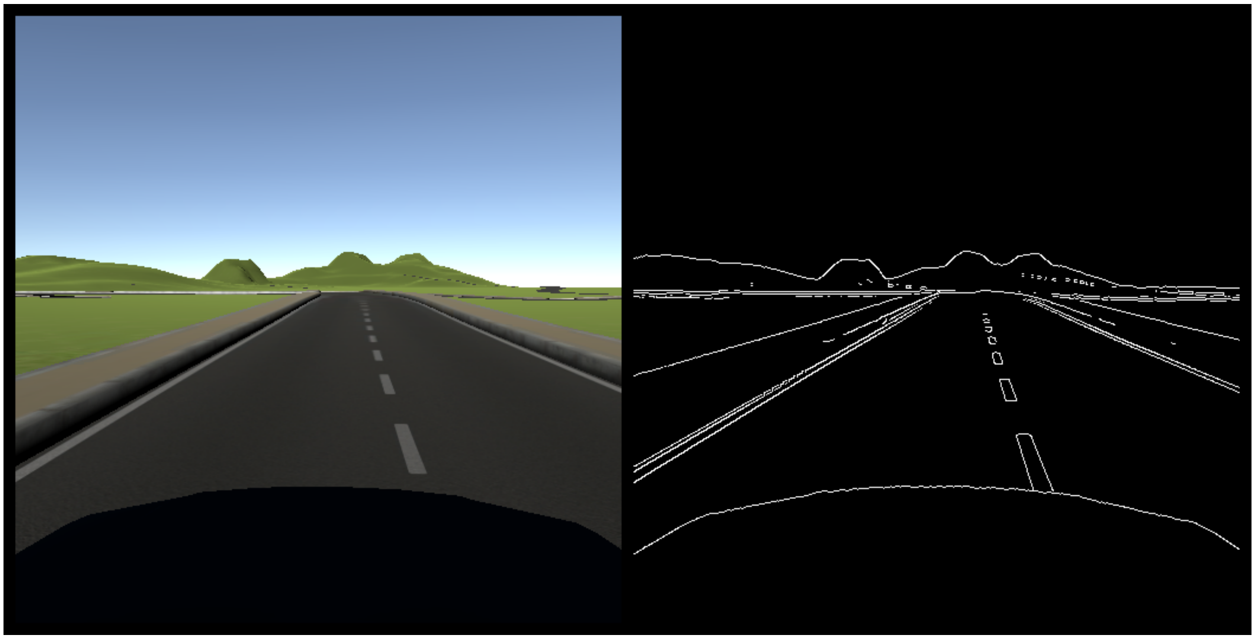


Figure 4. Still from video feed of simulation prototype test. Original camera feed *(left)* and canny edges of feed *(right)*

A. Computer Vision

1. Lane detection

Detection of straight lane lines has been completed in the familiarisation activities conducted as discussed in section A and demonstrated in figure 1. A harder challenge is identifying a curved lane. The current approach being investigated is lane edge detection via polynomial fitting of edge pixels detected by sliding window.

The sliding window detection process is as follows:

- Identify the near lane/road edge (bottom of the frame) by either edge detection or histogram peaks.
- A window of a defined width and height is placed over the detected start point and the average horizontal position of the edge pixels is determined
  - If the detected pixels is below a threshold, the average is set to the previous average
- The next window is placed one window height above the previous window at the previous average horizontal pixel position.
- This process is repeated until the window reaches an edge of the frame

The above process is completed for each side of the lane and polynomial regression is used on the relevant pixels to determine the final detected line.

In addition to the sliding window detection, supervised convolutional neural network lane detection has had success through fully convolutional (Zang et al., 2018) and instance segmented (Neven et al., 2018) approaches. This will be further investigated however the intent at this stage is to focus on the sliding window detection in the first instance. Another unique option involves spline based representation using random sample consensus for bezier splines based off road edge detection (Aly, 2008). This method has the benefit that it provides a different form of output (splines) that may be more suitable to matching with GPS data. The technical risk of this aspect is low to medium.

2. Intersection detection

Intersection detection presents a more challenging prospect than simple lane detection. The simplifying assumption of a single left and right lane edge is removed when intersections are introduced. One example of effective intersection detection as it applies to this project is model based recognition which matches intersection models to a series of road boundary points (Todd R. Kushner, 1987). The advantage of this method is edge detected pixels can be used as the boundary points and the intersection model can be built reasonably simply using the OSM map data.

An alternate approach that will be further investigated as required is using skeletonisation of the road binary mask:

- When the full road surface is represented by a binary mask, skeletonisation will reduce each road to a single pixel line. Intersections of the pixel lines will represent the intersection midpoint with each skeleton arm representing a road surface entering the intersection.
  - Skeletonisation can be achieved by morphological erosion/thinning operations or by obtaining the medial axis transform via the distance transform.

Once the skeleton has been obtained it can be matched to the possible intersection orientations in a similar way to the model based recognition initially outlined. It is assessed at this stage that the skeletonisation is likely a redundant step however it is an option if the road boundary points approach proves unreliable. The technical risk of this aspect is medium.

B. Navigation localisation

The final step involves correlating the GPS positional data, features from video sensor feed and the navigation data to localise the navigation data to the vehicle position. Static route map matching at its most trivial relates a GPS position to the nearest point on a polyline. It is anticipated that this will be the first step in the localisation to approximate the position on the navigation route. After this approximation, the road and intersection detection will be correlated with the GPS data to fully localise the position.

A LiDAR-based local trajectory generation has demonstrated through practical experiments that effective localisation using OSM data and global waypoints is possible using a LIDAR sensor suite Ort et al. (2018). General curve or spline matching options will be investigated and it has been identified that Markov and extended Kalman filter localisation techniques may also assist in this problem (Thrun et al., 2005) and will be investigated in the near term. The technical risk of this aspect is medium.

C. Simulation

The simulation development will follow the agile development process and deliver incremental feature additions and improvements. The basic features which are high priority for the early sprints are the implementation of the functionality required for the minimum viable product; the vehicle controller, the road network and the GPS sensor and navigation implementations. These aspects, as well as candidate functionality extensions are discussed in more detail in the coming subsections.

1. Vehicle controller

The vehicle controller will be implemented using the built in PhysX components within the Unity engine. The basic vehicle setup will consist of steering, accelerating and braking functions with vehicle movement simulated by the physics engine. An autopilot function that will follow defined waypoints will also be developed. It is important to note that this is not based on the external analysis of sensor input but will use the exact game data. The intent of this functionality is to provide a constant video feed of a driving vehicle for analysis.

Features such as engine power simulation will not be implemented in the initial stages but may be introduced as future features. The technical risk of this aspect is very low.

2. Road network

The road network is being implemented using a purchased tool called Easyroads 3D. This tool assists in mesh generation, placement, connection and texturing of roads as well as terrain moulding. Additionally the tool can accept map data exported from OSM. While each of these functions can be custom built individually it represents a significant time commitment and is unrelated to the core simulation. The road visible in the raw video feed of the IPC prototype test in figure 4 demonstrates the implementation of a test road in Unity. The initial implementation of the road network will be simple tracks. The technical risk of this aspect is low.

3. GPS sensor

The GPS sensor implementation is a simple function and required for the base sensor data analysis. The functionality will use the in engine coordinates for the GPS coordinates, as will all other data and sensors. An initial static random position offset will be used to simulate GPS positional error however additional functionality improvements may include a greater simulation of GPS error. The technical risk of this aspect is very low.

4. GPS navigation

The navigation functionality will initially be via hardcoded hand placed waypoints, using a simple queue system, with waypoint positions using the in engine coordinate system. This represents the minimum viable product. There is significant room for expansion of functionality here including the ability to automatically route plan based off the OSM data used to build the road network. This aspect has significant room for scope creep so will be managed tightly in the sprint reviews. The technical risk of this aspect is low to medium.

5. Possible Extensions

The implementation of further functionality is based on the sprint velocity achievable and the needs at the time. Once the minimum viable product is delivered with the above functionality it is likely that improvements to existing functionality will take a high priority in future sprints however additional functionality will be considered in the sprint backlogs.

Possible future extension ideas include:

- External sensor definition files for automatic setup in simulation
- External OSM data file loading
  - Implementing this will also result in the requirement for automated data extraction from OSM files for simulated GPS and navigation rather than using hand placed nodes of interest.
- Increased communication options between simulation and external process, for example vehicle control signals or specific sensor control/activation/deactivation.
- 3D visual clutter such as trees

V. Conclusions

The main technical risks for the simulation have been successfully addressed and the development of the simulation is ready to enter the agile development process. This will deliver iterative improvements in functionality and provide a wealth of sensor data for analysis testing. The main technical risks for the sensor data processing include the detection of curved roads and intersections and the navigation localisation. Of these risks, the intersection detection is the most significant risk as it has the burden of providing an output that can be effectively used by the navigation localisation step. Despite the identified risks, the research conducted has identified candidate solutions for each area of technical risk and rapid iteration through the areas of technical risk will assist in assessing the relative merit of the identified approaches.

Appendices

A - Gantt chart

References

M. Aly. Real time detection of lane markers in urban streets. In *2008 IEEE Intelligent Vehicles Symposium*, pages 7–12, June 2008. doi: 10.1109/IVS.2008.4621152.

M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *I. J. Robotic Res.*, 27:667–691, 06 2008. doi: 10.1177/0278364908091366.

GM. General motors 2018 self-driving safety report, 2018. URL https://www.gm.com/content/dam/company/docs/us/en/gmcom/gmsafetyreport.pdf.

Google. Encoded polyline algorithm format (google maps documentation). Website, Dec. 2018. URL https://developers.google.com/maps/documentation/utilities/polylinealgorithm.

KIA. Kia sorento vehicle information, 2019. URL https://www.kia.com/us/en/vehicle/sorento/2019.

T. Litman. Autonomous vehicle implementation predictions: Implications for transport planning, Mar. 2019. URL https://www.vtpi.org/avip.pdf.

D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. Towards end-to-end lane detection: an instance segmentation approach. pages 286–291, 06 2018. doi: 10.1109/IVS.2018.8500547.

NHTSA. Automated driving systems 2.0: A vision for safety. Online, Sept. 2017. URL https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf.

G. L. Oliveira, W. Burgard, and T. Brox. Efficient deep models for monocular road segmentation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4885–4891, Oct 2016. doi: 10.1109/IROS.2016.7759717.

OpenCV. Open Source Computer Vision Library. URL https://opencv.org/about/.

T. Ort, L. Paull, and D. Rus. Autonomous vehicle navigation in rural environments without detailed prior maps. pages 2040–2047, 05 2018. doi: 10.1109/ICRA.2018.8460519.

Qingmei Yang and Jianmin Sun. A location method for autonomous vehicle based on integrated gps/ins. In *2007 IEEE International Conference on Vehicular Electronics and Safety*, pages 1–4, Dec 2007. doi: 10.1109/ICVES.2007.4456376.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005. ISBN 0262201623. URL https://www.ebook.de/de/product/3701211/sebastian_thrun_wolfram_burgard_dieter_fox_probabilistic_robotics.html.

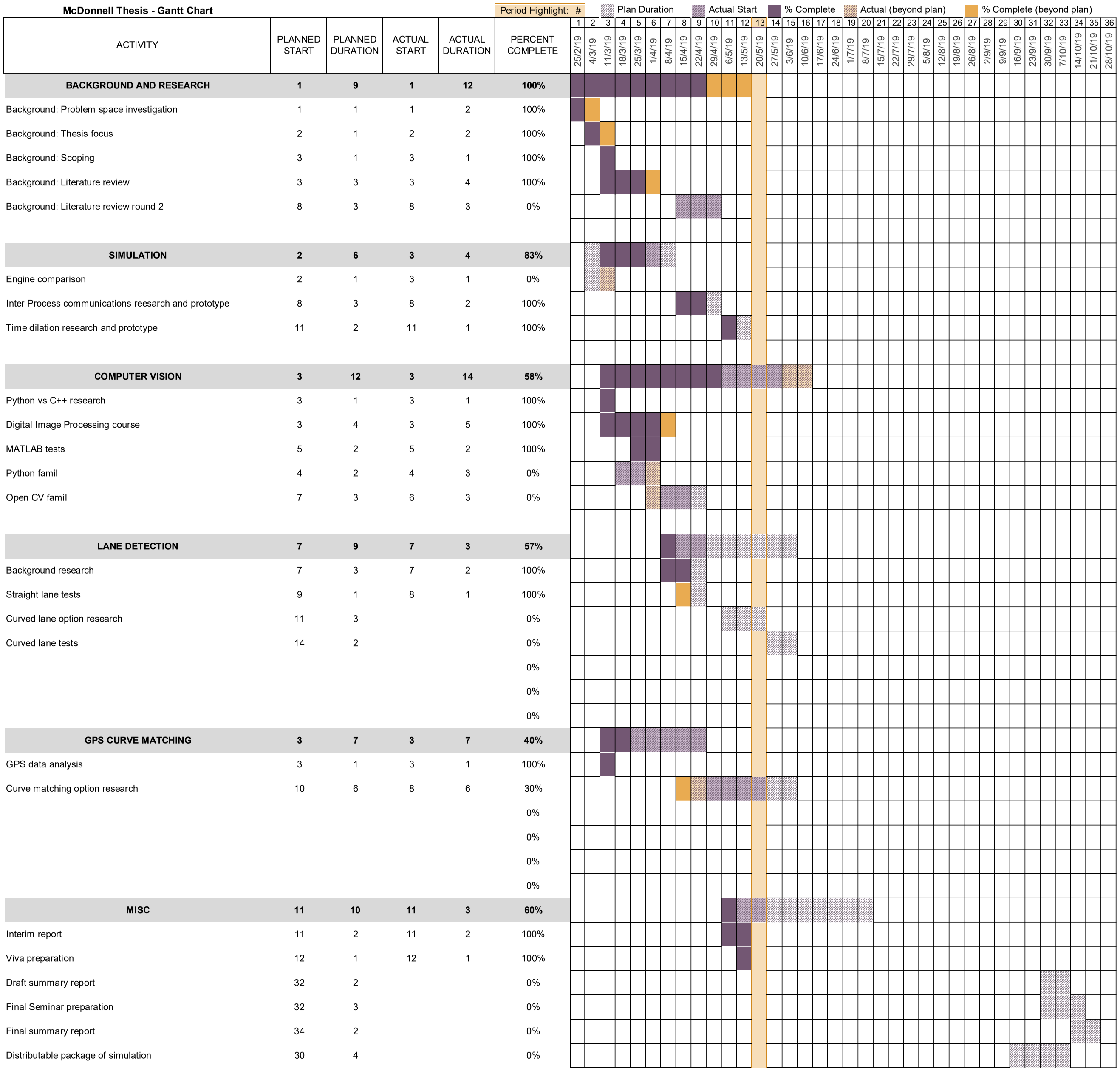
S. P. Todd R. Kushner. Progress in road intersection detection for autonomous vehicle navigation, 1987. URL https://doi.org/10.1117/12.968232.

R. P. D. Vivacqua, R. F. Vassallo, and F. N. Martins. A low cost sensors approach for accurate vehicle localization and autonomous driving application. In *Sensors*, 2017.

J. Zang, W. Zhou, G. Zhang, and Z. Duan. Traffic lane detection using fully convolutional neural network. pages 305–311, 11 2018. doi: 10.23919/APSIPA.2018.8659684.

J. Zhao, B. Liang, and Q. Chen. The key technology toward the self-driving car. *International Journal of Intelligent Unmanned Systems*, 6(1):2–20, 2018. doi: 10.1108/IJIUS-08-2017-0008. URL https://doi.org/10.1108/IJIUS-08-2017-0008.

A. Gantt Chart for Research Project



McDonnell Thesis - Gantt Chart

ACTIVITY	PLANNED START	PLANNED DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
BACKGROUND AND RESEARCH	**1**	**9**	**1**	**12**	**100%**
Background: Problem space investigation	1	1	1	2	100%
Background: Thesis focus	2	1	2	2	100%
Background: Scoping	3	1	3	1	100%
Background: Literature review	3	3	3	4	100%
Background: Literature review round 2	8	3	8	3	0%
SIMULATION	**2**	**6**	**3**	**4**	**83%**
Engine comparison	2	1	3	1	0%
Inter Process communications reesarch and prototype	8	3	8	2	100%
Time dilation research and prototype	11	2	11	1	100%
COMPUTER VISION	**3**	**12**	**3**	**14**	**58%**
Python vs C++ research	3	1	3	1	100%
Digital Image Processing course	3	4	3	5	100%
MATLAB tests	5	2	5	2	100%
Python famil	4	2	4	3	0%
Open CV famil	7	3	6	3	0%
LANE DETECTION	**7**	**9**	**7**	**3**	**57%**
Background research	7	3	7	2	100%
Straight lane tests	9	1	8	1	100%
Curved lane option research	11	3			0%
Curved lane tests	14	2			0%
					0%
					0%
					0%
GPS CURVE MATCHING	**3**	**7**	**3**	**7**	**40%**
GPS data analysis	3	1	3	1	100%
Curve matching option research	10	6	8	6	30%
					0%
					0%
					0%
					0%
MISC	**11**	**10**	**11**	**3**	**60%**
Interim report	11	2	11	2	100%
Viva preparation	12	1	12	1	100%
Draft summary report	32	2			0%
Final Seminar preparation	32	3			0%
Final summary report	34	2			0%
Distributable package of simulation	30	4			0%