Traffic Warning Sign Recognition & Vehicle Plant Model Integration

Wei Kit Wong
BSc (Hons) in Applied Computing
Waterford Institute of Technology
Waterford, Ireland
20075628@mail.wit.ie

***Abstract*—This electronic document outlines the implementation of a Traffic Warning Sign Recognition (TWSR) system that outputs an appropriate transmission response if a stop sign is detected, through the usage of a Vehicle Plant Model (VPM). This is a theoretical Advanced Driver Assistance System (ADAS) modelled in Simulink to show how Traffic Warning Sign Recognition may be used in a real-world setting in conjunction with other ADAS features.**

Keywords—detection, integration, MATLAB, recognition, Simulink, traffic sign, transmission

I. Introduction

TWSR systems are starting to become standard in the automotive industry, with many different implementations to produce the ADAS feature. It is considered as one of the "'most influential' innovation over the decade" in the context of vehicles [1]. One aspect of TWSR that is not researched thoroughly is its further application and how it can be used with other ADAS features. This project is an integration exercise of a TWSR and a VPM in Simulink, where if a stop sign is detected, an appropriate transmission response to stop the vehicle will be sent. To simulate the driving experience, a video input will be fed into the TWSR system and the output of the VPM's effect will be visualised in a graph.

II. Traffic Warning Sign Recognition

This heading looks at the functionalities of the TWSR system to be implemented for the integration exercise. A review of the expected scope will be performed, and an overview of the different components involved in its implementation will be provided.

A. Scope

As the implementation of TWSR is not the major focus of this project, the system can be a basic model that can identify the most common stop signs, with the stop sign being mandatory for this project, i.e. such warning signs include "Do Not Enter" and "Yield".

After researching through existing TWSR systems implemented in MATLAB and Simulink, the example one provided by MathWorks [2] proved to suit my needs the most as its implementation was well documented, therefore it was chosen as the base of the project as it could be easily tailored for integration.

B. Implementation

This looks at how the TWSR system was implemented from the chosen MathWorks example.

To begin, the entire pipeline is divided into three steps:

1. Detection
2. Tracking
3. Recognition

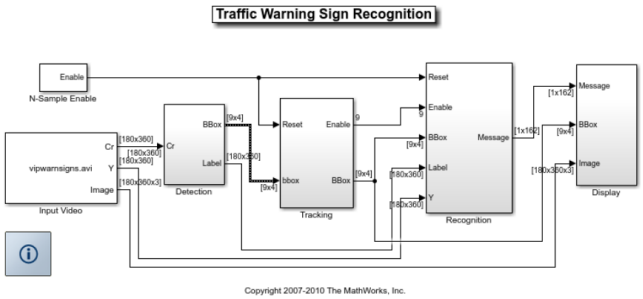


Figure 1. Simulink implementation of the entire TWSR pipeline

1) Detection

After an input video is provided for inspection, the first phase of TWSR is detection. For each frame, it looks at different portions of the picture and searches for pixels that match a potential traffic sign through Blob Analysis. It performs the matching through the usage of detection templates, which are a blob representation of each traffic sign to be detected.

Figure 2. Detection templates for each traffic warning sign.

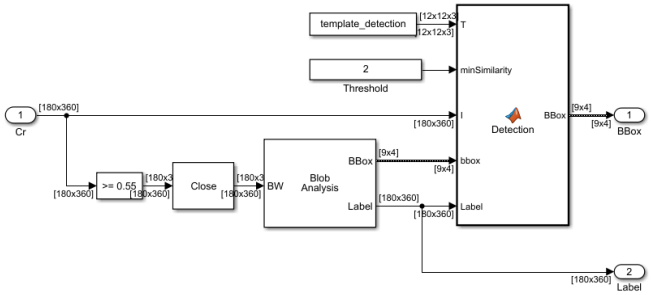


Figure 3. Simulink implementation of the Detection step of TWSR

2) Tracking

To add onto the detection process, tracking is the last step of the pipeline to ensure what is being inspected is a traffic warning sign. Its function is to ensure that the current potential traffic warning signs being inspected were also detected in the previous four frames. This is to ensure that the blobs detected were not caused by consecutive systemic errors or outside environmental factors. This is performed by comparing the blob matches and if the blob configurations are consistent for within four frames. If this

is the case, the potential traffic warning sign is considered an actual one.

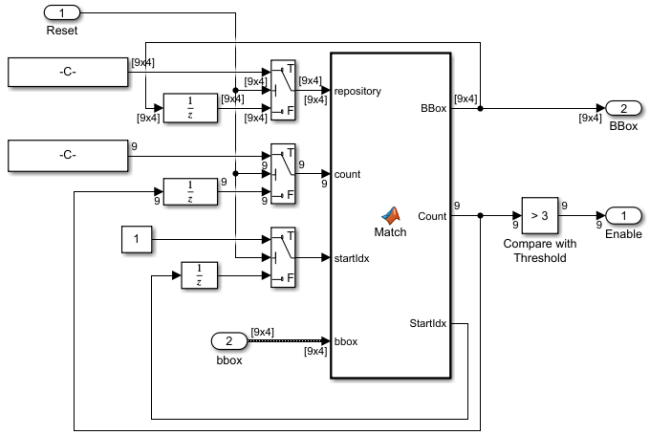


Figure 4. Simulink implementation of the Tracking step of TWSR

3) Recognition

The final step is to perform the actual recognition; to determine which traffic warning sign it is. This uses a similar process to Detection, but rather than using detection templates, recognition templates are used. These templates highlight the features of each traffic warning sign rather than the shape of it. By comparing each template with the sign's features, the warning sign is derived by choosing the one that has the most approximate features.

Figure 5. Recognition templates for each traffic warning sign.

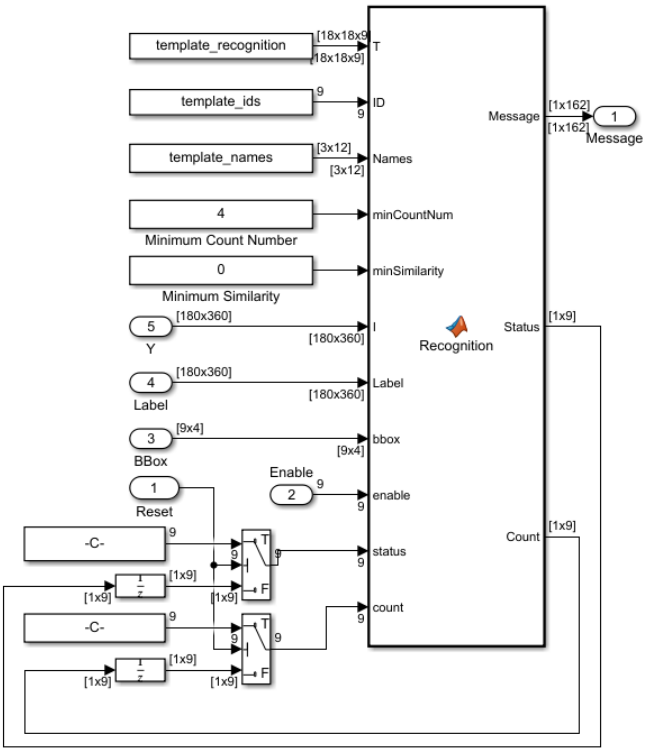


Figure 6. Simulink implementation of Recognition step of TWSR

As the purpose of the original MathWorks example was to just recognize each individual traffic warning sign in the video and highlight it, the recognition function also performs the labelling of what traffic warning sign it is. This will be used in the integration to figure out whether the detected sign is a stop sign or not.

III. Vehicle Plant Model

This looks at the scope of the VPM system to be implemented and used for the project.

Similarly to the TSWR system, a base model will be used rather than creating one from scratch. In this case, the "Vehicle Plant Model" from last year's Model Based Development class will be used, which simulates the action of a gearbox for a vehicle.

A. Scope

For the purposes of this project, the existing functionalities is satisfactory for this exercise. Currently, the system accepts an input matrix of throttle speeds in kmph that the vehicle will be going through over a time period, and the plant will perform the gearbox simulation, providing outputs of RPM, torque and velocity based on the fixed gear provided.

B. Implementation

This heading provides a brief overview at how the VPM system was implemented.

In addition to the throttle and gear inputs, an extra gradient value is provided, which indicates the flatness of the road that the vehicle is going through. To simulate the velocity output from these three inputs, the two forces are calculated;

1. Tractive
2. Damping

In addition to these forces, other features of the model will be looked at;

1. Aggregation of the two forces
2. PID Controller to smoothen the throttle input

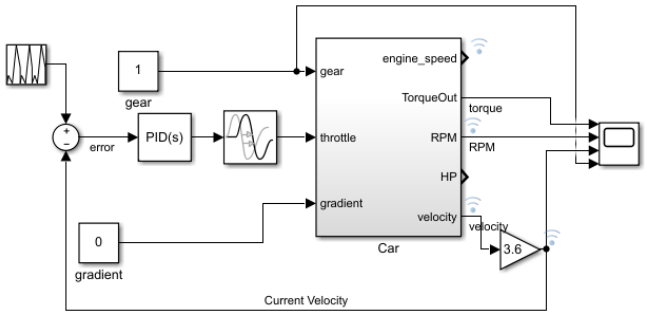


Figure 7. Simulink implementation of the entire VPM system

1) Tractive Force

The tractive force refers to the pushing ability of a vehicle and how much force is exerted based on the given throttle and gear.

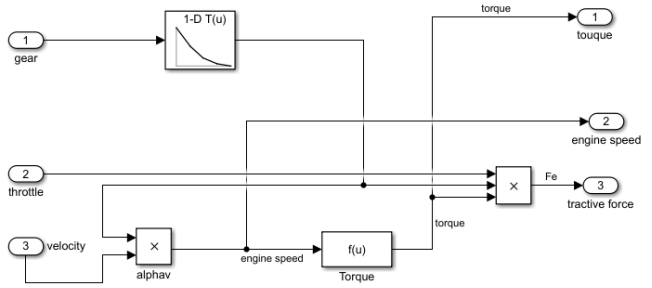


Figure 8. Simulink implementation of the Tractive Force in the VPM

2) Damping Force

This looks at the general effects of gravity and rolling resistances on a vehicle.

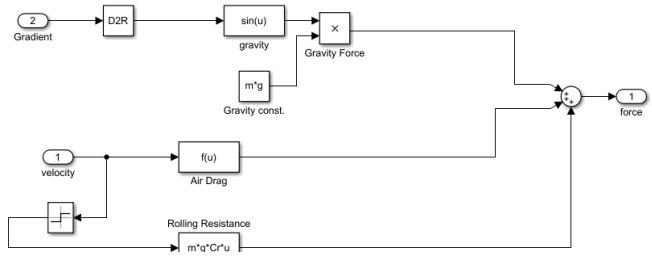


Figure 9. Simulink implementation of the Damping Force in the VPM

3) Aggregation

Once the two forces are calculated, they are applied together to output the resulting velocity of the vehicle for that second, based on the given throttle, gear and gradient input. Other variables can also be derived as shown, such as the RPM and resulting torque.

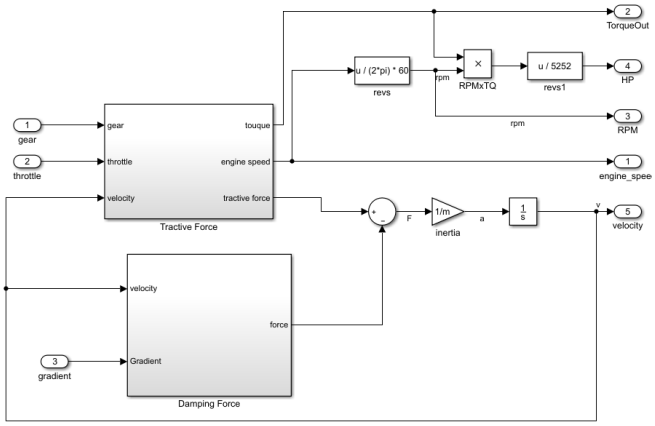


Figure 10. Simulink implementation of the entire Car in the VPM

4) PID Controller

As shown in Figure 7, a PID Controller is used to smoothen the throttle input for the main Car model in the VPM. The throttle input is compared with the output velocity.

This will be an essential part of the integration exercise as in order to provide a stop transmission response, the throttle input must be intercepted with this stop input. This PID Controller will help ensure the vehicle does not come to a sudden halt, but rather a smooth drop of speed.

IV. Integration between TWSR and VPM

This is the main workload of the project, to integrate the TSWR system with the VPM; by the recognition of a traffic stop sign, a transmission response to stop the vehicle should be outputted to the throttle.

As the TWSR and VPM components were pre-built, it was a matter of tailoring both models together and having them interact. This integration implementation can be divided into the following steps;

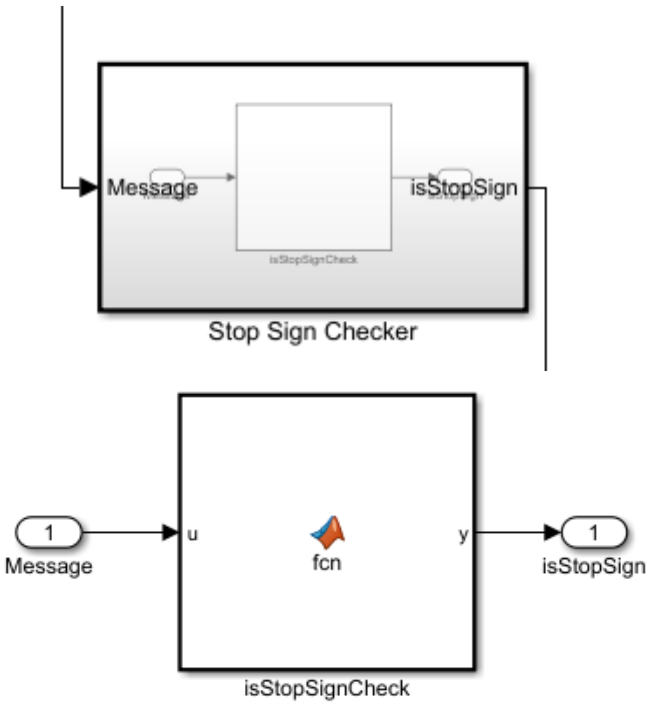
1. Stop sign checking
2. Stop transmission response

1) Stop Sign Checking

This step involves retrieving the recognized traffic sign from the TWSR system and ensuring it is a stop sign. This was a relatively simple operation, as the model already provides an output message variable which indicates the sign type. By checking this message, the sign type could be derived.

a) Stop Sign Checker Subsystem

A Simulink subsystem will be built to house the stop sign checking logic. This simply takes in the output message of the TWSR system and sends out a 0 or 1 signal, indicating whether the recognized sign is a stop sign or not.



Figures 10. Simulink implementation of the Stop Sign Checker Subsystem

b) MATLAB Function

This final workload involved in the stop sign checking was to write the MATLAB function to check if the traffic sign is a stop sign or not.

As the output message from the TWSR system stored its string data in integer matrixes, the “stop” string could be easily found through a match.

```
function y = fcn(u)
    % Gives a 0 or 1 based on whether a stop sign is detected or not
    % 83 116 111 112 => S t o p
    stop = [83 116 111 112]

    if any(ismember(u, stop))
        y = 1;
    else
        y = 0;
    end
```

Figure 11. MATLAB Function implementation of the Stop Sign Checker logic

If the match was found, then a 1 can be outputted, otherwise a 0.

2) Stop Transmission Response

With the recognition of the stop sign performed, the final step was to provide the stop transmission response via the VPM. By simply intercepting the existing throttle input from the vehicle with a stop signal, this can be performed. Rather than doing this in a separate module, this was done inside the VPM as the vehicle itself is handling the stop transmission.

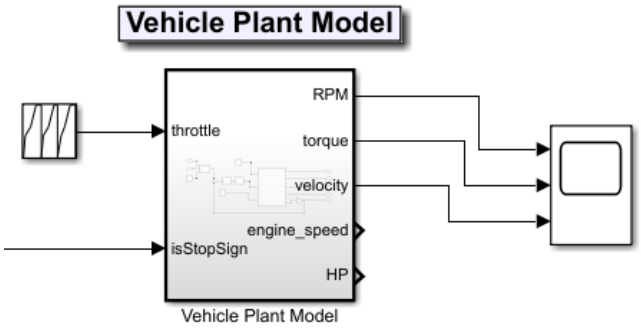
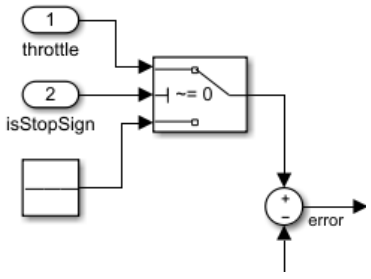


Figure 12. VPM as a subsystem now accepts both throttle and stop sign detection input.

As shown in Figure 12, the VPM now accepts an additional input of the stop sign checker output derived from the Stop Sign Checking heading. As this is a boolean signal of 0 or a 1, a switch block could be used to intercept the throttle input depending on the value of this input;

- If it's a 1, intercept the throttle input with a stop signal
- If it's a 0, keep using the throttle input



The switch block in Simulink accepts three input ports;

1. Main value
2. Value to check switch condition against
3. Alternative value

Figure 13. Simulink implementation of the stop transmission response

Note that the alternative value or stop transmission is simply a repeating sequence of 0s overtime, which overwrites the throttle input if a stop sign is detected.

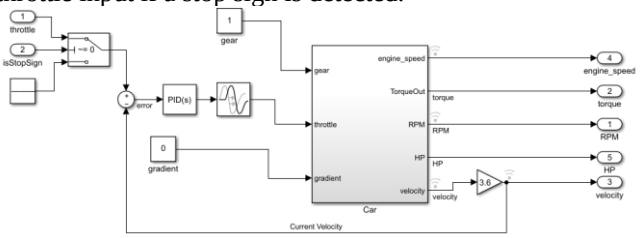


Figure 14. Simulink implementation of the stop transmission response inside the VPM

V. Simulink Model

Before performing the benchmark testing, this chapter looks at the final resulting Simulink model of the integration exercise.

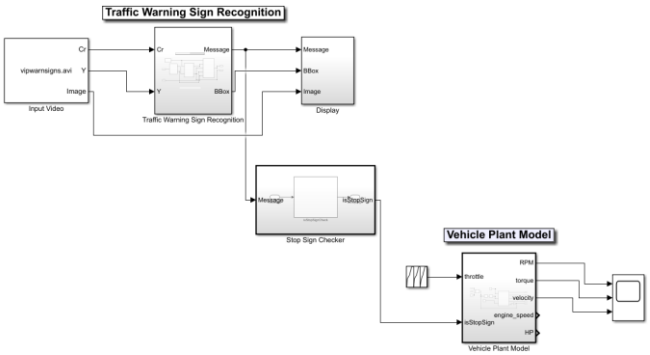


Figure 15. Entire Simulink model

A. Traffic Warning Sign Recognition

This was the TWSR system covered in the second major heading. Note that the original model was converted into a separate subsystem to modularise this component and the VPM.

Another aspect of the model that should be noted is the display module, which provides a live view of the input video and a highlight of the traffic signs being detected and recognized. This will be used in conjunction with the VPM visualisation tool to derive the performance of the integration exercise.

B. Stop Sign Checker

This was the main integration block that allows the interaction between the TWSR and VPM systems. As defined in the fourth major heading, this simply checks if the current recognized traffic sign is a stop sign.

C. Vehicle Plant Model

This final model is the VPM in the third heading, which allows the simulation of the stop transmission.

Note that in order to visualise and see the results of the entire integration Simulink model, its original Scope block will be inspected to view how the speed of the vehicle changes based on the viewing of a traffic sign in the TWSR system.

VI. Benchmark Testing

To view the results of the integration exercise, the Scope block from the VPM will be used.

The following benchmarks were defined before the implementation of this system. These scenarios will be checked against the resulting integration Simulink model to determine whether the desired objective was a success or a failure, which will be discussed in the Conclusion chapter.

A. Detection of a Traffic Stop Sign

Using the pre-built TWSR system, this final Simulink model passes this benchmark as it can correctly identify a traffic stop sign from an input video feed.

B. Provides a Stop Transmission Response

This involves a stop transmission being outputted to the VPM. This was a success as a method of intercepting the throttle input was implemented in conjunction with the stop sign checker module.

C. Provides a Smooth Stop Transmission Response

This benchmark requires more inspection to ensure it is being passed.

To validate this, the input video feed was configured to go through one traffic stop sign and then another at the end of a 5s drive. During these five seconds, the throttle is used to move the vehicle from standing to 40kmph. This is the Scope plot view of the resulting vehicle's velocity.

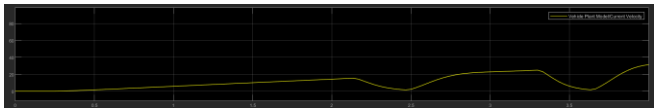


Figure 16. Graph of the vehicle's resulting velocity. Starting from standing to 40kmph within 5s.

As you can view from the graph, the stop sign can clearly be seen to be identified around 2.2s, and the final one at 3.3s. For each identification of the traffic stop signs, the vehicle's velocity drops in a smooth gradient rather an immediate drop to 0. Therefore, this tells us that the Simulink model provides a smooth stop transmission response.

D. Provides a Smooth Recovery Transmission Response

After the stop transmission response, an appropriate recovery transmission response should be provided after it, to ensure the vehicle continues moving in a desired speed but safely.

While this can be written as a success by inspecting the previous example, it would be better to test this functionality in a more high-risk situation, where the vehicle is driving in high speeds.

The scenario in this case is similar to the previous; an input video feed is configured to go through one traffic stop sign and then another at the end of a 5s drive. During these five seconds however, the throttle is used to move the vehicle from 60kmph to 80kmph. This is the Scope plot view of the resulting vehicle's velocity.

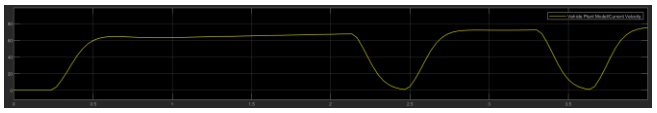


Figure 17. Graph of the vehicle's resulting velocity. Starting from 60kmph to 80kmph within 5s.

From the beginning, you can already see that the VPM has issues with simulating the vehicular speed from a certain desired velocity at the start. However, the main issue under inspection is the recovery process after a stop transmission.

Similarly to the previous scenario, both the stop and recovery response is smooth. But in reality, as the vehicle is moving at a consistent pace of 40kmph rather than 20kmph, the smooth recovery response is extremely dangerous due to the major shift in speed. This benchmark can be deemed a failure.

An ideal response would be to slowly build up the pace from rest to 40kmph over a 4-5s period, which would be much safer in execution.

E. No Stop Tranmission Response attempt from Rest

This basic benchmark ensures that no transmission response is caused from the detection of a stop sign nearby during rest. While this benchmark is unrealistic, as a vehicle should not detect multiple stop signs continuously from rest, this is a good indicator of whether the integration exercise is faulty or not.

The scenario to test this is simply a vehicle is on rest for 5s, but sees two traffic stop signs over this period.

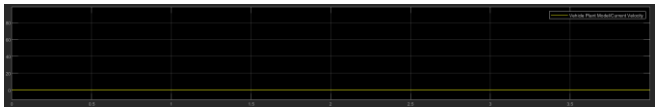


Figure 18. Graph of the vehicle's resulting velocity. Stays at rest for the 5s.

As we can see, no velocity activity was detected from the detection of a traffic stop sign. This can be deemed a success, and the integration exercise be fault-tolerant in this niche case.

VII. Conclusion

After looking at the performance of the project's resulting Simulink model, we can say that an appropriate stop transmission is given upon the recognition of a traffic stop sign. However, from the benchmark testing, we can see that it has a major safety-critical flaw in its stop transmission recovery state, where the vehicle attempts to quickly recover back to its original speed before the stop response, which could be a very dangerous operation depending on the speed of the vehicle. Therefore this ADAS feature can be deemed as a failure.

Despite this, the aim of the project was to showcase how a TWSR system can be integrated with another ADAS feature to extend its use case and functionality. While the resulting ADAS feature may be a failure, this was demonstrated sufficiently in that the TWSR can be extended and used in conjunction with others for better driving in the future.

REFERENCES

[1] Pilgrim, T. (2019). *Traffic sign recognition 'most influential' innovation of past decade*. [online] Phys.org. Available at: https://phys.org/news/2019-05-traffic-recognition-influential-decade.html [Accessed 31 Dec. 2019].

[2] Uk.mathworks.com. (2019). *Traffic Warning Sign Recognition*. [online] Available at: https://uk.mathworks.com/help/vision/examples/traffic-warning-sign-recognition.html [Accessed 2 Jan. 2020].