Towards self driving: Single camera navigation localisation
Interim report

Michael McDonnell*
UNSW Canberra at ADFA.

The aim of this project is to localise navigation data from GPS positional data to the observed road via a vehicle mounted camera. A subordinate aim which supports the main aim is to develop a simulation that will provide sensor data to an external program for processing. A general motivation for this project is the development of a 'low tech' solution which will enable automation of simple road movements, for example routine (non-tactical) military logistic resupply. The initial focus for this project was developing core competencies in Digital Image Processing and Computer Vision techniques which feed in to the design of the localisation system architecture. OpenCV in Python is used for the sensor data analysis which includes GPS position data, navigation route information and video feed from a simulated vehicle. The simulation design and tool analysis was conducted and key technical risks have been addressed. The simulation is being developed in Unity and requires interprocess 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 does not impact the time flow of the simulation. 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 with the likely solution being intersection model based matching with road edge pixels.

*CAPT, School of Engineering and Information Technology, ZEIT4901

Contents

I. Introduction

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 issues (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. For context, the automation levels identified by the US National Highway Traffic Safety Administration (NHTSA, 2017) are outlined in table 1. While it may be decades before true level 5 automation is developed, 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), which is approaching level 2 automation.

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 detection, as well as the ability to identify and track 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 vehicle location by using the geographical information from sensors such as GPS position, inertial data and map information from a mapping service (Zhao et al., 2018). Current cutting edge self driving vehicles require high fidelity 3D maps to operate effectively which are time consuming to develop and not adaptive to rapid local changes. Despite this, position localisation improvements have been achieved without high fidelity 3D mapping data using a data fusion of GPS and inertial navigation system data (Qingmei Yang and Jianmin Sun, 2007) correlating a detected back lane registry supported with computer vision, 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; navigation, 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. This project will use GPS position data and computer vision

techniques to correlate data from preloaded maps to localise localise navigation. The context for this project is general but will loosely use the goal of automating (non-tactical) military field logistics route transport. This will include simplifying assumptions of single lane roads/tracks without the requirement to consider other vehicles or traffic control. 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. This project will focus on a 'low tech' option to provide a minimum navigation capability that does not rely on more advanced tools such as LIDAR. Consideration of relevant literature will be further outlined in specific sections of current and future work as it is relevant.

In general, 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 also allows for more immersive driver aides such as navigation routes overlayed onto the visible road. In addition to the cognitive assistance, the ability to localise a navigation route opens the near future possibility for the automation of certain tasks in a military setting, for example routine field logistics supply transport. A true complete solution will rely on sensor fusion from a suit of complimentary sensors supporting each other and providing redundancy.

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.
- **Digital Image Processing (DIP)**. Use of computer algorithms to perform processing on digital images.
- **Navigation localisation**. Translating an overall navigation route into a local navigation goal.
- **Simulation**. The custom built autonomous vehicle simulation which provides sensor feed outputs.
- **External processing**. In the context of this report two programs are discussed, the simulation and 'external processing'. External processing refers to the standalone code which performs the computational calculations for navigation localisation.
- **Simulation tick**. One period of simulation processing. This is aligned to a specified update frequency that the external processing is running at 'in simulation'.
- **Interprocess Communication (IPC)**. The passage of data from simulation to external processes.
- **Open Street Maps (OSM)**. 'A collaborative project to create a free editable map of the world' (OSM, 2019). Comparable to Google maps.
- **Polyline**. A line defined by node coordinates. The line is drawn through all nodes in order, starting at the first and terminating at the last.

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. Mapping service GPS route data is often held as polylines which represents a road as a series of connected points or nodes (Google, 2018), (OSM, 2019). The approach for positional localisation is to use the vehicle GPS position as an approximate input location mapped to the closest point on a route. Detected road features are then used to determine an accurate position of the vehicle and identify the navigation 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 in 3D space. 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 on results from processing of the sensor data.

The data flow is anticipated to be as visualised in figure 1 and outlined as follows:

- Simulated sensor data passed to external program.

- External program processes data:
  - Road/Lane/Intersection detection
  - Curve/Map matching of current GPS area.
  - Control signals to simulation for next simulation tick.
  - Debug or visualisation output such as video or still images.
  - Potentially:
    * Reprojection of desired direction or route onto road image.
    * Identification of desired vector for local target.
    * Control signals for simulation vehicle control.

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. As outlined previously the loose framing of this project is the goal of automating military field logistics route transport without tactical considerations. While the scope is initially narrowly defined it will be extended towards the end of the project as time and opportunity allow.

The scope and deliverables have been identified as follows:

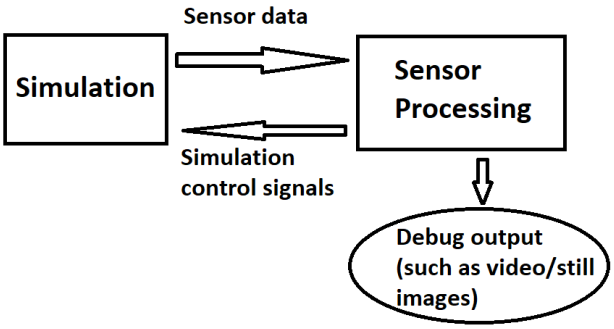


Figure 1. Anticipated data flow between simulation and sensor processing

- The solution must be able to reconcile GPS and CV data to identify the current location and required direction to travel through intersections based on a navigation route.
- **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 optimised road detection 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.
  - All roads considered will be of similar local colour and type.
- **Simulation deliverable requirements.** 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 navigation routes in a remote setting to level 4 autonomy.

II. Project management

Tracking of project progress is being maintained via a Gantt chart which is included as appendix A and is updated at least fortnightly based on current work. In addition to this, the development of the simulation

will use the Agile process with two week sprints with major monthly sprint reviews. Sprint tracking will be managed using Trello initially however may move to a more specialised solution as the project progresses.

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 DIP and general CV. This included both an understanding of the theory and mathematics behind DIP and CV tools and familiarity with implementation options and 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:

- 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 ie. only information that is present in, or able to be derived from, the OSM or similar data will be used.

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. This completed work is tracked in the Gantt chart in appendix A. Additionally to these aspects, familiarity projects were conducted on neural networks to consolidate the basic theory for potential future use. These 'toy' 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 DIP 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 in New York 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.

The Hough Transform was used to detect straight lanes with varying levels of success. The Hough Transform converts a global detection problem into a local peak detection problem in a parameter space (Illingworth and Kittler, 1988) where straight lines in an image can be represented by a single point in Hough space (Duda and Hart, 1972). A brief overview of simple lane detection as a conceptual primer for the interested reader, including a discussion on the Hough transform, is included as appendix B.

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 intermittent shadow. The final experiment to improve on detection involved lane detection on a video feed using a rolling average of a 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 2.

Figure 2. Still from video feed lane detection experiment. Lower area of interest where frames are averaged visible in the bottom central part of the image

In order to identify the true shape of the road, more advanced techniques are required. Inverse perspective mapping is 'a geometrical transform that attempts to remove the perspective effect present in an image captured by a camera pitched down towards a road' (D. Crisman and E. Thorpe, 1991a), allowing a birds eye view of the scene by weighing each pixel according to its information content (Bertozz et al., 1998). Inverse perspective mapping has many uses within the autonomous driving realm including vanishing point detection (Yang et al., 2018), attitude noise reduction (Yu Cao, 2011), distance determination (Tuohy et al., 2010), regularizing optical flow (Mallot et al., 1991), improving relative velocity determination (Tan et al., 2006) and detailed depth information in stereo (Bertozz et al., 1998). Importantly though, inverse perspective mapping is a key part in effective lane and obstacle detection (Mallot et al., 1991) (Tuohy et al., 2010) (Jun Wang et al., 2014) (Yang et al., 2018).

The basic approach to lane detection using CV then 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 for the simulation was developed using the IPC implementation discussed in section III-B-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 inverse 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 3. Key considerations within the Computer Vision implementation decision process are in the following subsections.

1. Use of OpenCV

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

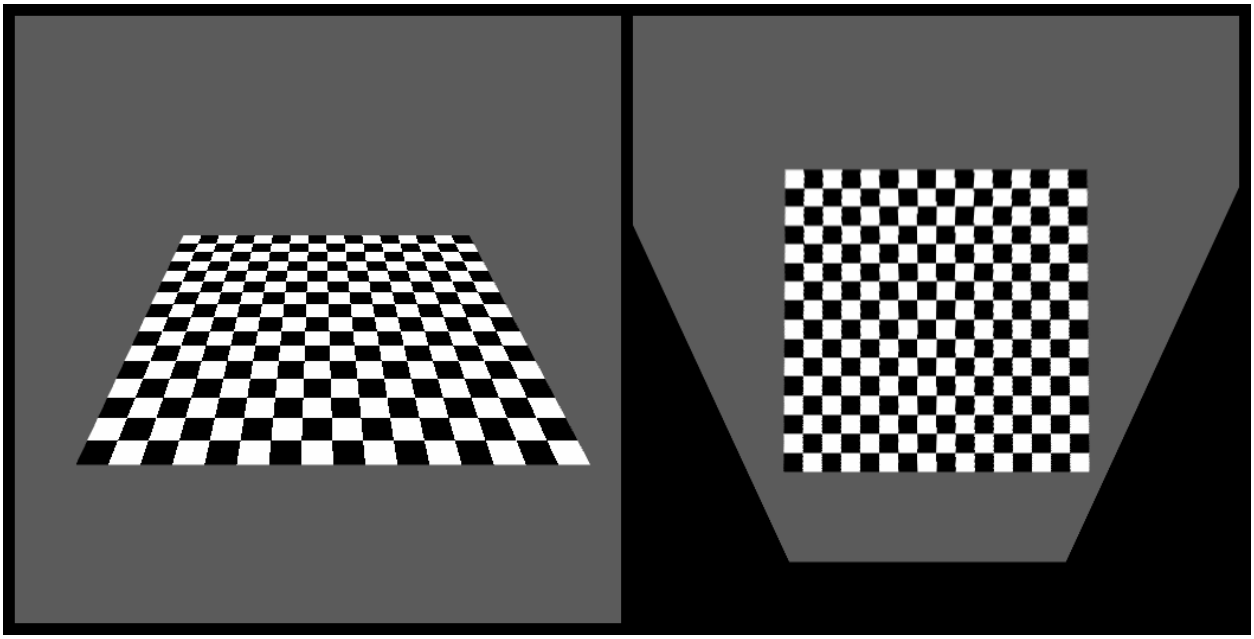


Figure 3. Inverse perspective mapping calibration. Original camera feed *(left)* and inverse perspective mapped image *(right)*

such as Google, IBM and Honda (OpenCV, 2019). 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 of CV algorithms.
- Allows a greater proportion of time to be spent on solving the problem as opposed to implementing known algorithms.
- Increase familiarity with industry level 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 III-B-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 additional highly optimised operations for array data manipulation. The decision to use Unity for the simulation engine, as discussed in section III-B-1, removed the major benefit of C++, being the potential for the simulation code to talk directly to OpenCV, 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 I-C with key elements of the simulation system discussed in the following subsections.

1. Simulation engine

Two courses of action were available for the implementation of the simulation engine; development of a full custom simulation engine or implement simulation logic within an existing game engine. 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 engines 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 and language (C#). Comfortable implementing desired features.
  - No C# interface with OpenCV therefore requirement for IPC 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 III-A-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 assessed as a significant technical challenge and is the single largest time budgeted feature in the initial simulation work. 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 options such as static memory buffers, shared memory and memory mapped files which would allow direct memory access to data. These options added another level of complexity and it was determined that a lower technical risk solution was to use TCP via the ZeroMQ middleware. ZeroMQ is a communications interface with implementations for both C# and Python and is well optimised for speed.

The use of this approach potentially results in slower than real time simulation processing which was mitigated 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 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 research 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 with four bytes for each (x, y) pixel coordinate, representing the Red, Blue, Green and Alpha image colour 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 (the Alpha channel) is redundant and is removed. This results 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.

3. Simulation time dilation

There is a requirement to implement dynamic time scaling in the simulation order to maintain time synchronisation with the external process. The simulation scales time based on the desired processing update rate and the real time that the IPC process and external data processing takes. This allows simulation complexity scaling without the danger of temporal desynchronisation between simulation and external processes.

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, sensor data is collated and sent to the external processing program via the IPC process outlined above and the simulation pauses. 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 was initially tested by sleeping the processing thread for several seconds each tick. Time dilation with IPC was also tested using an initial prototype of the simulation; two stills from this test from the Python process output video showing the raw feed and the canny edges is included as figure 4.

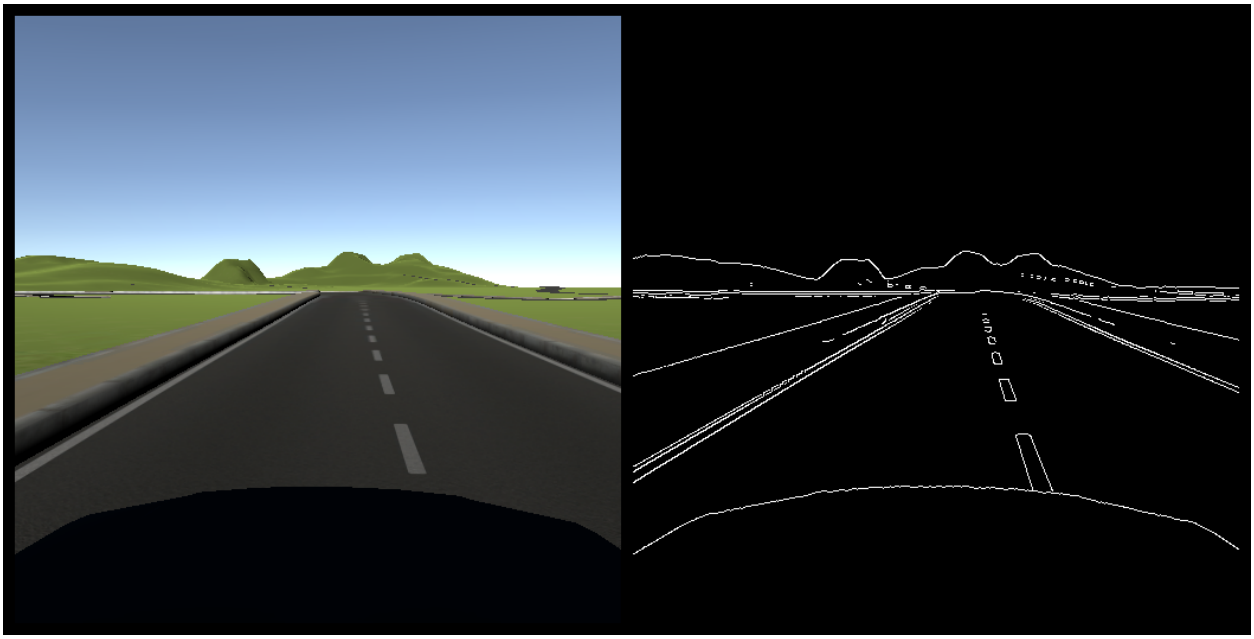


Figure 4. Still from video feed of simulation prototype test. Original camera feed *(left)* and canny edges of feed *(right)*

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. The Gantt chart included as appendix A outlines the initial planned approach however this will be amended based on the realised complexity of each task.

A. Computer Vision

1. Lane detection

Detection of straight lane lines has been completed in the familiarisation activities conducted as discussed in section III-A and demonstrated in figure 2. Alternative approaches to lane detection which may be considered include k-means clustering (Jun Wang et al., 2014), road classification from histogram based segmentation (Gonzalez and Ozguner, 2000), road surface mapping from near field driving surface models (Dahlkamp et al., 2006), matching road curvature models to detected lane boundaries (D. Crisman and E. Thorpe, 1991a) and open uniform B-spline model lane fitting (Jun Wang et al., 2014).

A more complex challenge is identifying a curved lane. Supervised convolutional neural network lane detection has had success through fully convolutional (Zang et al., 2018) and instance segmented (Neven et al., 2018) approaches. Spline based representation using random sample consensus for bezier splines based off road edge detection (Aly, 2008) has also shown to be effective. The current planned approach is lane edge detection via polynomial fitting of edge pixels detected by sliding window. An overview of the current implementation plan for the sliding window detection approach is included as appendix C. 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. The SCARF (Supervised Classification Applied to Road Following) approach of using Gaussian colour models to identify road surface before road model matching (Crisman and Thorpe, 1993) and UNSCARF (UnSupervised Classification Applied to Road Following) approach using feature identification, clustering and edge detection and road model matching (D. Crisman and E. Thorpe, 1991b) have both demonstrated an ability to detect lanes and intersections in non ideal conditions using a dual camera setup, although not in real time (Crisman and Thorpe, 1988). A data driven model using a 'virtual camera' supported by model based intersection matching has also shown promise (Jochem, 1995) however this requires prior information to reconstruct virtual images. Alternatively, once a binary road surface map is obtained, row based histograms can be used to identify intersection locations by interpolation between intersection models (D. Crisman and E. Thorpe, 1991a).

One example of effective intersection detection determined to be more directly applicable to this project is model based recognition which matches intersection models to a series of road boundary points (Todd R. Kushner, 1987). An alternate approach 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. This road skeleton can be matched to the model skeletons. The current intent is to use edge detection to determine boundary points with the intersection model being built reasonably simply using the OSM map data. 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.

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). The most likely solution will be a combination of the above and the model based recognition outlined by Todd R. Kushner (1987). 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 physics 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 from the navigation system discussed in section 4 will also be developed. It is important to note that this autopilot 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 by a simulated driver agent for analysis and should not be confused with the autonomy solution to the main problem this research is investigating. 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 commercial 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 single lane 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 customisable random position offset will be used to simulate GPS positional error however additional functionality improvements may include a higher fidelity simulation of GPS error. The technical risk of this aspect is negligible.

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. Waypoints will be dequeued from a distance threshold based on GPS position proximity to the next waypoint. 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 navigation network which can be queried using a path algorithm, most likely A*. 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 with the above functionality is delivered it is likely that improvements to the 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.
- Increased communication options between simulation and external process, for example vehicle control signals or specific sensor control/activation/deactivation.

V. Conclusions

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. 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. 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

B - A brief overview of a simple approach to lane detection

C - Sliding window detection for curved roads

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. Bertozz, A. Broggi, and A. Fascioli. Stereo inverse perspective mapping: Theory and applications. *Image and Vision Computing*, 16:585–590, 06 1998. doi: 10.1016/S0262-8856(97)00093-0.

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.

J. D. Crisman and C. E. Thorpe. Color vision for road following. In *Vision and Navigation: The CMU Navlab, C. Thorpe (Ed*, pages 9–24. Kluwer Academic Publishers, 1988.

J. D. Crisman and C. E. Thorpe. Scarf: a color vision system that tracks roads and intersections. *IEEE Transactions on Robotics and Automation*, 9(1):49–58, Feb 1993. ISSN 1042-296X. doi: 10.1109/70.210794.

J. D. Crisman and C. E. Thorpe. Unscarf–a color vision system for the detection of unstructured roads. volume 3, pages 2496 – 2501 vol.3, 05 1991a. doi: 10.1109/ROBOT.1991.132000.

J. D. Crisman and C. E. Thorpe. Unscarf–a color vision system for the detection of unstructured roads. volume 3, pages 2496 – 2501 vol.3, 05 1991b. doi: 10.1109/ROBOT.1991.132000.

H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: Science and Systems*, 2006.

R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15:11–15, 1972.

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

J. P. Gonzalez and U. Ozguner. Lane detection using histogram-based segmentation and decision trees. In *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.00TH8493)*, pages 346–351, Oct 2000. doi: 10.1109/ITSC.2000.881084.

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

J. Illingworth and J. Kittler. A survey of the hough transform. *Comput. Vision Graph. Image Process.*, 44 (1):87–116, Aug. 1988. ISSN 0734-189X. doi: 10.1016/S0734-189X(88)80033-1. URL `http://dx.doi.org/10.1016/S0734-189X(88)80033-1`.

T. Jochem. Initial results in vision based road and intersection detection and traversal. 1995.

Jun Wang, Tao Mei, Bin Kong, and Hu Wei. An approach of lane detection based on inverse perspective mapping. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 35–38, Oct 2014. doi: 10.1109/ITSC.2014.6957662.

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`.

H. Mallot, H. Blthoff, J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics*, 64:177–85, 02 1991. doi: 10.1007/BF00201978.

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, 2019. 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.

OSM. Osm file formats, Feb. 2019. URL `https://wiki.openstreetmap.org/wiki/OSM_file_formats`.

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. Tan, J. Dale, A. Anderson, and A. Johnston. Inverse perspective mapping and optic flow: A calibration method and a quantitative analysis. *Image Vision Comput.*, 24(2):153–165, Feb. 2006. ISSN 0262-8856. doi: 10.1016/j.imavis.2005.09.023. URL `http://dx.doi.org/10.1016/j.imavis.2005.09.023`.

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`.

S. Tuohy, D. O'Cualain, E. Jones, and M. Glavin. Distance determination for an automobile environment using inverse perspective mapping in opencv. In *IET Irish Signals and Systems Conference (ISSC 2010)*, pages 100–105, June 2010. doi: 10.1049/cp.2010.0495.

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.

W. Yang, B. Fang, and Y. Y. Tang. Fast and accurate vanishing point detection and its application in inverse perspective mapping of structured road. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5):755–766, May 2018. ISSN 2168-2216. doi: 10.1109/TSMC.2016.2616490.

Y.-t. Y. Y.-j. C. B. L. L.-s. Z. Yu Cao, Ying Feng. Monocular visual odometry based on inverse perspective mapping, 2011. URL `https://doi.org/10.1117/12.900010`.

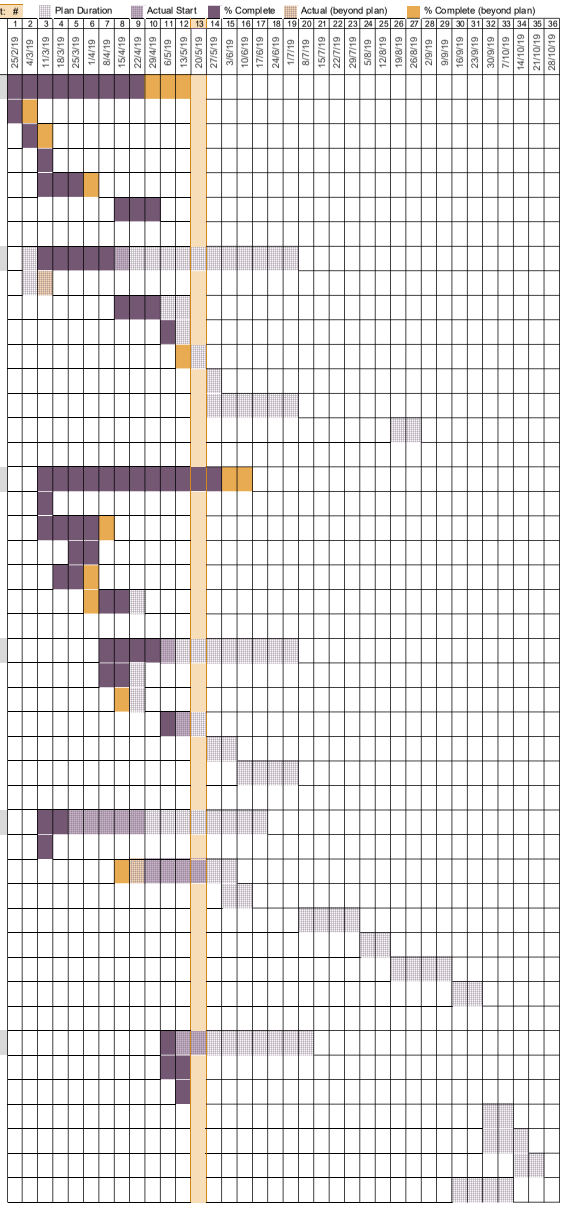
J. Zang, W. Zhou, G. Zhang, and Z. Duan. Traffic lane detection using fully convolutional neural network. pages 305–311, 11 2018. doi: 10.23919/APSIPA.2018.8659684.

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 **AS AT 24 MAY 19**

Period Highlight: #



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	100%
SIMULATION	**2**	**18**	**3**	**6**	**88%**
Engine comparison	2	1	3	1	0%
Inter Process communications reesarch and prototype	8	5	8	3	100%
Time dilation research and prototype	11	2	11	1	100%
Prototype	13	1	12	1	100%
Commence agile development	14	1			0%
Minimum Viable Product	14	6			0%
Release candidate polish	26	2			0%
COMPUTER VISION BASICS	**3**	**12**	**3**	**14**	**100%**
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	100%
Open CV famil	7	3	6	3	100%
ROAD DETECTION	**7**	**13**	**7**	**5**	**89%**
Background research	7	3	7	2	100%
Straight lane tests	9	1	8	1	100%
Curved lane option research	11	3	11	2	75%
Curved lane tests	14	2			0%
Intersection detection tests	16	4			0%
MAP MATCHING AND NAVIGATION LOCALISATION	**3**	**15**	**3**	**7**	**31%**
GPS data analysis	3	1	3	1	100%
Curve matching option research	10	6	8	6	30%
Intersection models tests	15	2			0%
Intersection detection matching to models	20	4			0%
Integration of Simulation data flow tests	24	2			0%
Refinement	26	4			0%
Consolodation of output visualisation products	30	2			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%

B. A brief overview of a simple approach to lane detection

A brief overview of a simple approach to lane detection

Michael McDonnell

1 Overview

This document provides an overview of a very simple approach to detecting straight lanes on a road surface. The approach will be outlined in general and basic theory of the supporting concepts will be covered. This is not intended as a standalone document to develop robust lane detection, merely a simple conceptual introduction for the interested reader.

1.1 General Approach

A basic approach to lane detection using a vehicle mounted camera is as follows:

- Isolate the area of interest in the frame as the image data to be operated on, I
- Convert the pixels in I from RGB to greyscale, I_g
- Identify edge pixels from I_g using an edge detection algorithm, resulting in a binary edge map, E
  - Optionally preprocess image data by low pass filtering (blur) prior
- Transform the edge pixels, E into Hough space H
- Identify a set of lines of interest l by thresholding H to an appropriate level
- For each lane/road edge, left and right:
  - From a subset of lines in l representing lines within an angle of interest for the lane edge (ie. limit the lines to the expected angle range), select the line with the highest value (vote count).

The end result of this approach, assuming the road lines are adequately marked, is a line either side of the lane representing the approximated lane lines. Note that the above approach did not include inverse perspective mapping. This is a technique that provides a 'birds eye view' of the image by transforming the pixels and is commonly used in lane detection however is not required in this simple case.

2 Isolation of image data I

Image data is stored as a two dimensional array of colours (usually represented by a three element vector for Red, Blue and Green channels). Isolation of the data in the region of interest is as simple as selecting a subrange of the array in the relevant area. A simple example of this is shown in figure 1 where the subset from (a, c) to (b, n) is defined as the region of interest. In this case the image data to be operated on, I would be a new array with the values (a, c) to (b, n). Conversion to greyscale is simply a weighted average of the channels, with different algorithm implementations weighting channels in different ratios.

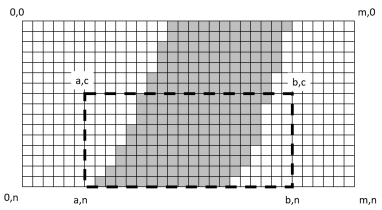


Figure 1: Subset of pixels from the full image array

3 Development of binary edge map E

The binary edge map is developed using an edge detection algorithm. There are many of these but one the simplest involves filtering with a Sobel operator. At the most basic level the Sobel operator is a 3x3 filter (or kernel) with positive coefficients on one side, negative coefficients on the other and zeroes in the middle. An example of a vertical line detection Sobel operator is below as equation 1. The binary edge map is created by taking the absolute value of the filtered result for each pixel and thresholding it over a certain value. For example if the absolute value of a filtered pixel is greater than 150, consider it an edge.

$$\begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & 2 \\ 1 & 0 & 1 \end{bmatrix} \tag{1}$$

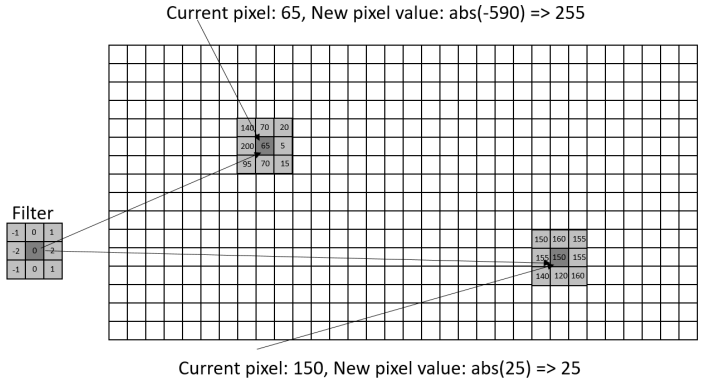


Figure 2: Example application of a filter to an arbitrary pixel

Applying a filter is similar to a form of 2D convolution. The filter is an odd dimensioned 2D array of coefficients. The central point of the filter is placed over the pixel being considered at (x, y) and the value of the new pixel, (x', y') is the summation of all the elementwise products of the relevant coefficients of the filter and the corresponding image area. This occurs for all pixels in the image; it can be considered that the filter is 'sliding' over the image pixel by pixel. As an example in figure 2, a Sobel filter to detect vertical lines is being applied to example pixels. The top left pixel is in a region of change and would be considered an edge pixel after thresholding, note the value of the pixel is capped at a ceiling of 255 in this instance. The lower right pixel is in an area of relatively constant values and after thresholding would be not considered an edge.

It is important to note that while the Sobel operator is a simple edge detection approach, this treatment has been at the most basic level. Horizontal and vertical kernels can be combined to develop a dual horizontal and vertical edge detector and a similar theory can be used to develop a detector sensitive to specific edge orientations. The Sobel operator approximates the gradient of the intensity thus the sign of the filtered output carries additional information that is not being considered here. Even so, one easy identifiable shortfall is the fact that regions of change over multiple pixels ('wide' edges) may produce multiple edge pixels after thresholding. The Canny Edge detector is an improved (albeit computationally slower) approach which includes non maximum suppression to develop a more refined edge map. These considerations are more complex and outside the scope of this brief overview.

4 Line detection using Hough Transform

For the Hough transform we need to consider a line not in the standard form of $y = mx + c$ but being represented by the orientation, θ, and length ρ of a line from the image origin to the line along the line's normal vector. An outline of this representation is included as figure 3. The Hough Transform maps cartesian coordinates x, y into Hough space θ, ρ coordinates.

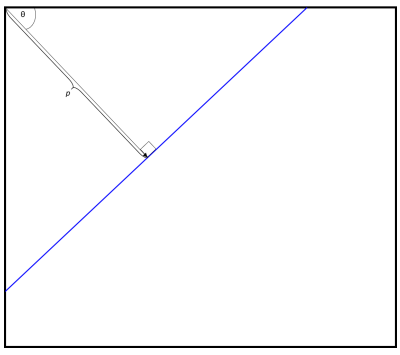


Figure 3: Representation of an image line from ρ and θ parameters

A single pixel in image space is represented as a single sinusoidal line in Hough space. This represents all the combinations of θ and ρ that would result in a line passing through the pixel location in image space. Each point in Hough space is then considered as a 'voting bin' where each pixel considered in the binary edge map 'votes' for lines it could belong to. In the case of a single pixel, there would be one 'vote' at each point in Hough space that the sinusoidal line corresponds to. This is demonstrated visually in figure 4.

If we then consider two pixels in image space, there is only a single possible line that could pass through both points. As per figure 4, each individual pixel is transformed into a single line in Hough space but as there are two lines, there is a point where they intersect. This is demonstrated visually in figure 5. Investigating the Hough space 'bins' would show multiple bins with a single vote (corresponding to the two sinusoidal Hough space lines) and one bin with two votes; occurring at the location where the Hough space lines cross as both pixels place a 'vote' at that location. This bin represents a peak in Hough space and corresponds to the θ and ρ value which would result in a line passing through the two pixels.

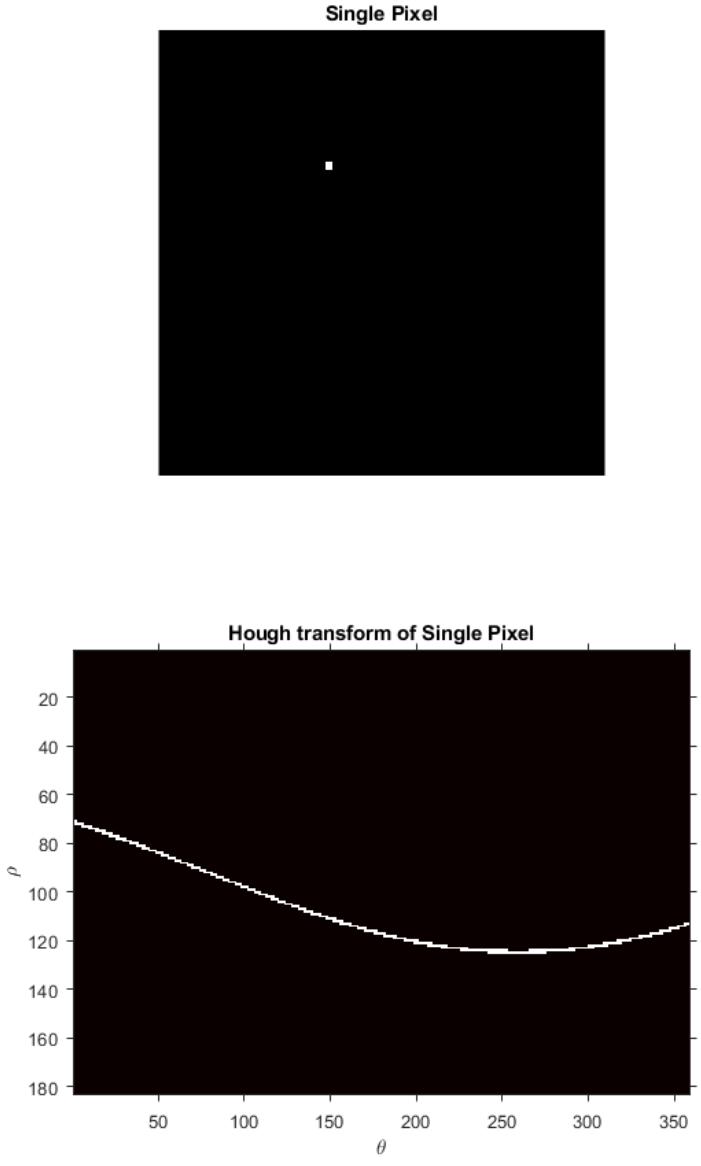


Figure 4: A single pixel in image space translating into a single sinusoidal line in Hough space

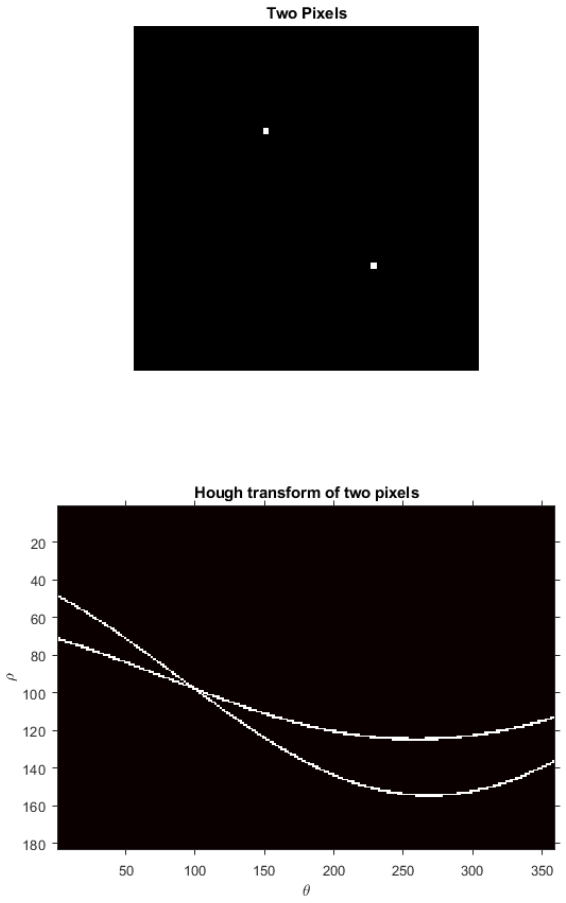


Figure 5: Two pixels in image space with intersecting Hough space lines

These examples are purely academic; the real power of the Hough transform comes where the input is a series of noisy edge pixels is provided as input. An indicative noisy edge map has been provided as input with the corresponding Hough transform in figure 6. In this figure the intensity of the lines in the Hough transform represents the number of votes in each bin. The blue asterisks represent the local peaks above a certain threshold which indicate the detected lines.

The detected Hough lines can be transformed back into image space using the respective θ and ρ values for the identified peaks. Figure 7 shows the raw output of the detected lines as the left image showing multiple lines in close proximity. This is unsurprising given the multiple peaks detected as per figure 6. Local non maximum suppression means of all the lines in close proximity, only the maximum is considered and all other 'close' lines are ignored. The output when this is applied is what we are after for our lane lines and is shown as the right hand image in figure 7.

As with everything in this document, this has only been a superficial introduction to the Hough transform. The Hough transform has multiple tunable parameters such as the quantised resolution of θ and ρ values and the thresholded vote count for a Hough peak to be considered a line. The intent of this document is merely for the reader to understand at a conceptual level how lane lines can be detected from the edge map so these aspects were not covered.

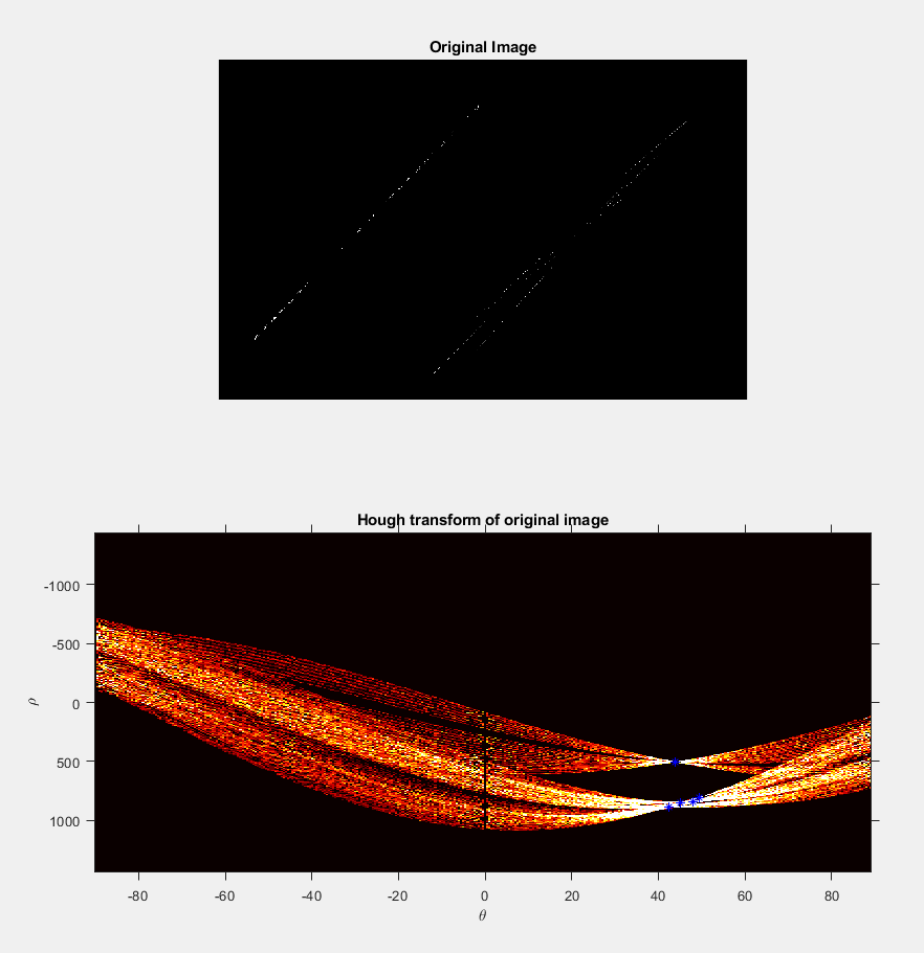


Figure 6: A representative binary edge map (top) and the relevant Hough transform (bottom)

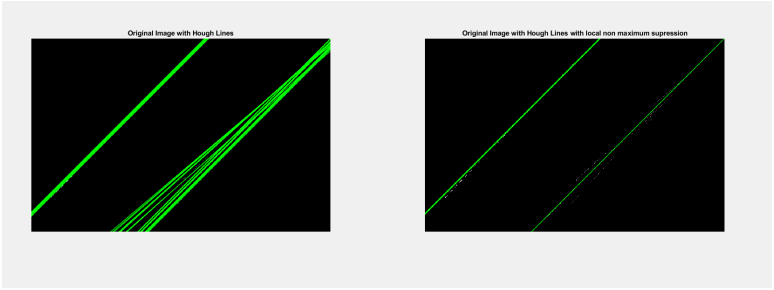
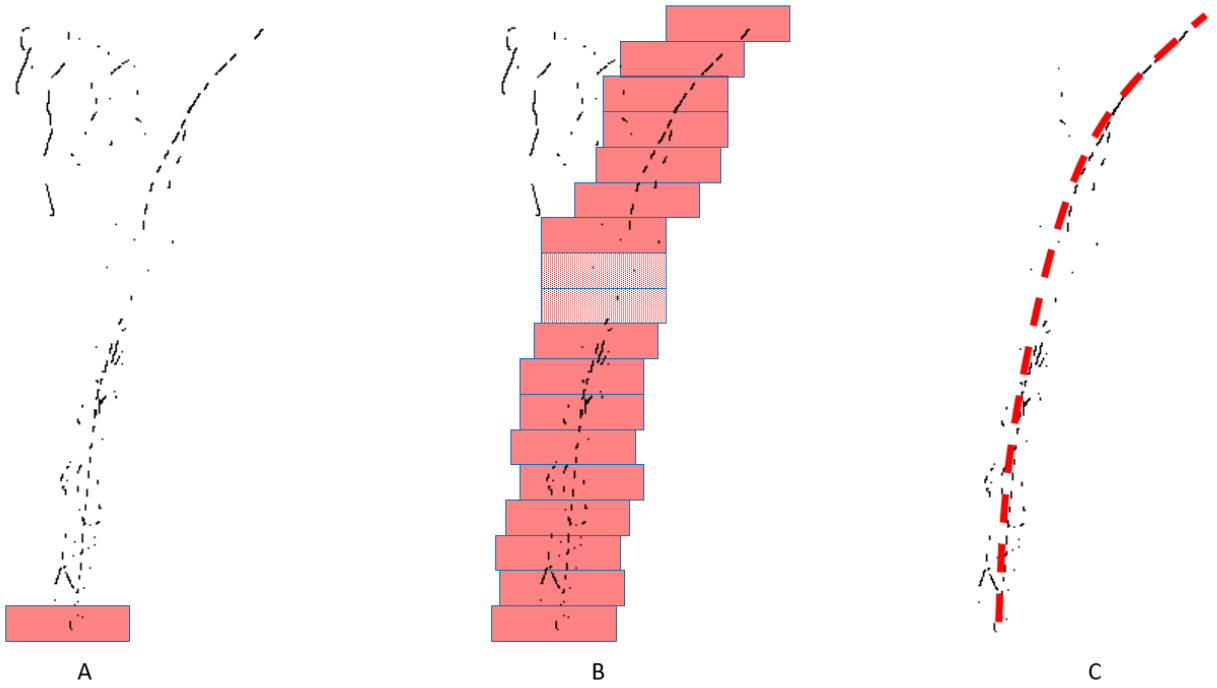


Figure 7: Binary edge map from figure 6 with all detected edges (left) and with local non maximum suppression of lines (right)

C. Sliding window detection for curved roads



After the road image has undergone inverse perspective transformation and thresholded edge detection, the start point for the sliding window is determined based off the peak locations of edge pixels at the bottom of the image. Image A shows the first sliding window in position over the detected pixels. The window is the local area of interest considered where edge pixels are counted and the horizontal location is averaged.

Each subsequent window is positioned at the horizontal average of the pixels in the preceding window as per image B. If the quantity of pixels present in a window is less than a minimum threshold, the previous window average is used, as per the two hatched windows in image B. The sliding window continues until the edge of the image is reached.

The pixels contained within the windows are then used as input for a polynomial regression to determine the curvature of the lane as per image C. Note that pixels outside the window are discarded for the regression calculation. This occurs for both sides of the lane independently.