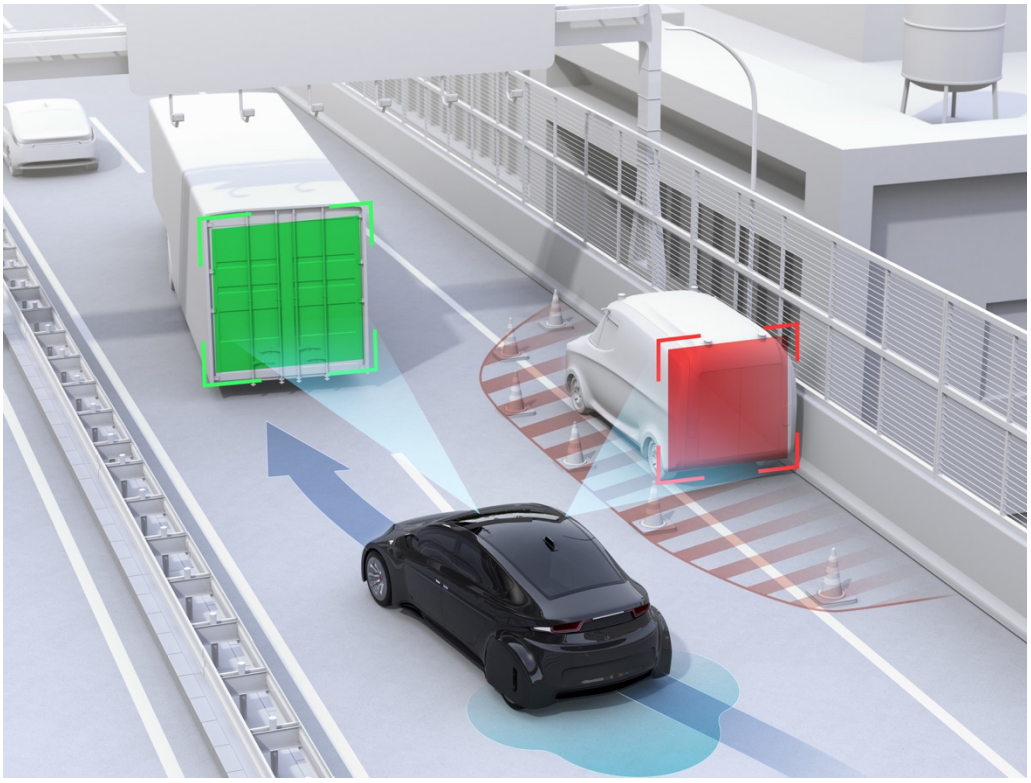
Safety and Security Analysis for Autonomous Vehicles
Technical Report V1.0



J. Robert Taylor (roberttayloritsa@gmail.com)
Jin Zhang (jin.zhang@ntnu.no)
Igor Kozine (igor.o.kozin@gmail.com)
Jingyue Li (jingyue.li@ntnu.no)

April 2021

THIS PAGE LEFT BLANK INTENTIONALLY

PREFACE

Our group has been working for a few years on the topic of safety of autonomous vehicles, including theoretical developments, actual risk analyses, hazard analysis methods effectiveness and quality studies, and neural network robustness studies. This report has been written to provide a framework for these detailed studies, and to avoid having to re-iterate the safety analysis background when each detailed study is published.

CONTENTS

THIS PAGE LEFT BLANK INTENTIONALLY

1. Autonomous vehicles and hazard identification

The development of autonomous vehicles is proceeding rapidly and promises safer and more efficient roads. However, safety and security problems remain, and the problem of drop out, that is the handover of vehicle control to a human driver presents a major problem (Banerjee et al., 2018). Hazard identification studies have been performed for most autonomous vehicles under development. Nevertheless, there remain problems of analysis methodology, which need to be solved before full confidence in autonomous vehicle controllers for mixed traffic can be obtained.
In *How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?*(Kalra & Paddock, 2016), Nidhi Kalra and Susan M. Paddock of Rand Corporation ask and answer this question. The answer is crucial for developing sound policies to govern their deployment. Their key findings are:

- Autonomous vehicles would have to be driven hundreds of millions of miles and sometimes hundreds of billions of miles to demonstrate their reliability in terms of fatalities and injuries.
- Under even aggressive testing assumptions, existing fleets would take tens and sometimes hundreds of years to drive these miles — an impossible proposition if the aim is to demonstrate their performance prior to releasing them on the roads for consumer use.
- Therefore, at least for fatalities and injuries, test-driving alone cannot provide sufficient evidence for demonstrating autonomous vehicle safety.
- Developers of this technology and third-party testers will need to develop innovative methods of demonstrating safety and reliability.
- Even with these methods, it may not be possible to establish with certainty the safety of autonomous vehicles. Uncertainty will remain.
- In parallel to developing new testing methods, it is imperative to develop adaptive regulations that are designed from the outset to evolve with the technology so that society can better harness the benefits and manage the risks of these rapidly evolving and potentially transformative technologies.

Kalra and Paddock assessed that to demonstrate with 95% confidence and 80% power that their failure rate is 20% better than the human driver failure rate would require 11 billion miles of on road driving, or about 500 billion vehicle years. This level of testing is impractical, and it is therefore desirable to analyze safety in the same way that other rare hazards are analyses, that is by risk analysis based on component reliabilities and by assessment of defense in depth. This does not mean that on-road testing would not be needed. The risk analyses would need to be validated since many hazard identification methods even at the best provide results which are far from complete for risk analyses (Taylor 2012). On-road testing is an evidence-based way of performing this validation. The risk analysis provides a way of amplifying the value of on-road testing, allowing near miss and partial failure cases to be included in the evidence base while providing a framework for assessment of such less serious incidents.

The motivation and approach describe in this report are similar in outline to that of Waymo's *Safety Methodologies and Safety Readiness Determinations* (*Waymo's Safety Methodologies and Safety Readiness Determinations*, 2020). Additional contributions in the project described here are the use of dynamic and hybrid fault tree methods, explicit reliability analysis of neural networks, the use of STPA style assessment of control loops but including emergent hazards (Taylor & Kozine, 2021a) as well as component functional failures, and the use of semi-automated fault tree construction to help obtain completeness and consistency in the fault tree analyses.

This report discusses the development of a range of methods to analyze autonomous vehicle safety and security. The approach is both theoretical and practical, in that proposed methods are tested using a development of an actual autonomous vehicle, actually a design for a ¼ scale vehicle.

One of the major problems in analyzing autonomous vehicle controllers is that of neural network (NN) components. These are notoriously difficult to analyze. A large part of this report is therefore devoted to discussing the difficulties of analysis of NNs in the context of autonomous vehicles.

2. Methodologies for hazard identification

2.1 A system theoretic approach to hazards and failure modes

As will be seen in later chapters, hazard analyses for autonomous vehicles can become very complex. Also, the methods in general use for hazards identification allow a good deal of flexibility in selection of descriptive terms and in the degree of analysis detail, and in the sequence in which hazards are considered. This means that it has traditionally been difficult to make analyses consistent and repeatable. This in turn means that it is difficult to determine whether a large analysis is complete. To overcome this, a first step is to provide an underlying model for analysis.

Any system may be described in terms of its behavior. Behavior is described in terms of system parameters which together describe system state, and the systematic changes in those parameters, depending on relations between external parameter values and the internal parameters, and the relations between the different internal parameters.

An event is a discrete change in a parameter or group of parameters taking place over a period of time which is short enough to be discounted when describing system behavior. “Tire rupture” is usually described as an event, although it is in some cases necessary to describe it as a process, for example when investigating the dynamics of the tire fragment paths and the secondary damage.

A condition is a specific set of parameter values or parameter ranges at a point in time, although that condition may endure over a period of time. For example, “speed is high, and visibility is low”.

A process is a change in parameter values or group of parameter values which takes place over a period of time which is of long duration when compared with other aspects of behavior, and sufficiently long to be significant in the description of behavior over time. For example, “corrosion” is a process which can lead to failure event of a vessel by “rupture due to wall strength falling below the stress due to internal pressure”.

Systems can be described in terms of interconnected sub-systems and sub-sub-systems, the interactions between connected subsystems etc. being describe in terms of the sub-system external parameters.

The behavior of a system can be described in terms of changes in any sub-system state, and the resulting changes in other sub-systems.

Failures are unwanted changes in system, sub-system, sub-sub-system state which may involve a change in a single parameter or a group of parameters.

A failure domain is a domain within a parameter space which is unwanted such as “brake is failed, speed is high”. A definition of failure which is equivalent to that above is that it is a transition into a failure domain.

A failure cause is a process or event within a system, sub-system etc. which leads either to another failure cause or leads to an actual failure event. An immediate or proximal failure cause is one which leads directly to a failure event.

An accident is a chain or coalition of failure causes which leads to harm. For example, “corrosion and pressure transient leads to leakage of hydrocarbon, and subsequent ignition leads to fire. This leads in turn to injury of exposed persons”.

A hazard, as in earlier definitions, is a condition which can lead to harm. A trigger is an event which activates the hazard, such as “technician spills acid”.

A failure mode of a subsystem is conveniently defined as a class of subsystem or sub-sub-system etc. failures such that the effect of any subsystem failure within the failure mode class on the rest of the system is identical to the effect of all failures into the class. For example, a failure mode for a reactor cooling system (i.e., a sub-system) could be failure of cooling water flow. Failure causes for this could be blockage of the flow piping, closure of a valve or stoppage of a pump. The definition of failure mode in this way is convenient because the failure modes of a subsystem can be defined in terms of the subsystem purpose, and can generally be tabulated as a logically complete set e.g., no function, spurious functioning, inadequate functioning etc. It also means that consequences or effects of a failure mode need to be evaluated only once for each failure mode, and for all of the failure causes within the failure mode set. This does not necessarily mean that failure modes are precisely defined in this way. For example, the consequences of “inadequate functioning” of the reactor cooling system may depend on the degree of inadequacy, such as “slightly below specification” or “significantly below specification”, with these terms defined according to consequences. However, in practice the failure modes defined in this way are usually selected appropriately by analysts. The approach as described here is almost always the one used in FMEA, irrespective of whatever definition is written in the standard or guideline used. It is also the core of the fault tree – event tree approach to safety analysis, for cause consequence analysis, and is the core also of the HAZOP approach, although in HAZOP the term “parameter deviation” is used instead of “failure mode”. The systems theoretic approach is also essential to automated FTA and FMEA, in that the models require to have a proper theoretic foundation. There are some so called “expert system” HAZOP approaches, but these never works well, with performance far poorer than manual HAZOP. In STPA, the failure modes are pre-defined.

Note that defining failure modes in terms of single failure causes rather defeats the object of FMEA (there is then no need for a failure mode column in the tables) and it leaves you with no guidance about the level of detail in the causal description. Should you just have a single line for “no flow though pipe” or one for “blockage”, one for “valve closed”, one for “deposits on pipe wall”, one for “foreign objects in the pipe”. You could of course do this, but in practice you would fill out the consequence column for the first cause in the list and then write “as above” for all the others, which is equivalent to the consequence class definition.

All of this can be expressed in set theoretic notation, ensuring precision, but we are leaving this out of this report since it is published elsewhere (Taylor 2017, Taylor and Kozine 2021).

2.2 FMEA/FFA, FTA, HAZOP CCA and STPA

There are several well-established methods for hazard identification in complex systems. The simplest is failure mode and effects analysis. FMEA has traditionally been used for systems reliability analysis, but the version described here is focused on safety analysis.

The FMEA approach starts with dividing a system into components. Then, selecting each component in turn, failure modes are identified for each. Having selected a failure mode, the potential causes for

the failure mode, and the effect of the failure mode on the system as a whole, are tabulated. For the failure modes, causes and consequences, the safety measures, in terms of detection, hazardous event prevention and hazardous event mitigation are tabulated.

For complex systems, it can be difficult to describe consequences in detail, especially when there are many safety measures. The cause consequence analysis approach allows a systematic and reproducible evaluation of consequences. The method involves tracing the sequence of events, beginning with the selected failure mode, through each of the affected system components. Where one event at the input to a component causes two or more events within or at the output of the component, the event chain branches (parallel branching). Where one input event to the component can have two or more effects within the component or at its outputs, the event chain branches (alternative branching). When there are delays in the even sequences, these are recorded.

Since the analyses to be performed for the autonomous vehicle are largely based on functional block diagrams, the failure modes assumed are the functional failure modes.

Functional failure analysis is often carried out with just a very simple failure mode in mind, “does not function”. Reliability block diagrams are based on this failure mode.

Functional failure analysis can be made more inclusive, for example with failures such as:

- Function fails on demand.
- Function fails while system is operating.
- Function fails partially (some part of the function is not performed).
- Reduced functioning (the function does not perform at the specified level).
- Over-functioning (the function performs above the specified level, possibly causing an overload somewhere)
- Delayed functioning.
- Premature functioning
- Intermittent functioning
- Cumulative performance deviations leading to malfunctioning
- Unwanted repeated functioning
- Erroneous function sequence
- Function omitted then remembered
- Correct functioning but without checking preconditions
- Correct functioning with latent hazards within the component

Fault tree analysis is a method which starts with an unwanted event and follows the possible causal paths backwards through the system. As each component is reached, the failure mode leading to the consequence event chain is identified, and then the internal failure mechanisms internal to the component are identified. Then the events at the inputs to the component which could cause the failure mode are identified. The search proceeds to the next component upstream.

The fault tree analysis method approach used here involves an initial selection of a TOP event in the form of an unwanted accident or incident. For the present purpose, these are “vehicle crash” and “drop out” of the autonomous vehicle control.

The second stage for the fault tree analysis is to localize the proximate causes of the crash. Since any crash involves movement of the vehicle, the localization involves steering and speed control, which can be localized to the wheels.

The causal events are then traced backwards from the wheels, through the steering and speed controls (brakes and engine) to the controllers and from there to sensors on which control actions are based. Hazard and operability analysis is a method in which possible deviations from normal or intended function of components are selected and causes and consequences of deviations are hen traced. With the extended list of functional failure modes given above, there are no differences between FMEA and HAZOP.

STPA (Leveson & Thomas, 2018) is a method in which a system is regarded as a hierarchy, or multiple hierarchies of control loops. The functional failure modes analyzed in the handbook are:

- Failure to provide a function when requires
- Providing a function when not required in such a way to cause a loss
- Performing a function too early, too late, or out of sequence
- Providing a function too long (too long duration) or stopped too soon (too short duration)

This list can be seen to be a subset of the functional failure modes listed above, and being a subset, will allow will require less effort than a full functional FMEA, at the risk of overlooking some kinds of accident scenarios.

STPA also introduces the concept of constraints:
A system-level constraint specifies system conditions or behaviors that need to be satisfied to prevent hazards (and ultimately prevent losses), which is similar but not identical to the concept of safeguards. The failure mode analysis is applied to the sensor or monitoring parts of a control loop, the control parts and the consequences. In principle functional FMEA should identify all of the accident scenarios that STPA does, but STPA, by focusing on control loops also ensures that the problems are regarded as those of control and works with control failure terminology.

An extension to STPA is to include "emergent" failure types, defined in this context as failures which are associated with control loops as a whole rather than with individual components (Taylor & Kozin, 2021a). An example is the presence of two control loops with competing actions. In such cases system failure can occur in the absence of component failure.
A list of such emergent failures is:

- Oscillation, hunting and resonance, surging
- Overshoot or undershoot and control inaccuracy
- Unwanted change of control mode, phase change in the controlled system
- Lag, hysteresis, backlash, stiction
- Intermittency, slow response
- Drift in parameter values
- Saturation and reaching the limits of control rangeability
- Wind up, bump transfer
- Chaotic behaviour
- Changes in performance and control parameters
- Poor turn down capability and instabilities arising when system throughput is reduced.
- Competing control loops
- Adverse overrides

Traditional STPA in itself does not add anything to the fault tree analysis of physical hardware and direct control because is covers only functional failure of the control loop components. These are included as a matter of course in the fault tree analysis process and in CHAZOP. The benefit of STPA

is in extending analysis to multiple control levels, including supervisory control and to management levels. Here the advantage is in providing analysis of the steering and speed control functions, the individual navigation algorithms, The navigation coordination and the journey goal setting The STPA focus on describing the system in a control loop oriented form allows emergent failure modes to be identified, using the approach in Taylor and Kozine (2021).

2.3 Cause Consequence analysis

Cause consequence analysis is a useful method for describing the sequence of events in an accident scenario, and describing the range of possible (e.g. counterfactual) variants in the scenario. An example is given in Figure 2.1.

1 >> blowout detected
- Blowout signal communicated to the navigation controller
-- Tire burst response for the wheel selected [selection assumed to be made correctly]
--- ~ Steerability evaluated {steerability = f(AV speed, steering angle)}
---- ? AV is in the steerable domain
----- Y Vehicle is controllable
------ ? steering system works correctly
------- Y vehicle under control >> 4
------ N No steerability, probable crash
----- N No steerability, probable crash

Figure 2.1 The cause consequence diagram for blowout detection response (textual form)

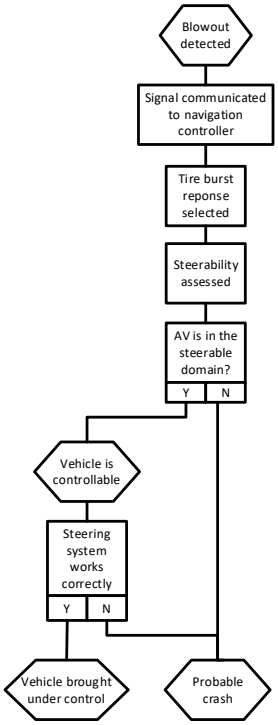


Figure 2.2a The cause consequence diagram for blowout detection response (graphic form)

The method begins with the selection of a critical event, such as the initial event in an accident sequence, or the start of activation (the trigger) of a safety system. The method then proceeds by listing the sequence of events in scenario. Any delays or significant time intervals are recorded by means of a delay symbol. If there are multiple event sequences in the scenario, such as the accident sequence itself and the parallel activities in emergency response, the diagram of the accident shows branching event sequences

Once the basic sequence of events has been described, alternative sequences which could have happened, for example if a safety system had failed or a latent hazard had been activated, are shown by means of a Yes/No box in the cause consequence diagram. For example, if a key event is an AV departure from the inner lane and crossing into an opposite lane, two following event sequences exist depending on whether there is oncoming traffic.

The cause consequence diagrams can be used as a basis for calculating event sequences.

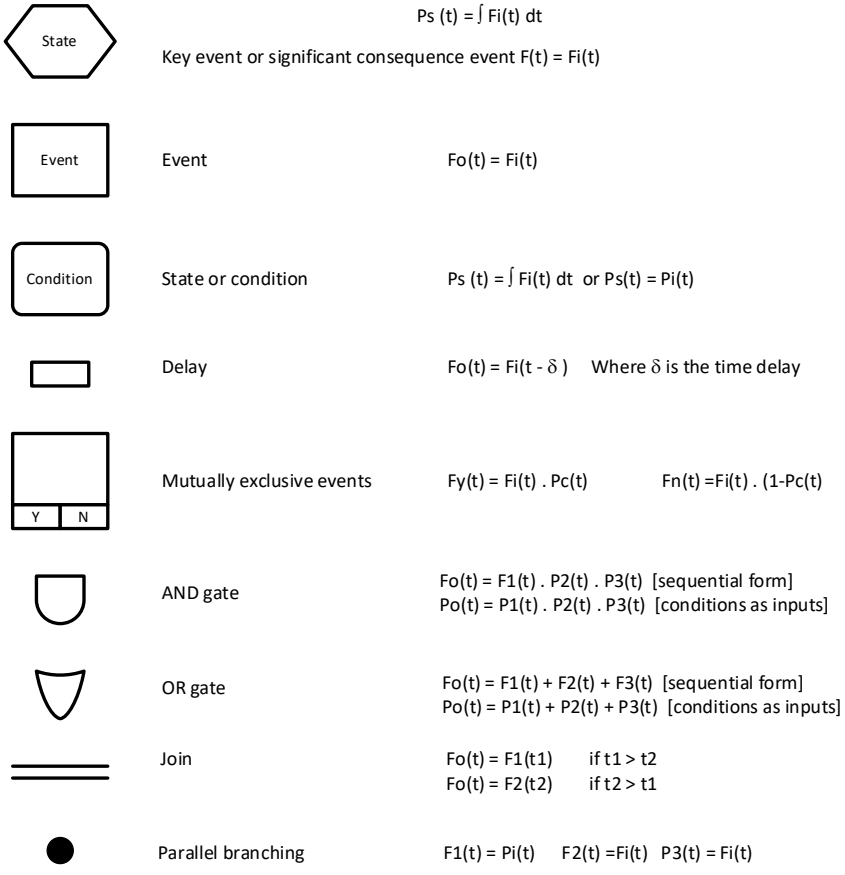


Figure 2.2 Cause consequence diagram tree symbols

2.4 Software FTA

The fault tree analysis method can be applied to software. A TOP event for a software fault tree is production of an unwanted or hazardous output, or the absence of a necessary output.
Chains of "events" within the software are then sought by tracing changes in pro-gram variables backwards from statement to statement, until program inputs are found which are necessary for the particular event chain. (The chains can then be extended once again to hardware, seeking the potential causes of the program input events found.) Alternative paths are recorded on the fault tree using OR gates. The conditions which allow a particular path to be followed are recorded using AND gates.

Software fault tree analysis can be carried out in different styles. In the simple case, flow paths are traced with colored pencils on paper copies of the software, either flow charts or source text, and paper check lists are used to remind the analyst of possible errors and failure effects at each stage. Software tools can sup-port this simple tracing, in which the intelligence is provided by the analyst. Programs can perform rapid look up of calls to programs, can help maintain a record of alternative branch points and parts of the analysis which are incomplete, and can record the results.

A much more sophisticated approach is possible, in which support tools record the logical conditions for flow along a control paths and can keep a continuous record of whether a path is logically possible by recording path predicates. Path predicates are the logical conditions which must be true in order for a particular path through a program to be followed. In this case, the programs become equivalent to a proof of correctness (proof of safety) for example using Hoare axioms (Ch 9, ref. 2) or Dijkstra weakest precondition theory (Ch. 9 ref. 6,7). The mini fault tree notation can be applied to individual program statements, allowing a uniform process of hardware/software fault tree construction. For each statement a set of mini fault trees is generated according to the schemes shown in Figure 2.. These are a modified form of weakest precondition predicate transformers used elsewhere in proving program correctness.

The "events" in the program mini fault trees are statements of the form.
"at time t, predicate P becomes true of the program variables".

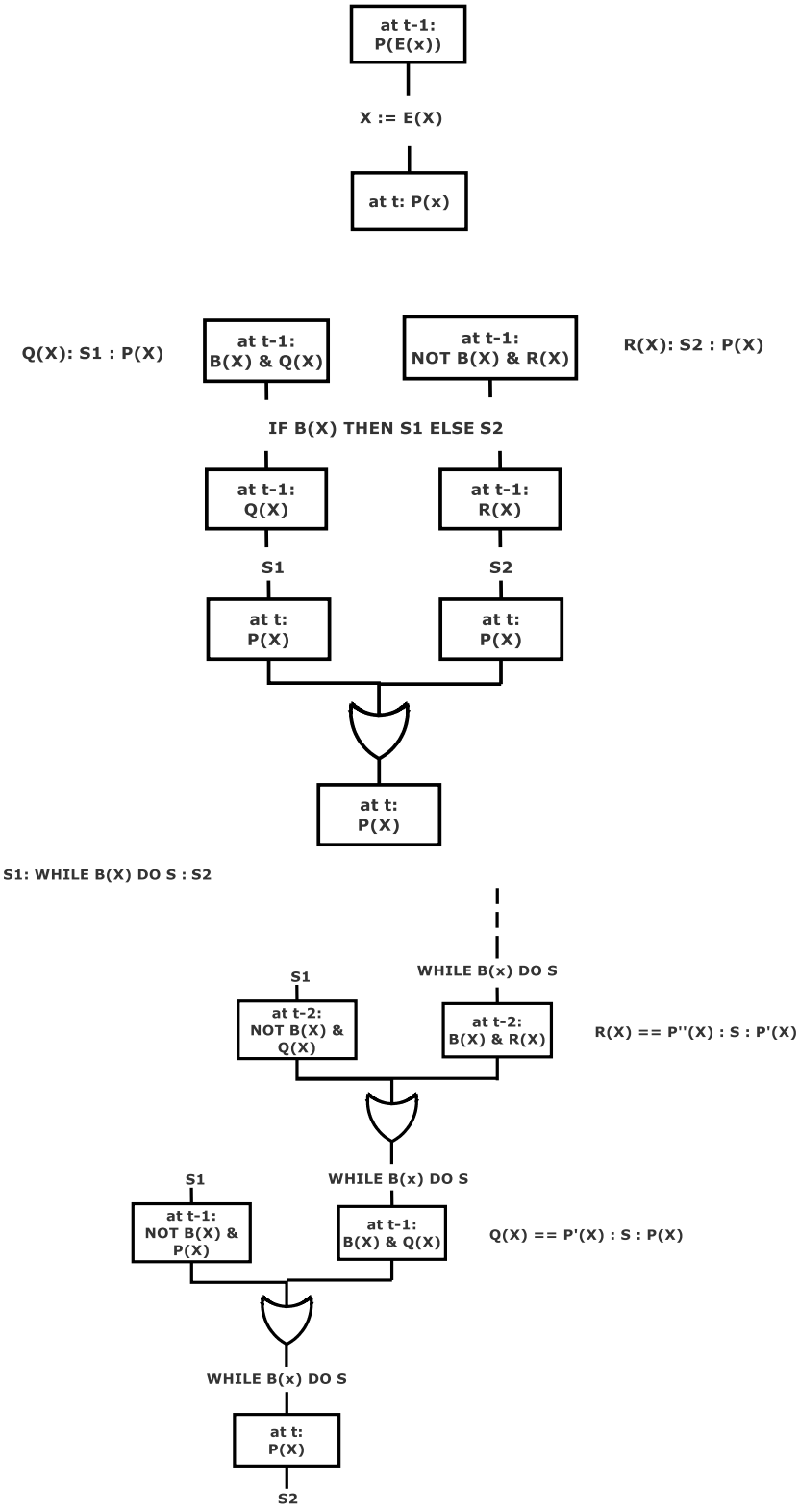


Figure 2.3 Mini fault trees for basic program statements

There are three main differences in the process of building fault trees for hard-ware and software. Firstly, instead of there being an exact match between event phrases and mini fault trees, as for hardware, the software mini fault trees may match a predicate representing variable values. Secondly, the mini fault trees, when matched, must be modified by substituting the output predicate into the mini fault tree, both at the input and output position. Thirdly, it is generally necessary to simplify the predicates rep-resenting input and output events in the mini-fault trees. If this simplification results in the predicate FALSE, the event chain is abandoned.

Note that the form of fault tree for IF and WHILE statements provide an OR branch for each direction of branching through the statement. This ensures that all paths through a program will be treated as different branches of the fault tree. This has the advantage of ensuring greater under standability, and of keeping individual predicates relatively simple. It has the disadvantage of producing very large trees in many cases. For this reason, it has been found advisable to insist on quite severe structuring rules for programs to be used with this technique. Separate (possibly parallel) programs should be provided for each separate control or safety function within a computer system. This rule ensures at least that the program paths followed are relevant to the analysis safety problem. An alternative formulation of the structuring criterion can be applied systematically. - If there are two program outputs X and Y for which the values are functions of sets of inputs I(X) and I(Y), then if $I(X) \cap I(Y)$ is empty, programs producing outputs X and Y should be disjoint.

2.5 The need for dynamic analysis

With the exception of cause consequence analysis, the traditional methods of hazard identification, FMEA, FTA and HAZOP, are highly static, that is they do not take timing or the sequence of events into account. Dynamic problems are important in vehicle accident scenarios in general, and in autonomous vehicle accidents in particular. Examples are:

- The time taken for hazardous situation identification versus the time needed to avoid the hazard or to take emergency action.
- The time taken to switch navigation modes in response to external conditions (primarily dependent on detection times since information processing is rapid).
- Priority of navigation strategies and timing of strategy change.
- Timing of braking or acceleration versus timing of steering maneuvers.

Event sequence can be important in situations such as:

- Clutch engagement/disengagement versus gear change
- Navigation segment changes versus external conditions

Sequential fault trees have been used based on priority AND gated, in which one event must occur before others. Taylor (Taylor, 1975) introduced sequential fault trees with a semantics based on system theory, in which all AND gates are sequential, with just one input being an event i.e. a change in state, and other inputs being condition i.e. sets of states(see Figure 2.). Such fault trees can be built up via simulation or reverse simulation and can incorporate time delays.

2.6 Need for hybrid analysis

The standard methods of hazard identification have a weakness in that they deal only with discrete failure events, such as brake failure or failure to detect an obstacle on the roadway. In reality many subsystems can suffer from degrees of failure such as reduced sensitivity of object detection in low lighting conditions and with low object/background contrast. Lin et al. introduced hybrid fault trees

using fuzzy parameter classes (Lin & Wang, 1997). Hybrid fault trees involving continuous variables were introduced by Taylor and Kozine (Taylor & Kozin, 2021b) to deal with this kind of problem in a way that uses continuous physical and behavioral models directly.

As an example of this, consider braking, whether braking will lead to a loss of vehicle control depend on the vehicle speed, tire and road conditions and on the extent of steering curvature. Most of these variables are continuous. It is possible to write the physical equations for the forces involved at the tires, and behavioral equations for the point at which tire grip will be lost. These equations define a multidimensional space in which there are different domains, in which braking performance is either successful or leads to loss of control. The degree of loss of control also affects whether there is any recovery strategy.

Continuum hybrid fault trees introduce continuous AND gates. These have an output which is an adverse parameter value or value domain. The inputs to the continuous AND gate are parameters of the performance/ failure equation. As an example, consider the braking force/vehicle speed/wheel lock up equation. At low speeds or low braking force braking will be successful. At high speeds with moderate braking force, brake lock up can occur, as it also can at moderate speeds and high braking force. Braking failure in the lock up mode can arise due to departure of the either speed or braking force alone from the intended design domain, or it can arise from smaller deviations of both variables from design intent values.

Hybrid analyses are particularly important for vision algorithms and neural networks, since inputs to these steering components can be dependent on external conditions such as the amount of background noise entering cameras, lidars and radars, and the response at different distances.

2.7 Completeness of analyses

The main objective of this study is to develop and validate an approach which can provide an approach to hazard identification which can be trusted as a supplement to testing and can be used to support safe design. For This purpose, analyses must be as complete as possible, and must be more powerful than on-road testing in predicting possible accidents.
A logical measure of completeness is:

$$Absolute\ completeness = \frac{Number\ of\ hazards\ identified}{Number\ of\ hazards\ existing}$$

The problem with this measure is that $Number\ of\ hazards\ existing$ cannot be known – there will always be some further accident types which were not predicted, or even predictable. An alternative measure is:

$$Historical\ completeness = \frac{Number\ of\ hazards\ identified\ by\ analysis}{Number\ of\ hazards\ recorded\ in\ an\ extended\ \ record\ of\ accidents}$$

The extended record of accidents in this measure could be all the accidents reported to authorities and collected in a database, but this is often a rather limited set. The historical basis can be improved by including near miss events. Many more possibilities can be obtained through by recording all the credible scenarios and accident phenomena in earlier analyses. The automated analysis support tools described in the next section automatically register all new hazards and accident phenomena from any analysis and allow moderated updating of a hazards database.

Completeness levels vary depending on the degree of detail in the analysis. Generally, the more detailed an analysis, the more difficult it is to obtain a high level of completeness. However, the value of an analysis to a designer increases strongly as the analysis becomes more detailed.

A very high level of completeness could be obtained by regarding the hazards database as a check list and just including all the recorded hazards in any new analysis. The analysis would not be useful however because it would include many irrelevant accident scenarios and designers would not be able to trust the analysis as a design basis. The discrimination of an analysis is defined as:

$$Discrimination = \frac{Number\ of\ actually\ physically\ possible\ hazards}{Number\ of\ hazards\ identified\ in\ the\ analysis}$$

A good analysis will have both a high level of completeness and a high level of discrimination. Several methods are used in this study because it has been observed that all currently available hazard identification methods have gaps, and it is rare that any one method achieves better than 50% completeness.

2.8 Manual and automated analysis

As will be seen below in Ch. 4, the analyses for an autonomous vehicle can be very large. It is difficult, during manual analysis, to retain consistency in terminology and in the level of detail, especially in fault tree analysis, and to ensure completeness of analyses. Automated analysis allows the use of generic models for individual components of a system, and then to recall these generic models, find a failure mode for a specific component and chain the effects of these from component to component through the system. At each stage the conditions which facilitate the propagation of the consequence event chains can be recorded. Similar tracing can be made backwards in time from an accident event such as a crash to the different failures in in the vehicle or the control system which can cause the accident event.

Making the models for complex control systems is difficult, since each functional block can have a wide range of function and performance. An initial manual analysis is preferable, in order to establish the kinds of functions, failures and errors which are involved. Generic models can then be developed on the basis of the manual analysis. The automated analysis then ensures that the ideas from the manual analysis are applied consistently. The automated analysis also enables analyses of modified designs to be completed very rapidly.

Fully automated analyses do not provide good discrimination. The reason for this is that in order to avoid incompleteness in the analyses, the generic models need to record all of the theoretically possible failure causes, even those that are very unlikely to occur in practice or this which are prevented by inherent safety measures in design. Semi-automated analysis allows the computer program to propose accident types and failure causes, and the human analyst to either accept or reject the proposals. The HAZEX program allows this, and where failure possibilities are rejected, records the rejection and allows the analyst to record the reason for the rejection, and any assumptions made in the analysis refinement.

2.9 Fault tree and cause consequence diagram compact notation

The notation for fault trees, cause consequence diagrams and HAZOPS used here is one developed for convenient and efficient analysis of large systems. The notation uses lines of text to define events, and leading hyphens to define the level of an event in a fault tree. OR gates are not used. If two events follow each other in a list, and are at the same level (i.e., have the same number of hyphens) in a fault tree they are defined as being alternative causes. AND gates are recorded by means of an ampersand symbol. As an example, the following event list form defines some kinds of crashes:

```
Vehicle crash
- Vehicle leaves the road
-- Loss of vehicle steering control
-- Loss of road keeping due to ice on the road
- &
-- Obstacle on the road
-- Too late braking
```

Consequence diagrams can similarly be written as event lists, for example:

```
Loss of lane keeping
- ? Vehicle is on a straight stretch of road
-- Y Vehicle continues in a straight line along the section
--- ? Vehicle recovers lane keeping
---- Y No crash
---- N Crash when a curved stretch of rod is reached
-- N Crash
```

Safety measures (safety barriers) are represented by a + symbol

This notation is much faster to write than traditional graphic representation of event trees and cause consequence diagrams and is much more expressive than FMEA or HAZOP tables. The HAZEX program can automatically translate the event list form to a graphic form.

The only problem found with this notation has been that it can become quite difficult to keep track of the numbers of hyphens. For this reason, it has been found preferable to divide fault trees into pages and provide connections between the trees. The symbol >> is used for this.

3. A design for an autonomous vehicle

This chapter describes the design for an autonomous vehicle intended to be used as a basis for selection of hazard identification of risk analysis techniques. The vehicle is to be developed as a ¼ scale model, to be driven on a protected track, but the design is made for a saloon car vehicle which could, if full scale, be driven in general traffic. The motivation for the development is to ensure that practical details are identified which can contribute to failure.

Detailed studies of vehicle accident and recall data show that vehicle failure and design error induced accidents are often the result of very detailed, sometimes very small, deficiencies in design Taylor 2020). These set the target level of detail for hazard identification in autonomous vehicles. While the model does not show the same detail design errors as a full-scale vehicle, it does show some of the deficiency types which are typical. The ¼ scale model provides a level of reality which theoretical models and simulations cannot.

3.1 Design philosophy

The design is intended to be able to be driven in general traffic under normal driving conditions. This implies that it must be able to cope with a full range of driving conditions, traffic conditions and circumstances, anomalous conditions on the roadway such as partial road closures for roadworks, traffic accidents et. It must also be able to cope with driving in the presence of vehicle failures, such as being able to cope with brake failures, engine failure etc. and should be able to perform the emergency driving procedures, such as stopping safely at the road edge.

For simplicity, the design is made for level 5 capability, that is completely automatic driving. No fallback capability, with the possibility of a human driver taking over. This is of course not practical, because it would involve large difficulties in system testing, but is convenient for risk analysis methodology validation, because it allows analysis human machine interaction issues to be postponed to a later design.

GPS and electronic maps are used for route finding but not for detailed navigation tasks such as lane keeping. The design does not include off-road driving capability.

The design is made using diverse redundancy. Visual systems are used for lane following and road sign recognition, other vehicle recognition and pedestrian recognition. The visual systems are backed up by lidar detection for objects on the road. Radar sensors are used for parking.

The design is also made to interface to a standard engine control system and a brake by wire system. To be able to cope with vehicle failures the design includes a diagnosis system which can take failure signals from safety sensors and data from performance system such as braking performance and engine performance.

In order to be representative of real-world conditions the design is based on readily available hardware components, and on open-source software modules. The use of a ¼ scale model vehicle necessitates deviations from the engineering of a full-scale vehicle. For instance, the steering and braking systems for the full-scale vehicle are not the same as for the model. For this reason, the hazard identification for these subsystems is made on the basis of full-scale vehicle design.

Software is kept simple, primarily with single loop direct scheduling of functions, and with separate processors for functions requiring high performance. This design avoids the need for an operating system and requires just low-level device drivers for sensor input and actuator output, processor to processor and function to function communication. The rationale for this design is that it makes software validation relatively simple when compared for example with an interrupt driven operating system kernel.

3.2 The system functional description

There are several functions which are essential for an AV intended for mixed traffic driving:

- Destination input
- Route finding and route following
- Lane keeping
- Lane change and traffic filtering
- Lane merging
- Speed control
- Obstacle detection and avoidance
- Negotiation of junctions and cross roads
- Overtaking
- Parking
- Traffic sign and traffic signal recognition and response
- Other vehicle detection and avoidance (Trucks, cars, motorcycles, bicycles
- Human detection and avoidance
- Animal detection and avoidance
- Special road conditions recognition and response such as wet surfaces, icy conditions
- Special traffic situations recognition and response, such as road works, traffic jams
- Emergency situations monitoring and emergency action
- Vehicle condition monitoring and response

In the following each of these functions is described separately. The integrated system is shown in Figure 3.1.

Destination input and route finding are based directly on the functions of an automobile navigation aid, using GPS and electronic maps. Additional details in this though are segmentation of the rout into sections which require different steering and speed control, such as identification of straight and curved route segments, junctions, filter lanes and identification of the number of lanes. For this demonstration model, some road features such as single lane roadways with passing points are omitted from the design.

Route finding is made by a standard network search algorithm with priority weighting for selection of the next route section based on road type and destination distance, and with overall route optimum detection.

The output from route finding passed to the navigation control function as a stream of route segments with segment parameters. Feedback is also obtained from the overall navigation control to take into account possible route problems arising, and from GPS to ensure that the vehicle is actually following the correct route (this last function should in principle be unnecessary for the final validated design, but is useful during testing). The feedback activates a route correction function if this is needed.

The main navigation control function determines the navigation algorithm to be selected in steering and speed control. Its specification is given later in the form of a decision matrix, with input from the route-finding system in the form of route segment type, from the current speed and steering mode, obstacle and traffic finding, road condition and vehicle performance information. The output selects steering and speed control modes.

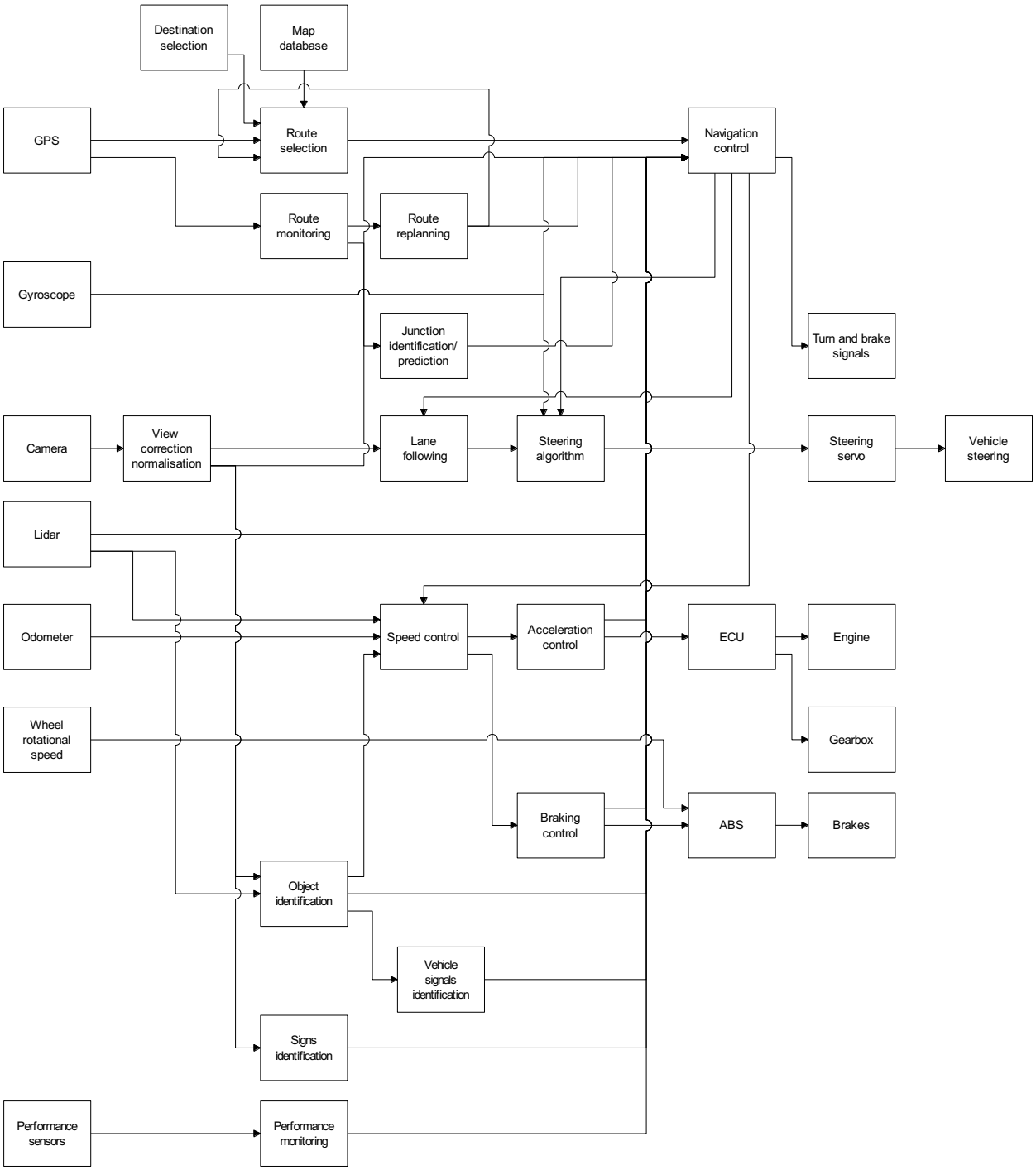


Figure 3.1 Top level functional block diagram for the autonomous vehicle

Lane keeping is a core function in the design, in that keeping lane is essential during straightforward driving, and is essential also when lane changing, and junction navigation. Lane keeping and changing are controlled via a vision system with camera input. This input is corrected to a cartesian view with correction for lens view coordinates (fisheye effect correction) and for lighting level normalization. Histogram calculation and peak finding is used for lane marking detection, with a dynamic algorithm to bridge gaps and noise in lane marking detection. Lane center target points and lane curvature are passed to the steering function. The steering function has several algorithms, for simple lane following, for junction approach and turning strategies, for overtaking strategies and for obstacle avoidance strategies. For this purpose, the steering function needs information about other traffic and obstacles, which could in principle be obtained directly from Lidar and camera data, after object identification. However, to ensure coherence of functioning, this information is first processed by the main navigation control, and from here passed to the steering function.

The output from the steering function is passed to the steering servo which in turn sets the steering track angle.

The engine is assumed to be of internal combustion type, with gasoline fuel. This assumption is perhaps unrealistic for an autonomous vehicle, but the assumption is made because it introduces problem types which place additional demands on the hazard identification methods, including operating phase transitions at gear change.

The speed control function receives commands for the desired speed and the acceleration or braking response curves to be followed and from the odometer to provide feedback on actual speed. The main navigation control also provides information on road condition and brake/tire condition, to modulate the braking response. The actual braking is carried out by an automatic braking system, which also provides anti-skid capability

The AV needs input about other traffic in order to be able to adjust vehicle spacing, and therefore speed. Obstacle and other vehicle location are provided by Lidar and vision based input from the camera, which pass data to object and traffic recognition. The data are passed directly to speed control for vehicle distance keeping, but also to the main navigation control for navigation strategy selection. Emergency action is performed as a high priority function of the main navigation controller.
Traffic and object recognition are carried out by the respective functions. The processing uses vision algorithms to detect and localize objects and neutral networks for identification of object types. The algorithms are dynamic in order to provide for changing distance and to dropouts for individual camera frames. Neural networks are also used for reading of sign text.

The performance monitoring function takes sensor input for steering, engine and braking performance, and component function, and passed modulating or emergency information to the main navigation control function.

3.3 The navigation control module

For this kind of system, parametric brainstorming is currently the most effective hazard identification method. For the car, the starting point for the hazard identification can be a crash or a simple failure to operate (car slows down or stops, for example in the middle of a highway). Concentrating on crashes, a car can crash into:

- A vehicle in front
- A vehicle from behind
- An overtaking vehicle
- An oncoming vehicle
- An object on the highway
- Roadworks on the highway
- A person on the highway
- A cyclist on the highway
- An animal on the highway (consider for example, an elk)

The car can also leave the road and crash into:

- Crash barrier
- Parked vehicle
- Tree
- House or other building
- Ditch
- Person

A second dimension of the hazards is the road situation:

- On a straight clear road
- On a curve
- At a turn
- At a turn off
- At a road merge (several types)
- Approaching a side junction
- At a crossing
- At a roundabout

The driving situation can also vary:

- Straight driving on a clear road,
- Approaching traffic ahead
- Pulling out for overtaking
- Driving in an overtaking lane
- Cutting in after overtaking
- Several other possibilities when there are three of more lanes

There are several other parameters to this hazard identification problem, such as road state, visibility, cyclist position, and importantly, failures in the car itself. It is not necessary here to continue the list since the point is made. There is a large number of scenarios, each of which needs to be taken into account in designing for safety. There is, in a full evaluation at this level of detail, an extremely large number of accident scenario types. The autonomous vehicle systems need to be designed and validated for each of these. In some cases, two kinds of adverse condition may need to be taken into account at the same time.

Obviously, it will be very difficult to design controls for an autonomous vehicle which applies a specific set of control rules for every possible driving accident scenario. Fortunately, this is not necessary, if the risk analysis is used as an input to the design process. Firstly, the factorial analysis from the brainstorming can be used to construct an event tree for the actually possible scenarios. Secondly, it is possible for some of the accident factors to be consolidated as a single parameter. Ice, mud or poor road surface, and tire wear can be conflated, for example, into a single parameter, road holding capability. What the hazard identification does do is identify the wide range of sensors needed

for a fully functional autonomous vehicle, including road temperature, braking capability, obstacles ahead and behind, and obstacles ahead which may cross the path, both vehicles, bicycles, motorcycles, humans and animals.

Taking as an example for analysis, leaving the road on a straight section with good road condition, this can be caused by failures in the steering (steering jammed), failures due to driving instability, failures due to looseness in the steering subsystem and steering excursions (such as hard right or left due to broken linkages). In other words, it is possible to take the general scenarios and localize them to individual subsystems in the autonomous vehicle. This localization can cover both problems generated by the vehicle, such as steering errors, and problems arising due to external situations such as a sharp curve in a road, or a section of road susceptible to aquaplaning. From this point hazard identification can use methods such as FTA or HAZOP to trace the causes of the hazardous events.

3.4 System description for STPA

The STPA method was developed specifically for enabling analysis of control loops, and particularly, multi-level control loops. For this purpose, it is important to be able to recognize the control loops in the system. Figure 3.2 shows the same system as that in Figure 3.1, but redrawn with the control loops made obvious.

3.5 Lane following

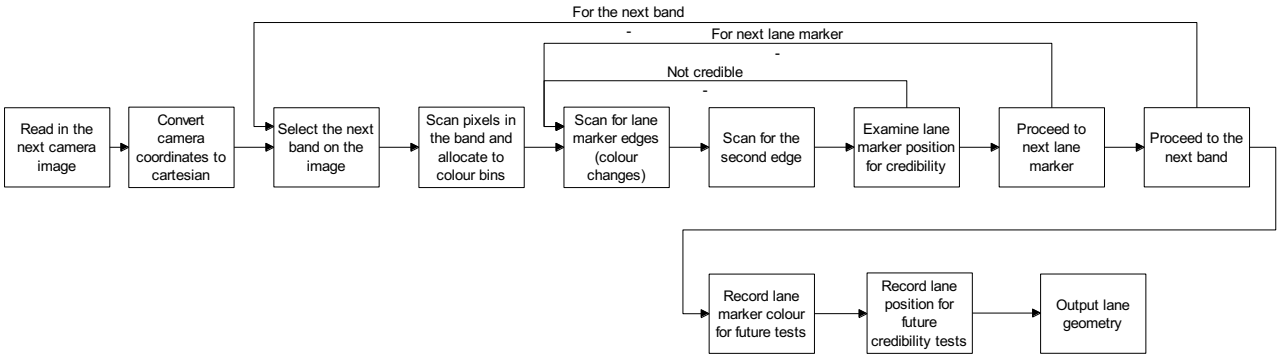


Figure 3.3 shows a functional block diagram for an adaptive lane detection algorithm. It involves scanning across the camara image for changes in pixel group color which indicates the left edge of a lane marker, then scanning for a right edge. The algorithm is adaptive in that the color range for the roadway and lane marker are adjusted continuously. The scanning is made in selected bands to minimize calculation load. Credibility values derived from earlier lane detection are used to reject false positive detection. In the full algorithm the search for edges starts from the predicted lane marker position.

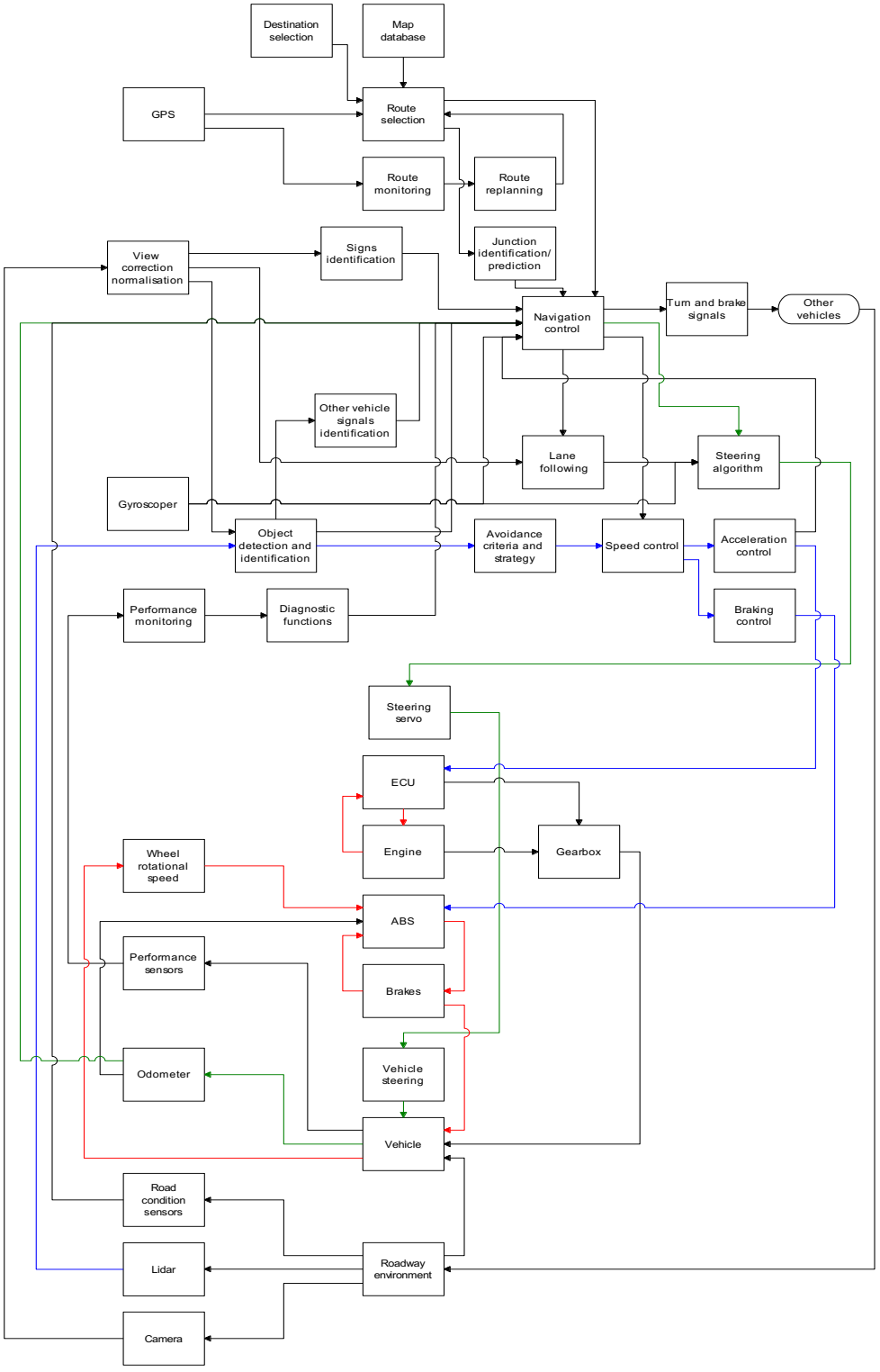


Figure 3.2 The AV system drawn with control loops emphasized and some control loops highlighted in color. Diagram to facilitate STPA

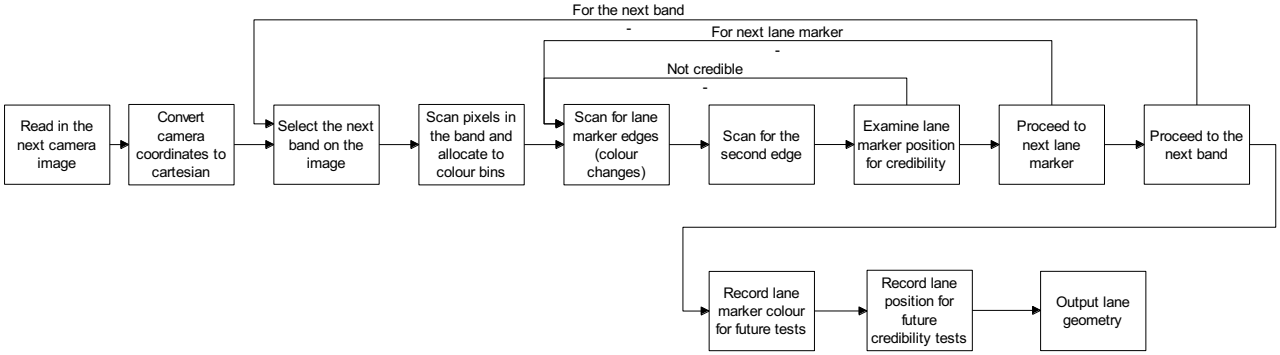


Figure 3.3 block diagram for a lane following algorithm

The credibility checking for a lane marker involves continuity checking from previous scans and marker position prediction, and consistency between lane marker positions found at different bands. If no credible lane markers are found, the lane marker position recorded is based on prediction from previous lane marker detections.

3.6 Steering algorithms

A number of steering algorithms are implemented to manage the steering tactics and direct steering control. The different algorithms are activated by the navigation controller in response to the recognition of different road and traffic situations.

An example is the algorithm for the first stages of overtaking. Checking that there is sufficient distance to the vehicle ahead, checking that there is an overtaking lane and checking that it is free is carried out by the navigation control. Calculating and timing the direction changes is performed by the overtaking steering module.

The strategy for moving into an overtaking lane follows observations of human drivers, The first part is changing direction by applying a constant steering angle until the desired heading is reached, followed by a return to straight line driving until the desired lateral distance is reached, followed by a new steering angle to return the car to the road heading. The path is shown in figure 3.4. Before initiating the maneuver, the algorithm first calculates its parameters, and then check feasibility. For example, the own vehicle should not be too close to the leading vehicle.

The overtaking trajectory module monitors the manoeuvre continuously from inception to completion, using the odometer to determine the distance travelled and the gyroscope to determine heading as feedback signals, correcting any deviation from the intended trajectory. The steering module reports back to the navigation module as each segment of the manoeuvre is completed.
An important feature of the module is that it has exit strategies for cases of safe lane change when the AV is approaching a car ahead (see Table 3.1 below).

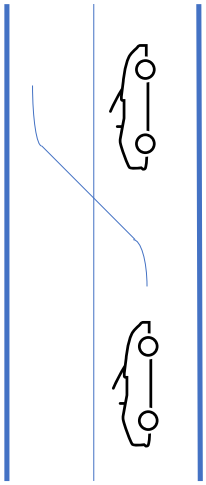


Figure 3.4 Steering trajectory for overtaking

3.7 Speed control algorithms

Speed control algorithms are needed for:

- Starting the vehicle
- Attaining the target speed
- Accelerating for overtaking
- Deceleration to the target speed
- Braking to achieve the target speed when speed limits change
- Deceleration to avoid coming too close to a leading vehicle
- Deceleration when approaching a junction, a crossing with slow or full stop
- Deceleration when approaching a turn off
- Emergency braking to avoid a crash
- Braking to come to a full stop

When decelerating, a choice must be made to decide whether easing off of the engine is appropriate, or whether braking is needed, and the degree of braking to apply. For this, current speed, speed of any other vehicles, and speed limit values are needed (including speed limits for approaches to junctions etc.). Braking and engine speed performance curve are also needed, and the design includes monitoring of performance in order to keep the curves up to date, especially for the braking in adverse weather. The speed control function is also responsible for gear changes.

3.8 Navigation control

The navigation control is intended to determine what kind of steering and speed control should be carried out at each condition of driving. The software is designed as a finite state machine, with state transitions depending on a large number of conditions. The signals for the conditions come from sensors directly, and also from calculation, detection and recognition modules. The navigation controller is responsible for adapting to road and traffic conditions, including emergency conditions, and is also responsible for responding to internal conditions such as loss of braking performance.
Some conditions for transition may arise at the same time, and the decision about which transition to take is based on a priority system with normal driving being priority 1, and more important transitions such as those for emergency conditions having higher priority.

The navigation system is described by means of a finite state diagrams (example in Figure 3.5), but the transition conditions are too complex to be described on the diagram and are therefore specified in the form of a decision table. An example of part of the table is shown as Table 3.2.

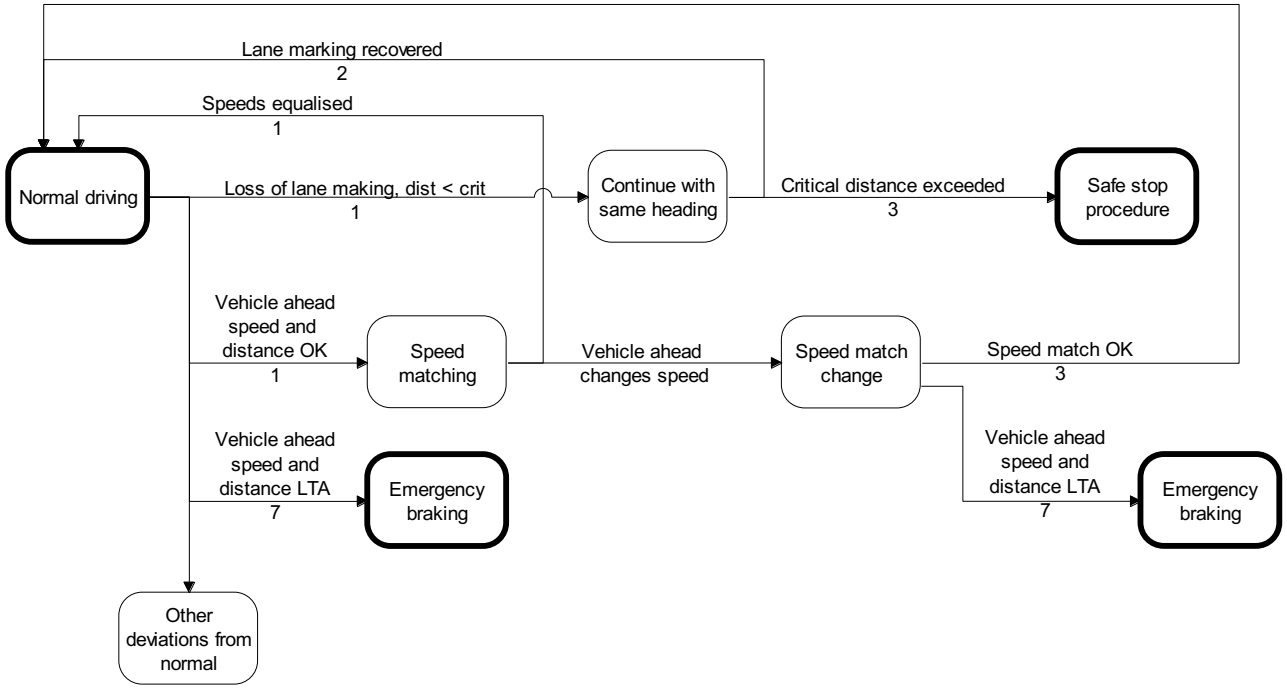


Figure 3.5 A small part of the finite state diagram for the navigation controller

In the course of driving there will be many navigation modes changes. It is important that each driving mode there is a full range of mode termination actions. It is also important that the changes are smooth, with no drastic accelerations and decelerations. It is also necessary that the changes are timely.

Table 3.1 Decision table for speed adjustment when approaching a leading car

Current state	**Condition**	**Group**	**Priority**								
Normal driving	Road straight or curve	1	1	*							
	No junction ahead within hazard threshold			*							
	No turn out ahead within prep distance			*							
	No filter lane ahead within prep distance			*							
	No car ahead within speed eq distance			*							
	No obstacle ahead within safe braking			*							
Normal driving	Loss of lane marking < threshold distance	2	2		*						
	Road is straight				*						
Normal driving	Loss of lane marking > threshold distance	3	3			*					
	Road is straight					*					
Normal driving	Loss of lane marking < critical for curve	4	3				*				
	Road is curved						*				
Normal driving	Vehicle ahead	5						*			
	Speed delta allows for speed adjustment							*			
	Next state = Speed matching							*			
Normal driving	Vehicle ahead	6									
	Speed and distance LTA										
Speed matching	Speeds equalised	7	1						*		
Speed matching	Vehicle ahead changes speed	8	7								
Sp match change	Speed match OK	9	3							*	
Sp match change	Speed and distance LTA	10	7								*
	Action										
	Next state = Normal driving			*	*				*	*	
	Follow lane			*							
	Continue driving with same heading				*						
	Next state = Emergency braking					*	*				
	Next state = adjusting speed							*			
	Adjust own speed							*		*	
	Next state = Speed match change										
	Next state = Emergency braking										*

Table 3.2 Decision table for the overtaking function

Current state	**Condition**	**Group**	**Priority**							
Normal driving	Vehicle ahead									
	Speed much lower than target speed	11	2	*						
	Overtaking lane exists			*						
	No car ahead in overtaking lane in crit dist			*						
	No car alongside			*						
	No car coming from behind in overtaking lane			*						
	No crossing, lane merge or filter lane ahead									
Overtaking1	Correct steering angle reached for pulling out	11	2		*					
Overtaking 2	Correct distance reached for pull out	13	2			*				
Overtaking 3	Distance for overtaking adequate	14	2				*			
	Leading car has not increased speed too much						*			
Overtaking 4	Overtaking distance is sufficient	15	2					*		
	No cars ahead to overtake							*		
Overtaking 4	Cars ahead in inner lane	16	4						*	
	Insufficient space to pull in								*	
Overtaking 3	Car ahead accelerates too much	17	4						*	
Overtaking 3	Car ahead in overtaking lane	18	4							*
	Car ahead too close									*
	Action									
	Next state = Overtaking 1			*						
	Start overtaking algorithm			*						
	Set steering angle to zero				*					
	Next state = overtaking 2				*					
	Set steering angle for for straigtening out					*				
	Next state = overtaking 3						*			
	Next state = overtaking 4						*			
	Start pull in algorithm							*		
	Next state = Pulling in 1							*		
	Start pull back algorithm								*	
	Next state = pull back 1								*	*

4. Hazard analysis for the vehicle

4.1 Hazard identification strategy

Because this study is aimed at investigating the capabilities of various risk analysis approaches, a variety of hazard identification methods and method combinations were used. As will be seen, the main method used for hazard identification is fault tree construction. The reason for this is that the critical events for the system are crashes and disengagements, and that the systems are redundant so that AND/OR logic is needed. To ensure consistency and depth in the analysis, an initial phase of failure mode and effects analysis for the components of the system is useful. Also, STPA is useful for the control loops. Cause consequence analysis is needed for the parts of the system which require sequence and timing, such as transition between navigation modes ad emergency actions. The results from these supplementary analyses are incorporated into or are used as a basis for the fault tree.

The key events for the risk analysis are crashes and brainstorming and morphological analysis were used to identify the types of crashes which would be of interest, and the proximate causes of these. This approach leads directly to the use of fault tree analysis as a suitable for identification of crash causes. Many of the possible causes of crashes are dependent on timing, for example time for braking versus time for obstacle recognition. For this reason, sequential, timed fault trees are used. Since some failure modes are dependent on numeric parameters, such as speed and road condition versus braking, hybrid discrete and continuous fault trees are used where relevant.

The descriptions for the AV design use a hierarchy of functional block diagrams, with low level algorithms described by pseudocode and grafcets. For this reason, the fault trees are also multilevel, with hierarchies of functional failure fault trees at the upper levels, and software fault trees at the detailed level.

The design involves several control loops, with feedback of speed, location, performance etc. to correct the feed forward control system. For this reason, an extended version of the STPA method was used as part of the FTA process.

Several of the AV functions require neural networks. No standard method for hazard analysis of these could be found in the literature.

Functional failure mode and effects analysis is a standard approach to failure analysis for functional block diagrams and was used as an alternative to fault tree analysis. An extended list of functional failure modes was used. Symbolic simulation was used to perform the failure mode consequence determination at the program code level.

Methods
Since crashes and disengagements are the primary hazards of concern in this study, the fault tree analysis method is initially the most promising method for hazard identification. However there is a need to be able to deal with dynamic situations, including delays, such as braking vs separation distance, cornering vs vehicle speed etc. There is also a need to be able to analyze systems in which continuous variations are important, such as speed and acceleration, lane width, vehicle proximity and road curvature. For this reason, the fault trees used are sequential, recognizing the difference between events and states, and incorporating time delays, and are also hybrid, including not only discrete events but also continuously varying processes.

The FTA was initially made on the initial functional block diagram. For some of the high level block more detailed block diagrams were made, so that the analysis is multi-level.

The fault tree is very extensive, and it was found to be difficult to retain consistency during the analysis process, both in terms of the degree of detail incorporated into the tree and in the terminology used to describe events. The analysis was repeated by making a failure mode and effects analysis for each functional block (FMEA) and then using these as models in an automated fault tree analysis.

4.2 FMEA/FFA/HAZOP for the system components

Failure mode and effects analysis was carried out for all of the system components, Since a large part of the system consists of software components described in block diagram form, the failure modes are largely functional failures so that the FMEA is actually a functional failure analysis (FFA) Since many of the failure modes involve continuous parameters such as speed and braking time disturbance modes with values such as too high, too low etc. are needed, and the analysis effectively becomes a HAZOP. Table 4.1 show an FMEA for one component of the control system.

One thing which became immediately clear when carrying out the FMEA after completing an initial FTA analysis is that it is much easier to create an in-depth failure cause analysis using FMEA than it is with an overall system FTA. The reason was also clear. With FMEA, the analyst focuses on one small part of the problem and is not distracted by the need to consider the overall system context.The FMEA can fairly easily be integrated into the overall FTA, especially if semi-automated analysis is used.

Table 4.1 FMEA for the steering calculation for overtaking

Component	Failure mode	Failure causes	Consequences	Safeguards
Steering turn calculation for overtaking	No calculation No output	Error in navigation control Error in state diagram Interfering mode transition with higher priority blocks the function Controller failure to function	No steering output - AV fails to turn -- AV continues in same lane --- AV crashes into leading vehicle	Lidar provides distance measurement to leading vehicle - Emergency stop
	Steering turn angle too large	Attempt to overtake at too close distance - Error in minimum distance criterion Error in turn calculation parameters	Too large steering turn output - Excessive lateral acceleration -- Passenger discomfort - Possible oversteer -- Possible uncontrolled skid --- Possible crash	Gyroscope feedback check Possibly introduce an oversteer monitoring function in future
	Steering turn angle too small	Error in turn calculation parameters Leading vehicle has a wide load Lane width greater than norm and leading vehicle not centered in lane	Too small steering turn output - Possible inadequate lateral distance -- Possible side swipe to leading vehicle -- Possible rear end shunt	Gyroscope feedback check Obstruction detection and width check
	Turn duration too long	Error in turn calculation parameters Error in turn calculation	Too large steering turn output - Excessive lateral acceleration -- Passenger discomfort - Possible oversteer -- Possible uncontrolled skid --- Possible crash	Gyroscope feedback check
	Turn duration too short	Error in turn calculation parameters Error in turn calculation	Too small steering turn output - Possible inadequate lateral distance -- Possible side swipe to leading vehicle -- Possible rear end shunt	Gyroscope feedback check

Component	Failure mode	Failure causes	Consequences	Safeguards
Steering drift calculation for overtaking	Drift phase too short	Error in drift calculation parameters Lane width less than norm Leading vehicle has a wide load	Too short duration of lane shift - Possible inadequate lateral distance -- Possible side swipe to leading vehicle -- Possible rear end shunt	Odometer and gyroscope check
	Drift phase too long	Error in drift calculation parameters Lane width greater than norm and leading vehicle not centered in lane	Too long duration of lane shift - Excessive lateral distance - Possible collision with overtaking lane barrier	Odometer and gyroscope check Possible overtaking lane boundary chek in future implementation
Steering heading restoration	No calculation No output	Error in navigation control Error in state diagram Interfering mode transition with higher priority blocks the function Controller failure to function	No steering output - AV fails to turn -- AV continues moving to overtaking lane barrier --- AV crashes into barrier	Odometer and gyroscope feedback check
	Steering turn angle too large	Error in turn calculation parameters	Too large turn angle - AV moves back into inner lane -- Possible crash into leading vehicle	Odometer and gyroscope feedback check
	Steering turn angle too small	Error in turn calculation parameters	Too small a turn angle - AV continues to drift to overtaking lane barrier - Possible collision with overtaking lane barrier	Odometer and gyroscope feedback check
Steering heading restoration	Turn duration too long	Error in turn calculation parameters	Too large turn angle - AV moves back into inner lane -- Possible crash into leading vehicle	Odometer and gyroscope feedback check
	Turn duration too short	Error in turn calculation parameters	Too small a turn angle - AV continues to drift to overtaking lane barrier - Possible collision with overtaking lane barrier	Odometer and gyroscope feedback check

4.3 STPA and emergent hazards analysis for the control loops

While FMEA focusses on the components directly implementing the system functions, STPA focusses on the feedback loops. An STPA like analysis for the overtaking steering control loop is given in Table 4.2.

Table 4.2 STPA like analysis for the overtaking steering control loop

Component	Failure mode	Failure causes	Consequences	Safeguards
Steering feedback control loop **for overtaking drift phase**	Gyroscope not functioning	Instrument failure Power failure Signal bus failure Connector failure	No feedback signal to check heading - Conflict between predicted heading and heading as measured - - Consequence depends on failure response policy	A zero signal or no signal is easily detected
	Gyroscope output too low	Instrument failure Calibration drift	Conflict between predicted heading and heading as measured - Consequence depends on failure response policy	Emergency response subsystem activated by lidar - Abandon overtaking -- Pull back mode -- Emergency stop
	Gyroscope output too high	Instrument failure Calibration drift	Conflict between predicted heading and heading as measured - Consequence depends on failure response policy	Emergency response subsystem activated by lidar - Abandon overtaking -- Pull back mode -- Emergency stop
	Odometer not functioning	Instrument failure Power failure Signal bus failure Connector failure	No feedback of distance travelled - Inability to determine necessary speed for overtaking - Inability to determine when lane switching is complete -- Failure of overtaking -- Possible rear end crash into vehicle being overtaken	The most important protection gains odometer failure is that it will be impossible to commence driving without the odometer. Only failure during the overtaking phase will be relevant . Odometer functional failure is easily detected and should result in emergency stop.
	Odometer output too low	Instrument failure Calibration drift	Inaccurate feedback - Inaccurate distance travelled calculation in overtaking control module -- Effective lane transfer not achieved --- Possible rear end crash	Failure can be checked by odometer- lidar comparison. Errors would be detected early in a trip.
	Odometer output too high	Instrument failure Calibration drift	Inaccurate feedback - Inaccurate distance travelled calculation in overtaking control module -- Possible movement to opposing lane or crash into barrier	Failure can be checked by odometer- lidar comparison. Errors would be detected early in a trip.

Component	Failure mode	Failure causes	Consequences	Safeguards
	Controller failure to function	Error I the navigation control module Interruption by another higher priority	Inability to initiate overtaking Possible accident when approaching a vehicle in front or a stationary vehicle	This requires confirmatory feedback between the overtaking module and the navigation controller
	Controller output delay	Interruption of the overtaking controller by higher priority actions	Drift phase prolonged and AV drifts into opposing lane or into barrier - Possible crash	Needs investigating
	Steering function fails stuck	Steering hydraulic steering fails, see FMEA	Crash	
	Steering function fails stuck	Communication fails	Crash	Communications monitoring
	Steering function fails to zero angle	Steering hydraulic steering fails, see FMEA	Crash	Gyroscope monitoring detects failure
	Steering function fails to zero angle	Communication fails	Crash	Communications monitoring
	Steering function fails high	Steering hydraulic steering fails, see FMEA	Moves beyond overtaking lane - Crash into barrier or oncoming vehicle	Gyroscope monitoring detects failure
	Steering function fails low	Steering hydraulic steering fails, see FMEA	Rear end crash into leading vehicle	Gyroscope monitoring detects failure
Emergent hazards	Slow response and overshoot, reset wind up	Any failures introducing delays in the control loop. When the feedback from the odometer and/or the gyroscope is delayed the control delta increases and can lead to overshoot in the steering. Possible causes in delay are changes in navigation mode arising from incoming high priority events, and work loads in the control processors	Possible crashes if there is a steering overshoot or undershoot while overtaking due to loss in trajectory accuracy.	Careful determination of computation load, and use of dedicated processors. Detailed analysis of the navigation controller simultaneous progesses generating input, including AV equipment failures, changes in the road environment and changes in the traffic situation
	Changes in control loop parameters	Various failures in control loop parameters are possible including response of hydraulic drives (for the physical model of the study, the electrical servo) to the steering	Over or under response in steering Possible hunting or judder in the steering	
	Backlash	Backlash in the steering can be caused by looseness or wear in the steering.	Delay in steering control can lead to steering trajectory errors	

4.4 Cause consequence analysis for a tactical navigation control sequence

There is a fairly high number of failure types for the AV which involve dynamics. These include speed equalization when approaching a leading vehicle, overtaking, approaching a junction, turning at a junction etc., as normal tactical navigation acts, and emergency actions such as responding to a vehicle crossing in front. For dynamic failure problems, the cause consequence analysis metod is the most suitable

The cause consequence analysis can be easily converted into a fault tree and can then be incorporated into the overall analysis. The cause consequence analysis process is much easier to understand and use in making an accurate and complete dynamic analysis.

An interesting omission in this analysis is that described in Ch 9, case 4, in which a collision into a stationary truck occurred because a leading car overtook a stationary truck, and the AV could not respond sufficiently quickly to the reveal and also overtake This emphasizes the need to try to imagine all possible alternative situations which could arise at each stage in a scenario. The analyst, when completing part of a study like that in Figure 4.1, should make an effort to challenge it, for example using morphological analysis. As an example, knowing the problem of hidden stationary vehicles from case 4, the following list can be derived:

- Other obstacles, like the accident attenuator from case 4
- Poorly lit stationary vehicles or obstacles
- Obstacles in fog conditions
- Obstacles with low contrast against the background e.g. camouflage against a background of trees
- Approaches around a blind corner

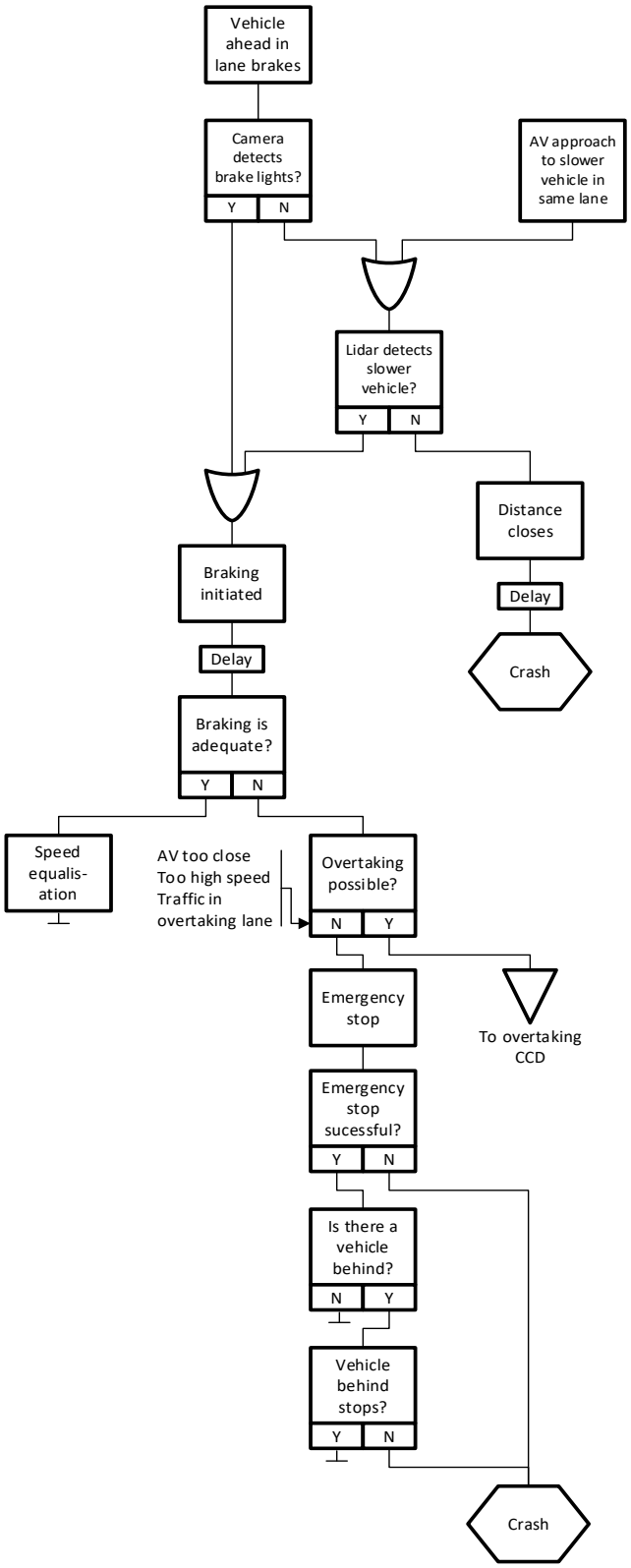


Figure 4.1 Cause consequence diagram for failures in speed equalization

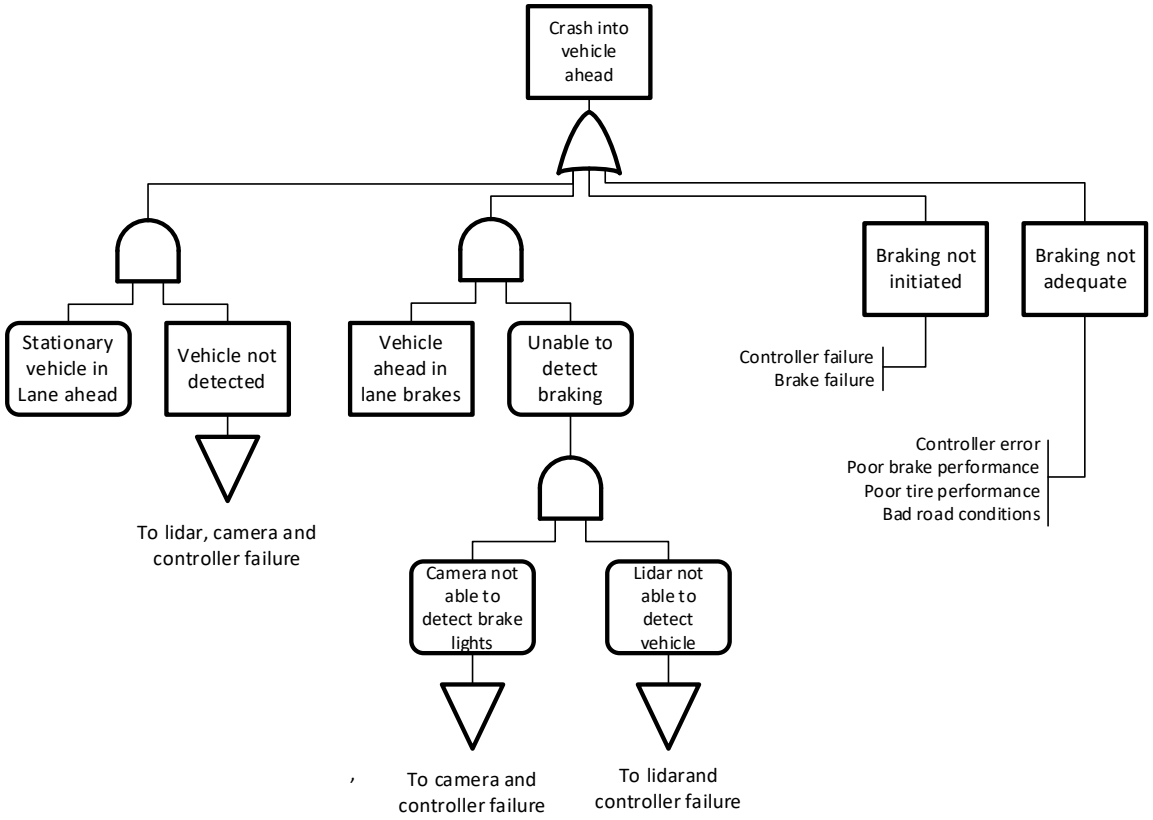


Figure 4.2 Sequential fault tree corresponding to the CCD

4.5 The AV system FTA

Figure 4.3 shows the highest level of the analysis for the top event "Crash". It shows the different types of crashes which can occur. In making the selection of crash types of the different crash geometries are taken into account and also the different kinds of detection required.

1 Crash
2 - Crash due to car leaving the road >> 20
3 - Shunt into leading car >> 1
4 - Following car ear end shunt >> 30
5 - Head on crash
6 -- Own car in the wrong lane >> 40
7 -- Oncoming car in the wrong lane >> 50
8 - Rear end shunt >> 60
9 - Crash into oncoming car while in lane >> 70
10.- Crash with car in outer lane when pulling out to overtake >> 80
11 - Crash into oncoming car while overtaking >> 90
12 - Rear end crash while overtaking >> 100
13- Crash due to vehicle entering from a side road >> 110
14 - Crash at junction >> 120
15 - Crash into obstruction on the road >> 130
16 - Crash into person on the road >> 140

Figure 4.3 The highest level of the fault tree

The remaining parts of the fault tree each start with a particular type of fault tree, and then consider what kind of unintended or adverse movement of the vehicle is involved. Starting at the wheels, the potential causes of the crash are then traced backward through the block diagram, identifying the possible failure modes of each block, the possible causes of the failure mode which are internal to the block, and the input events which could cause block functional failure. The failure modes considered at each stage were:

- Failure to function
- Spurious (unintended) function
- Erroneous function
- Excessive function (some parameter of the function has too high a value)
- Inadequate function (some parameter of the function has too low a value)
- Delayed function
- Premature functioning

The notation in the fault tree uses simple text lines to give the events, and hyphens to indicate the level of events in the fault tree. If an event A in the tree is preceded by two hyphens, for example, and the two following lines B and C have three hyphens, this represents an OR gate in which event A has two possible causes, event B or event C. An & symbol is used to represent an AND gate. This notation can be automatically translated to a conventional graphic fault tree by software.

Line 71 to 86 considers closing speed prior to a crash into the rear end of the leading car. This is a problem with several continuous parameters, own car speed, leading car speed, separation distance at the time of detection, and braking effectiveness and is shown here because it provides a good example of the use of hybrid fault tree methods. This gives a five dimensional space, in which there is a domain which lead to safe speed equalization,

a domain where the own vehicle speed decreases so that the leading vehicle is not approached, and a domain in which a crash occurs. Figure 4.4 shows a projection of the domains onto the distance / initial speed axes. This description is in fact a simplification. In practice a good speed control algorithm stars by reducing engine speed, and only applies brakes if speed equalization distance is observed to be greater than the actual distance. An algorithm of this kind would need to recalculate stopping distance continuously until speeds were equalized, and this is incorporated into the design.

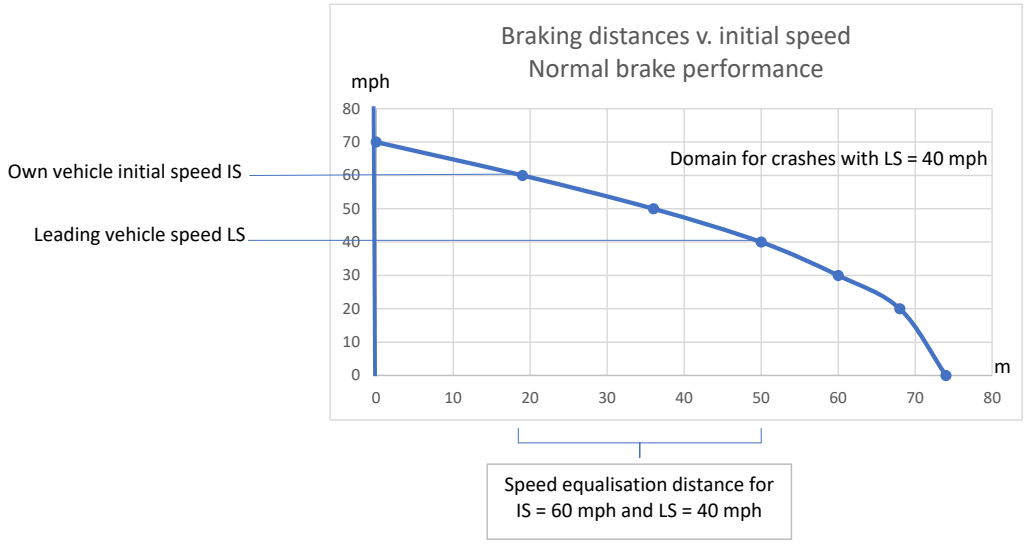


Figure 4.4 Braking curve and determination of minimum distances for speed equalization

67 8 >> Critically delayed signal from avoidance strategy
[braking to avoid a collision with the leading car]
68 - Erroneous avoidance strategy
69 - Algorithm error
70 - Criterion error
71 - ~ Speed equalization distance (SED) > separation distance (SD)
{SED = f1(Braking force, road condition, leading car speed, own car speed}
{SD = leading car relative distance at detection}
72 -- Speed equalization distance critically high
73 --- Braking force critically low
74 --- Road condition critically bad
75 --- Leading car speed low (or stopped) [in this case either the design safety distance is too low or the detection distance is too large]
76 --- Leading car speed estimate critically high
77 --- ~ Erroneously high speed estimate for leading car from object recognition >> 9
78 --- Own car speed critically high >>
79 --- ~ Critical combination of own car high speed and poor braking or bad road condition
80 -- Separation distance at braking critically low
81 --- leading car distance estimate critically high
82 ---- Erroneously high speed estimate for leading car from object recognition >> 9
83 ---- Erroneously high distance estimate for leading car from object recognition >> 10
84 - Wrong prediction of behaviour of object in front >>
85 - Delayed input from object detection >>

Figure 4.5 Subtree speed adjustment failure when approaching a leading vehicle

(made independently of the analysis in Figure 4.1. Other parts of the fault tree are shown in the following chapters and in the appendix.)

5. Hazard analysis for vision and lidar algorithms

Vision algorithms can be used for specific functions in an AV controller. These may be less flexible than neural networks, but can be much more efficient. Examples are:

- Searching for road lane markings and road edges
- Searching for vehicles and providing bounding boxes for vehicles
- Searching for humans and providing bounding boxes

When compared with the use of neural networks for vision, vision algorithms can have the advantage that they can in some cases be analysed for performance and safety. For the av used for evaluation here, an algorithm for lane following is used.
This algorithm is sufficiently simple that the hazards can be identified using software fault tree analysis. As an introduction to the analysis is made using a fault tree analysis based on functional failure modes

5.1 The lane following algorithm fault tree

The first part of the fault tree for lane following is an extract from the overall fault tree.

>> Vehicle leaves the road
- &
-- Lane detection is needed
--- Roadway is curved
--- Vehicle is manoeuvering
-- Lane detection fails, one or more lanes not detected >>

Figure 5.1 The subtree for vehicle leaving the road due to lane detection failure

The next subtree covers failures in the lane detection algorithm:

>> Lane detection fails, one or more lanes not detected
- Lane geometry not correctly transmitted
-- communication error
- The lane scanning process fails
-- The credibility checking process rejects a legitimate lane marker
--- Criteria for credibility set too tight
--- To rapid change in lane maker position e.g. there is a turn off or filter lane
--- There has been too big a gap in the lane marking
- The lane marker scan fails
-- Rapid change in lane marker color
-- Too little contrast between the marker and roadway colors
-- Obscuration of the lane marker e.g., by another car
-- Algorithm coding error
-- Lane marker is missing
-- Mismatch between the color histogram from earlier scans and current marker color

Figure 5.2 The subtree for failure in lane detection

A more severe failure mode for the lane following algorithm is that of finding a false lane:

>> Vehicle leaves the road
- erroneous lane boundary geometry communicated
-- Communication error
-- Erroneous geometry from credibility checking
--- &
---- Credibility criteria too permissive
---- Erroneous lane boundary detected
----- Old or incorrect lane boundary marking
----- Color histogram allows too wide a color band
-----Image does not have sufficient contrast
----- Algorithm coding error
-- &
--- Lane detection fails, one or more lanes not detected >>
--- Erroneous lane prediction

Figure 5.3 Subtree for erroneous lane geometry calculation output

>> Too little image contrast for lane marker color
- Low lighting level
- Wet road pavement
- Worn lane marking
- Light colored roadway
-- Gravel on tarmac road
-- Concrete road

Figure 5.4 Subtree for causes of too little contrast in lane marker color

The fact that parameters such as lighting, roadway color etc. are continuous means that the fault tree is actually a hybrid one, and the possibility of failure lies in a continuous domain. If only one variable at a time is considered, the events in the tree can be discretized. A lighting level can be determined beyond which the algorithm simply does not work. There are many cases where two or more variable deviations are important at the same time, e.g., poor lighting together with worn or dirty lane markers. Failure will then lie within a two-dimensional domain. Figure 5.xx shows such a two-dimensional failure domain derived from systematic testing of the above algorithm.

When robustness is assessed by testing, the values determined depend very much on the test set used. A test set which does not contain a representative set of values may generate an over-optimistic result. Again, careful failure cause analysis alongside the testing can give more reliable results.

The probabilities of failure for the events in the fault tree are then the probabilities of occurrence of conditions outside the envelope of robust algorithm performance.

5.2 The sliding window algorithm for object detection

One algorithm for object detection, such as vehicle detection, is the sliding window algorithm. This creates a rectangular window, places the window at a range of positions on an image and then inspects the sub-image in the window to determine whether it represents and object. The sliding window algorithm is widely used, even for methods which use neural networks for object classification.

5.3 Failure probabilities and algorithm robustness

Events in the algorithm fault trees such as “image does not have sufficient contrast” or “credibility criteria set too tight” involve continuous variables. The robustness of these algorithms must be derived from testing, and further elaboration of the tests should be based on more detailed fault tree analysis. “Too little contrast” for example, can have many causes.

6. Neural networks in the design of AV controllers

Neural networks (NNs) have shown their remarkable performance of perception in their application in autonomous vehicles. However, NNs are intrinsically vulnerable to perturbations, such as occurrences outside of the training sets, scene noise, instrument noise, image translation and rotation or intentionally added small changes to the original image (called adversarial perturbations). Incorrect conclusions from the perception systems (e.g., missing objects, wrong classification, and traffic signs misdetection or misreading) have proved to be a major cause of disengagement incidents in autonomous vehicles (AVs).

In this chapter the failure modes of perturbations in the NN-based perception methods are analyzed (J. Zhang et al., 2021).

6.1 Neural networks in the AV controller design

Some AV control functions, such as distance sensing are best carried out by radar or lidar which inherently measure distance. While working algorithms exist for recognising objects in 3D using camera's, they are quite complex, and even if working, are difficult to validate for a full range of possible inputs. In other cases, neural networks and vision algorithms compete more or less directly. An example is the recognition of other vehicles on the road, where good neural network based designs are available, and distance can be inferred from location in the field of view and cues such as road edge recognition. In these cases a cooperative approach using two or more types of sensor (sensor fusion) can provide a more robust sensing. For still other functions neural networks provide better solutions. In the present design neural networks together with lidar are used for other vehicle detection and classification, with lidar to measure distance, and neural networks alone are used for detection of humans, animals, cyclists, roadworks and street signs.

6.2 NN types

Convolutional neural networks (CNNs) , recurrent neural networks (RNNs) , and deep reinforcement learning (DRL) are the three most common deep learning methodologies used in AVs. CNNs are suitable for processing spatial information, such as images, recurrent neural networks. RNNs are suitable for temporal sequence data, such as text, or video streams, and DRL are for path planning (for example learning driving trajectories)

Computer vision and deep learning are two major approaches to autonomous vehicle (AV) control. Traditional computer vision techniques can play an important role in lane detection and redundant object detection at moderate distance, but are unlikely to meet the robustness requirements for handling human-level tasks such as distinguishing between different metal objects or unexpected obstacles. Deep learning approach are able to learn meaningful road features from raw input data automatically and output driving actions. Traditional computer vision techniques can also be a useful supplement to NN techniques.

Nowadays, Convolutional Neural Networks(CNNs) are the most widely adopted deep learning model for AV perception (Grigorescu et al., 2020). The perception algorithms are the most critical module to detect objects and make image classification. Any incorrect conclusions from the perception algorithms, such as missing objects, wrong classification, and traffic signs mis-detection, will lead to potentially fatal incidents.

6.3 Failure and hazard analysis of neural network as AV controller components

Neural networks are notoriously difficult to analyse. Kalra and Paddock of Rand Corporation assessed that to demonstrate with 95% confidence that their failure rate is 20% better than the human driver failure rate would require 11 billion miles of on road driving, or about 500 billion vehicle years (Kalra & Paddock, 2016). This level of testing is impractical, and it is therefore desirable to analyse safety in the same way that other rare hazards are analyses, that is by risk analysis based on component reliabilities and by assessment of defense in depth, for example using fault tree analysis.

This does not mean that on-road testing would not be needed. The risk analyses would need to be validated since many hazard identification methods provide far from complete basis for risk analyses(Taylor, 2012). On-road testing is an evidence-based way of performing this validation. The risk analysis provides a way of amplifying the value of on-road testing, allowing near miss and partial failure cases to be included in the evidence base while providing a framework for generalization of test results and for assessment of such less serious incidents.

A more difficult problem is that of randomness in the design of neural networks. When reverse analysis is performed on most neural networks which have been trained with a given set of test images in order to determine the features on which results are based, the features which are recognised seem to be distributed in inexplicable ways among the network layers. This can be understood by considering the case of a well-structured neural network in which each layer carries out a specific recognition function, for example recognition of lines in the first layer, connected line patterns in the next, and representations in the third. If a new training example is introduced which provides a new kind of discrimination, for example a new stop sign, the training algorithm could provide the discrimination by taking any of the new features of the image and code for it in any of the network layers. An example of this is a set of traffic signs used as the training basis for a network. It worked well in development laboratory tests, but poorly in the field. On analysis it was found that the network was responding more strongly to the background than to the traffic signs.

While it seems that analysis in detail will be possible for some future neural network types, such as capsule networks, and it is known to be possible for very simple networks, this is not possible at present. For realistic AV applications. Nor is it necessary. Provided that the functional requirement placed on a neural network is that of a simple function, such as recognition of a stop sign, the neural network can be considered as a black box. The failure modes can be defined as failure to identify an image, incorrect classification of an image,

or in some cases incorrect estimation of an image parameter. The neural network will have a certain correct performance set and a certain level of robustness against image imperfections or distortions. The probability of failure of the network is then the probability of the observed image lying in a domain which is outside the network's capability, or is in a domain for which the network is not robust. The hazard analysis can then be completed using standard methods to determine the possible causes of the inputs lying outside the networks reliable domain. This is the approach taken here. It means that emphasis is placed on developing robustness measures for neural networks against different types of threats.
For these reasons, the approach taken here is to perform functional failure analysis for neural network components of the AV control; to determine the robustness and performance limits of the NNs; to determine the conditions under which the performance limits for the NN would be challenged by the operating environment; and to determine the degree to which additional robustness enhancements and other safety measures could compensate for any deficiencies in performance for the NNs. A generic template for incorporation of NN failure into a fault tree analysis is shown in Figure 6.1.

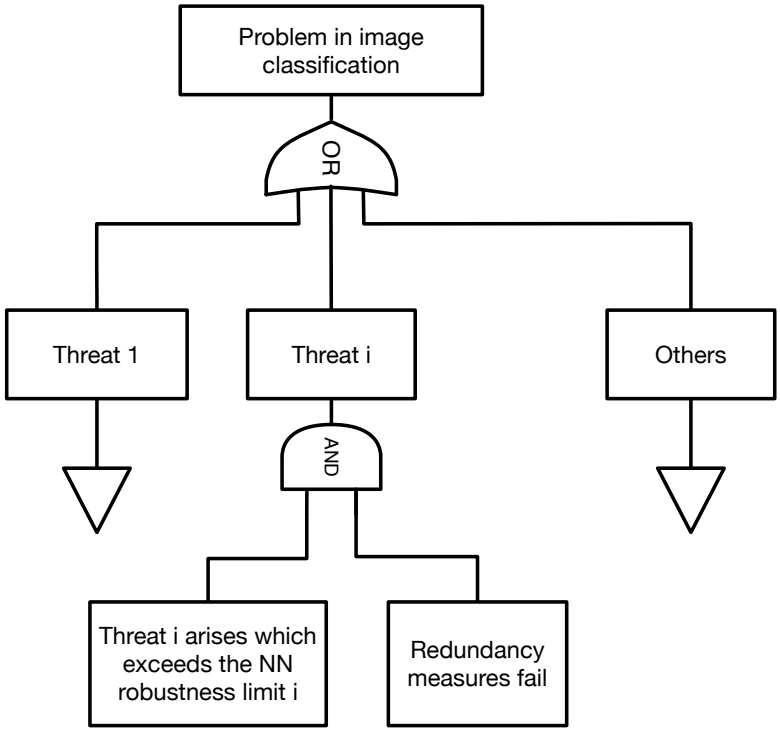


Figure 6.1 Schema for a NN component failure for different threats to NN performance

Functional failures of the NN can then be incorporated into fault trees in the form of multiple subtrees in an OR relationship. The probability of failure of the NN in any subtree is then:

$$P_{functional\ failure\ i} = P_{robutsness\ limit\ i\ exceeded} \times P_{Redundancy\ measures\ fail}$$

The events for mechanical and electrical components in the fault tree are therefore functional failures identified by failure analysis, with probabilities determined by testing

or by field observations; the functional failure probabilities for the NNs are determined by robustness testing and by determining the probability of threats exceeding the robustness threshold. For example, the treat to performance arising from low illumination levels can be determined by driving representative routes at different times and weather conditions and measuring and logging the illumination brightness levels.

6.4 Safety threats to NNs

There is a wide range of situations which can affect the performance of a neural network for AV control. The following list was identified by morphological brainstorming:

- Fundamental functional omissions (such as lack of training to recognize road diversion signs, or failure to recognize vehicles crossing a roadway due to lack of trajectory prediction)
- Sensitivity to ambient conditions especially low lighting
- Sensitivity to low contrast conditions
- Sensitivity to misleading patterns (such as camouflage) or to textures
- Obscuration, intended objects hidden behind others or behind a blind curve or behind vegetation
- Obscuration by snow, blown sand, frost or ice
- Reduction in visibility due to fog, snow, sandstorm
- Inadequate training set
- Poor separability of different object types due to similar feature sets
- Interference with well-trained recognition by extensions to the training set
- Orientation of the objects to be recognized ("pose")
- Unusual elevation of objects to be recognized (such as lane markings on a transition to a steep hill)
- Road reflectance, lights reflected from wet roads
- Strong back light, (e.g., driving into a sunset)
- Mirage effects in reflections from roadways
- Shadows

Further threats were identified from the validation studies, that is by reviewing accident reports:

- Unstable object recognition (with consequent erroneous or absence of emergency response)
- Obscuration by leading vehicle
- Vehicles crossing the roadway (vehicle not in the detection field or failure to measure velocity)

An important impact factor for NN hazards is the selection of the training set. Any omissions of an important phenomenon in the training set will result in a system which fails to recognize critical cases. This results in the strategy of massive training sets. Waymo, for example trains its vision systems for autonomous vehicles with millions of real traffic scenarios and billions of simulated scenarios (Schwall et al., 2020). However, meeting new phenomena can still lead to accidents.

The speed and the distance at which recognition can be made is also a problem for NNs. In case history 4 in chapter 9, a semi-autonomous vehicle was following another, but could not respond quickly enough when the leading car diverted around a parked fire truck, and the autopilot could not identify and warn about the obstacle fast enough.

A straightforward solution is to improve the vision system by data augmentation, sensor fusion, etc. Hendrycks et al. (Hendrycks & Dietterich, 2019) evaluated NN robustness to common corruptions and perturbations, such as Gaussian noise, motion blur, and snow. They found that as accuracy improvement of NN architectures, for instance, from AlexNet to ResNet, corruption robustness has no significant changes. All tested NN models are surprisingly vulnerable to common perturbations. Zhong et al. (Zhong et al., 2020) reported robustness of thirteen image classifiers and three object detectors to five real-world perturbations, i.e., luminance, spatial transformation, blur, corruption, and weather. Based on their results, some models outperform others for a particular perturbation, and a more complex NN architecture does not necessarily lead to a more robust model. Their results also showed that object detectors are more robust than image classifiers across various real-world perturbations.

Obscuration by another vehicle is a one of the most direct threats to recognition of traffic signs or other objects. Obscuration will occur if the vehicle is following a large truck and depends on the lines of sight. The distance between vehicles should preferably be sufficient to allow a line of sight, and recognition at a distance is preferable. The actual performance of the neural network at a distance is therefore important. Failure to recognise a stop sign will not necessarily cause an accident. The navigation rules implemented by the controller would ensure either a very low speed or a stop at a T junction. Also the electronic map should indicate a full stop location.

6.5 Security threats to NNs

In an adversarial context, threats to the neural network could arise from:

- Training data poisoning
- NN model attack
- Adversarial example
- Physical adversarial attack
- Sensor sabotage

Training data poisoning refers to deliberately introduce false data during the training process. NN model attack takes advantage of the model flaws to fool the system. An adversarial example is small changes intentionally added to the original input that are invisible to human eyes. There is a long history of work on understanding, detecting, mitigating impact, and increasing the robustness of CNNs through using adversarial examples (J. Zhang & Li, 2020). Physical adversarial attack aims to fool NN models by creating perturbations on physical objects. Sensor sabotage can be conducted by using spotlights to blind cameras or laser-targeting of cameras. In this study, we focus on the practical consequences of adversarial examples on the design of AV perception models. Typical examples of such attacks are as follows.

- Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014). The idea of the FGSM attack is to perturb the image in the sign direction of the gradient of the loss function with respect to the input pixels. Goodfellow et al. hypothesized that the adversarial examples are caused by neural networks being too linear, which makes them vulnerable to linear distortions.

- Projected Gradient Descent (PGD). The PGD attack (Madry et al., 2017) is an iterative method in which, after each iteration, the perturbation is projected on an lp-norm of the specified radius (in addition to clipping the values of the adversarial sample so that it lies in the permitted data range).
- Jacobian-based Saliency Map Attack (JSMA). The JSMA attack (Papernot, Mcdaniel, et al., 2016) is a family of adversarial attack methods in which a few pixels in a given image are saturated to their maximum or minimum values to fool CNN models. JSMA attack can cause the model to misclassify the resulting adversarial image as a specified erroneous target class.
- DeepFool (DF). The idea of a DF attack (Moosavi-Dezfooli et al., 2016) is to make the normal samples cross the decision boundary with minimal perturbations. DF attack is a simple and efficient method of computing adversarial perturbation and is also a method to evaluate the robustness of CNN models.
- C&W attacks. Carlini and Wagner (Carlini & Wagner, 2017) created their attacks as a follow-up to the defensive distillation (Papernot, McDaniel, et al., 2016), which was a proposed defense against adversarial examples. They also show that their attacks are able to create adversarial examples with less perturbation than other attacks such as FGSM, BIM, DF, and JSMA.

Evaluating the security threats to NNs is a safety consideration, and adversarial examples can further be used to improve the NN robustness.

6.6 NN Robustness measures

Robustness metrics can be developed to determine the functional correct performance range of NNs during testing.

Most of the previous works propose accuracy-based metrics to measure NN robustness, i.e., the accuracy (fraction of intended targets recognized) of the NN when inputs are perturbed (Hendrycks & Dietterich, 2019; Zhong et al., 2020). In an adversarial setting, the minimum perturbation distance (i.e., size of deviation for a loss of function) robustness, i.e., the accuracy (fraction of intended targets recognized) and adversarial accuracy (i.e., the accuracy of the model when an attack takes place) are two common metrics to evaluate NN robustness (Moosavi-Dezfooli et al., 2015; H. Zhang et al., 2019).

The AV we analyse in this study is a relatively simple AV, but the perception module of our testing car has over 50 NN's and vision algorithms for different purposes and different navigation situations. For each of these there are tens of potential disturbances which will affect performance, most being continuous factors rather than discrete yes/no influences. Each of these, and in many cases combinations of these, require robustness tests. Each test can involve hundreds, or even thousands of test cases in order to obtain a stable measure of robustness. Laboratory testing is used for robustness determination because it seems doubtful that on-road testing could ever generate sufficient cases to fully explore the space of potential failures. Laboratory testing to support hazard identification has been found practicable because the components can be set up and tested automatically.

6.7 The need for hybrid fault trees

One of the problems of the "threat versus robustness" approach to hazard identification and fault tree construction, as described above, is that NN performance failure often depends on small deviations in several parameters. These deviation cannot be described by simple threshold criteria, there will be a continuous relationship between the parameters and the

degree of correct functional performance. Event probabilities in the fault trees need to be replaced by probability density functions (pdfs). In the fault tree fragment in Figure 6.1, this means that pdfs are needed for the degrees of threat, for the robustness levels, and in some cases for the success of robustness enhancement measures.

As an example, one of the possible hazardous events triggered by decision made by the NN is "Wrong classification of a road sign". This event can occur because of inadequate robustness of the NN, which in turn can be caused by naturally or intentionally perturbed inputs. Robustness can be measured, as above by the probability of correct image classification (i.e., prediction accuracy) given perturbed inputs as a function for example the lighting level or the contrast in the image. These probability density functions do not achieve unity. In the case that there are two variables that influence whether the sign is classified correctly, the performance can be represented as in Figure 6.2.

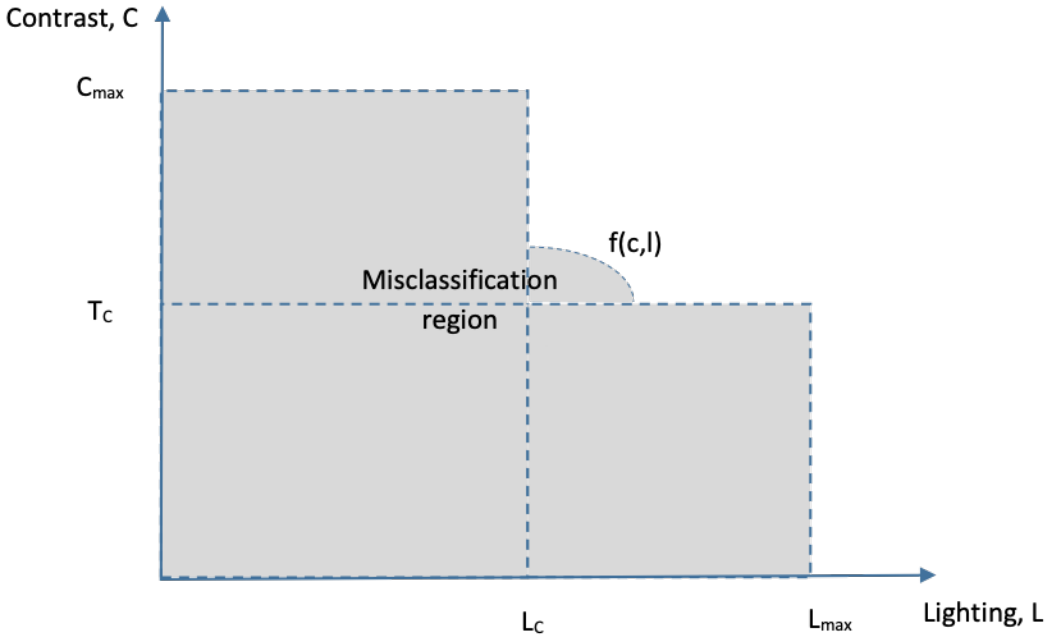


Figure 6.2 Misclassification region(Conceptual)

Here there are two input variables for the NN. One is contrast intensity, C, and the other is light intensity (i.e., brightness), L. If TC represents the lower limit for C, below which the sign cannot be classified correctly, we can define the event EC = {EC : c < TC } that is "too low contrast to recognise correctly". Similarly, EL = {EL : l < TL} is the event "too low lighting to recognise correctly". The third misclassification event is de-
fined by the following condition: ELC = {ELC : (l, c) < f(c, l), c > TC , l > TL}. This should be understood as follows: while contrast and lighting both lie in the correct classification region, their combination may belong to the misclassification region. The border dividing the two regions is determined by function f(C, l).

Usually, this type of event takes place when variables (parameters) lie in the vicinity of the border points. That is to say, the effect of small deviations results in a failure.
The region of misclassification can formally be written as follows:

$\Omega = \{(c < TC) \cup (l < TL) \cup ((l, c) < f(c, l), c > TC, l > TL)\}$
As soon as the misclassification events are determined, a simple fault sub-tree can be constructed as in figure 6.2 above.

Given C and L are independent random variables and their probability density functions are known, that is $f_C(x)$ and $f_L(y)$ the probability of misclassification P_M can be calculated:
$P_M = \iint_\Omega f_C(x)\, f_I(y)\, dx\, dy$

As soon as the misclassification events are determined, a simple fault sub-tree can be constructed (see Figure 6.3).

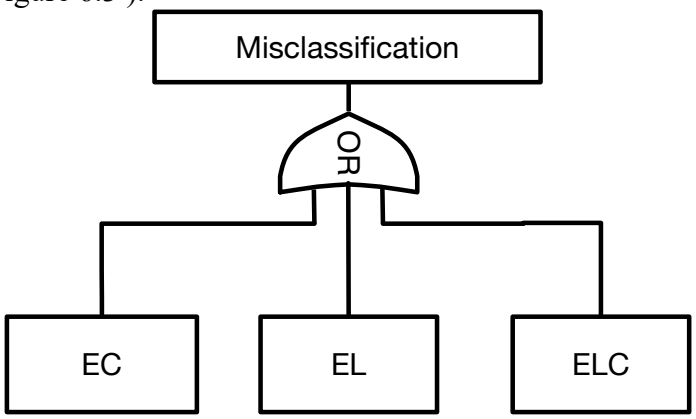


Figure 6.3 A simple fault sub-tree for misclassification inclusion individual threats EC and EL and interacting threats

6.8 Robustness enhancements

Data augmentation and increasing model complexity are commonly used approach for improving robustness. However, robustness improvement is not uniform across perturbation types. For instance, increasing performance in the presence of Gaussian noise may cause reduced performance on other perturbations (Hendrycks & Dietterich, 2019) . In Table 6.1, we identify robustness enhancements to perturbations based on perturbation types. We also map these robustness enhancements into appropriate safety strategies, i.e., inherently safe design, fail safe design, and safety margins on component (Varshney, 2017). Inherently safe design aims to exclude potential hazard from the system. Fail safe design is to keep system in a safe state at the time of failure. Safety margins on component is to reserve an additive space for achieving safety.

Achieving NN robustness is a challenging goal. Yin et al. found that both Gaussian data augmentation and adversarial training improve robustness to corruptions in the high frequency domain while reducing robustness to corruption in the low frequency (Yin et al., 2019). An adversarial defense method called \cite{kannan2018adversarial} adversarial

logit pairing (ALP) (Kannan et al., 2018) was proved to be less valuable against adversarial attack (Engstrom et al., 2018) but can remarkably enhance common perturbation robustness (Hendrycks & Dietterich, 2019). The interplay between adversarial robustness and common perturbation robustness should be better investigated because understanding when and why trade-offs caused between different data augmentation is a crucial step towards mitigating them.

Table 6.1 Robustness enhancements to perturbations

Perturbation type	Method to improve the NN robustness against perturbation	Safety strategy
Natural perturbation	Multiscale networks	Inherently Safe Design
	Feature aggregating	Inherently Safe Design
	Adversarial Logit Pairing	Inherently Safe Design
	Run-time out-of-distribution detection	Fail-safe design
	Histogram equalization	Safety Margin
Adversarial perturbation	Adversarial training	Inherently Safe Design
	Input transformation	Inherently Safe Design
	Randomized smoothing	Fail safe system design
	Adversarial detection	Fail-safe design

6.9 An example of robustness assessment – traffic sign recognition

To demonstrate the influence of threat perturbations and its combination, we trained a 5-layer-CNN with the German Traffic Sign Recognition Benchmark (GTSRB) dataset for the traffic sign classification (Stallkamp et al., 2012). The GTSRB dataset has 43 different traffic signs under various sizes, lighting conditions, and the images are very similar to real-life data. The prediction accuracy for clean test image (i.e., taken directly from the dataset) is 98.97% after training to a stable level, and avoiding overtraining.

We adopt the algorithm from Zhong et al. (2020) to emulate the deviation of brightness and contrast, and algorithm from (Goodfellow et al., 2014) to implement FGSM (Fast Gradient Sign Method) attack. Figure 6.4 presents (a) a set of misclassified images with brightness=0.8, the prediction accuracy dropped to 84.8%, (b) brightness=0.6, FGSM attack with strength=0.2, the prediction accuracy dropped to 18.76%.

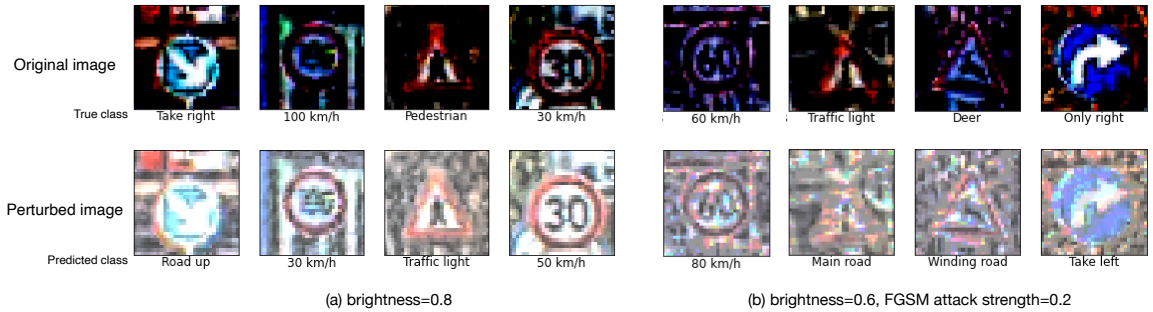


Figure 6.4 Examples of misclassified traffic signs when brightness=0.8

1) Brightness X' = Clip(X + l), where X is the original test image, l is a constant value to be added, X' is the resulted new image, Clip is a function to make sure X' is in a valid range of [0,255] or [0,1].
2) Contrast Reduction X' = Clip((1 - C) · X + c · C), where X is the original test image, C is the contrast level, c is a constant factor.

In this experiment, we set prediction accuracy as 90% as the acceptance level of model robustness. Instead of showing the case of low brightness/ contrast, we test the influence of increasing brightness and contrast reduction due to the low brightness/contrast nature of GTSRB dataset.

Figure 6.5 shows prediction accuracy curves corresponding to (a) brightness variations, (b) contrast variations. It shows that the upper limit for brightness increasing is 0.66 in Fig.5 (a), and the upper limit for contrast reduction is 0.54 in Fig.5 (b). Then we test the combination of brightness and contrast reduction. Brightness level is set from 0.01 to 1, and contrast reduction is from 0.01 to 1 respectively. This experiment is to show how the small deviation of contrast and brightness affect prediction accuracy.

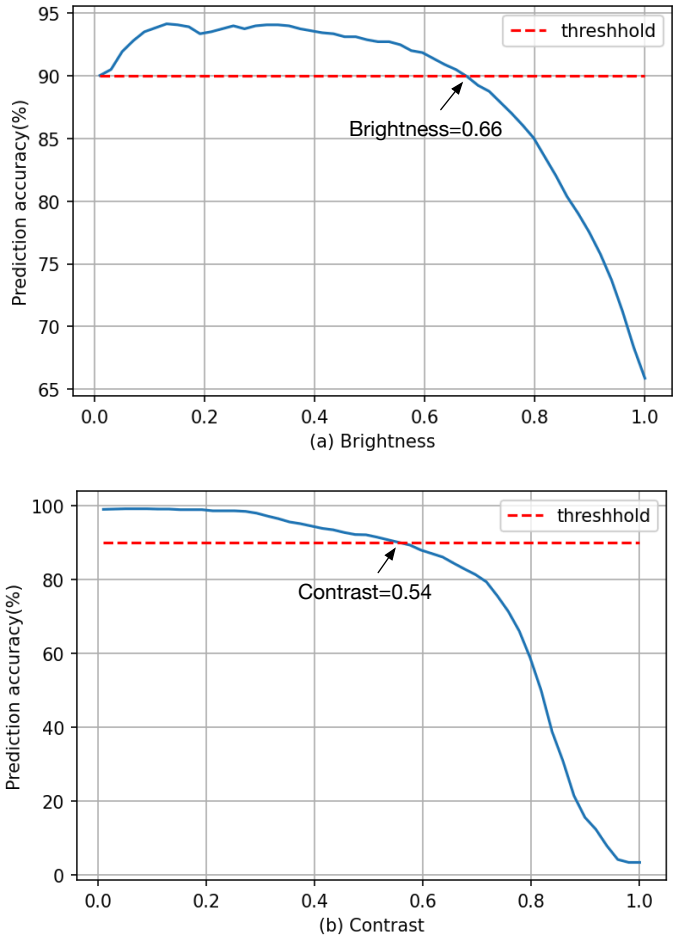


Figure 6.5 Examples of prediction accuracy curves when brightness and contrast vary.

Then we test the combination of brightness and contrast reduction. The brightness level is set from 0.01 to 1, and contrast reduction is from 0.01 to 1, respectively. This experiment is to show how the small deviation of contrast and brightness affect prediction accuracy. In ,the values of prediction accuracy are represented as colors. The lighter the color, the higher the prediction accuracy. It shows that even brightness level and contrast reduction are not exceeded their upper limits (i.e., in the correct classification region). Their combination can fall into the misclassification region (i.e., prediction accuracy is lower than 90%).

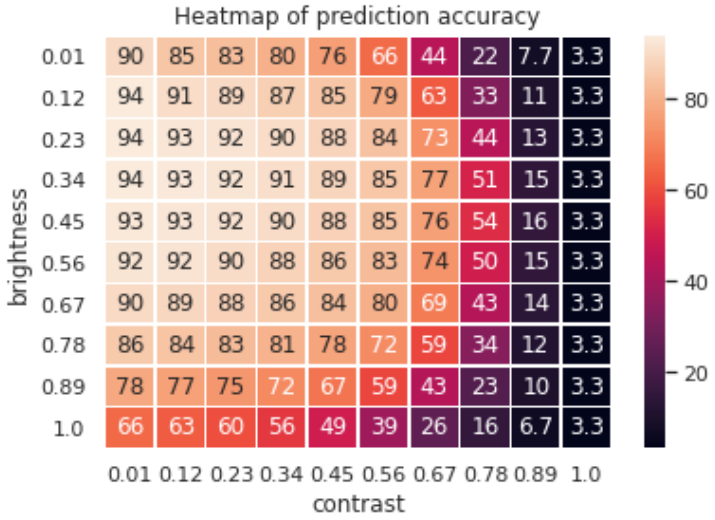


Figure 6.6 prediction accuracy matrix when small deviation of brightness and contrast in combination

6.10 Procedure of extending FTA for NNs

From this study, it became clear that detailed hazard identification can be made for AVs, including both hardware and NN components. The procedure is:

- Complete the overall high-level hazard identification using an FTA approach.
- Identify the functional failures of the NNs which contribute to the overall FTA.
- Identify the challenges which can cause the NN functional failure, e.g., using the checklist in Section NN Safety threats to the NNs and Security threats to the NNs
- Determine the robustness of the NNs when challenged by both perturbations of single parameters or by the combination of parameter perturbations via testing NN performance and making heatmap as in Figure 6.6.
- Determine the probability of the occurrence of parameter perturbations.
- Incorporate the contribution of NNs to the FTA using the templates given in Figure 6.1 and Figure 6.3.

7. Response to hazardous situations

Current AV's have an "operator" able to take over driving if the autonomous controller detects a situation with which it cannot deal. There is an inherent conflict in this approach. As the autonomous controllers become increasingly competent, the operator has less to do. It is well known that humans have difficulty in retaining concentration when trying to perform a task in which nothing significant happens. Even if observation is continuous the operator is not "tuned in" to the driving task and will have a slower response to an emergency situation than a human driver actually controlling a vehicle, even if there are good visual and audible warnings. It is known from experience with aircraft autopilots that it takes several second to respond even to a high-quality alarm. Many of the accidents noted in Ch. 9 involved emergencies which were detected by the autonomous controller but where the operator was unable to react quickly enough.

In our experimental design, the vehicle is fully autonomous, and there is no "operator". The AV must therefore be able to respond to a full range of emergencies.
When a hazardous situation arises for a fully autonomous AV, the controller should detect and assess the situation. This means that there must be some ay to assess the risk, dynamically, or at least to classify the risks. If there is an imminent probability of a crash, for example, avoidance or minimization of consequences is critical.

The hazard identification is therefore an important input to the design of the emergency response plans, and it becomes extremely important that the hazard identification is complete and accurate.

When a hazardous situation is identified by a controller, it is not sufficient simply to sop the vehicle. If this is done in fast traffic, the act of stopping itself can present a higher level of risk than the initial hazardous situation. Strategies and plans are needed for safe recovery, at least to the point where the AV can find an open lane which is safe, or can navigate to a stop on a hard shoulder.

These plans can be more complex and can present a higher range of hazards than the initial hazardous situation. If for example, it is detected that a crash into ta car in front is likely, the plan needs to take into account whether there are other vehicles in the inner and outer lane, whether the car in front is simply going slow or braking hard and whether the car in front is losing control and whether avoidance is at all possible. The plan also needs to take into account the own vehicle speed, braking system performance and roadway conditions. It also needs to take into account other traffic, and the likely response of other traffic. While the emergency plans may be stylized and may be relatively simple in themselves, the hazard identification for these plans is actually more complicated and dynamic that that for the AV performance under normal conditions.

Thorough investigation of these issues is beyond the scope of the present report, but an example of one emergency situation is given later in this chapter.

7.1 Response to AV failures

All vehicles can fail, the engine, the steering, the braking and the wheels, for example. Modern vehicles have a large degree of instrumentation both to detect and diagnose failure, and far more detection is possible, as evidenced by the extremely large level of monitoring on Formula One vehicles, which have hundreds of instruments to monitor component performance.

As in the case of potential accident situations, there needs to be a policy, strategy and plans for vehicle failure response. The failure analysis for the vehicle should be a basis for the design of this function. Also, as in the case of imminent accidents, the necessary emergency response can be very complicated, depending as it does not only on the possible AV failures, but also on the wide range of road and traffic situations.

AV failures and the resulting incidents and accidents are very dynamic. The hazard analysis then involves a combination of possible events, including adverse conditions and failures, and vehicle dynamics analysis. As with hybrid fault trees, hybrid cause consequence diagrams will involve investigating ranges of dynamics parameters such as brake performance and road friction.

7.2 An example – tire blowout

Tire blowout is a serious failure event in an AV, affecting steering stability and braking. The sequence of actions and their timing are very important both for the response and for any resultant accident effect. Cause consequence analysis is therefore used for the analysis. In the following a cause consequence diagram is given for tyre blowout response. Two main separate chains of events are described, one the vehicle response itself, the other the emergency response in the AV.

One specific response strategy is chosen for the case (if the failure is detected in time), that is to set the engine into idle and coast to a standstill while correcting steering as far as possible. This is a very dynamic situation, and simulation is required to determine the dynamics of the vehicle and controllability.

The cause consequence diagram for tire blowout is shown in Figure 7.1.

Tire blowout
- ? blowout detected << 3 [detected by tyre pressure loss or by changes in steering "feel"]
-- Y blowout detected >> 1
-- N blowout not detected
--- ? Which tyre affected
---- 1 Rear tyre affected >> 2 [this branch of the CCD is not evaluated in this short example]
---- 2 Front tyre affected
----- (delay 0.5 sec) Steering affected [typical steering response delay]
------ ? steering and speed corrected << 4
------ Y
------- ? ~ AV is able to stay in lane [behavioural domain for successful control from the simulation]
------- Y Safe response [A simplification ignoring any following car problems and moving off the road]
------- N AV leaves the lane
-------- ? Vehicle in adjacent lane and unable to stop
--------- Y Crash
--------- N AV moves into another lane
---------- ? ~ Vehicle leaves the road
---------- N No crash
---------- Y Crash

Figure 7.1 The cause consequence diagram for the tire blowout accidents

1 >> blowout detected
- Blowout signal communicated to the navigation controller
-- Tyre burst response for the wheel selected [selection assumed to be made correctly]
--- ~ Steerability evaluated {steerability = f(AV speed, steering angle)}
---- ? AV is in the steerable domain
----- Y Vehicle is controllable
------ ? steering system works correctly
------- Y steering under control >> 4
------ N No steerability

Figure 7.2 The cause consequence diagram for blowout detection response

7.3 The range of possible emergency situations

It is useful to itemize the range of possible emergencies in order to be properly able to carry out all the analyses and to provide a basis for the emergency response controller design (which is part of the navigation control.

The first important aspect of the emergency analysis is to determine the external conditions for the response: These conditions include:

- Normal road conditions
- Road conditions with poor lane marking
- Low light conditions
- Poor visibility due to fog, snow, rain or smoke
- Obscuration of signage, missing signage
- Erroneous signage
- Poor road surface with ice, snow, aquaplaning conditions, poor adhesion
- Contamination on the road, such as spilled liquid
- Number of lanes
- Barriers between opposing road directions
- Hard shoulder, safety lane or none
- Roadside barrier
- Roadside ditch
- Roadside trees, light posts etc.
- Obscuration of the road ahead (by trees, vehicles etc.)

Also important are the driving navigation situations

- Normal driving on a straight road
- Driving on a curved road sections
- Sharp bend in the road or multiple curves
- Accelerating
- Slowing
- Approaching an obstruction (single or multiple lanes)
- Queueing and traffic jam negotiation
- Overtaking
- Approaching a junction
- Turning at a junction (right and left)
- Filter out lane(s)
- Filter in lanes
- Negotiating a roundabout
- Parallel parking
- Echelon parking
- Reverse parking

Hazardous situations include:

- Cyclists in an inner lane
- Cyclists in a central lane
- Motorcyclists
- Pedestrians crossing
- Animals on the road

Emergency events external to the AV include:

- Oncoming vehicles in the AV's lane
- Oncoming vehicles in an overtaking lane
- Vehicle ahead suddenly braking
- Vehicle pulling out from a parking position

- Vehicle pulling out from a side road or a site exit
- Vehicle crossing at a junction without right of way
- Vehicle oversteering, understeering, skidding, spinning or rolling
- Person suddenly crossing the road
- Animal suddenly moving onto the road

Emergency events internal to the AV include:

- Tire blowout
- Loss of adhesion
- Brake failure
- Brake poor performance
- Steering failure jammed
- Steering failure decoupled
- Steering failure backlash
- Hydraulics failure
- Hydraulics poor performance (e.g., air in system)
- Structural failure
- Drive seizing
- Unable to shift gear
- Engine failure
- Engine poor performance
- Unable to slow engine
- Camera failure
- Lidar failure
- Gyroscope failure

All of these possibilities lead notionally to over 5000 different incident types. Although many can be ruled out as highly improbable, they still need analysis in order to demonstrate their improbability. Many of the cases can be "conflated", that is combined. The cause consequence method is useful for this because each diagram can describe many similar accident types or accidents which have a similar initial event sequence. The number of cases is also be reduced by the fact that in many of the cases aspects such as road conditions, weather etc. do not affect some navigation types such as parking. The number of combinations of external conditions, vehicle steering, and navigation and emergency types is nevertheless very large.

8. A holistic view of safety and security analysis

One of the things which became apparent in carrying out the detailed hazards analysis as given in the appendix and summarized in the previous chapters is how much easier it is to make a holistic safety assessment with several sensors than it is when safety depends on just a single sensor. This works in two ways. One is redundancy, in that with two or more sensors to detect a hazardous condition, the reliability and robustness requirements of one sensor does not need to be so stringent. This is especially true if the sensors are based on different technologies. The strengths of one can cover the weaknesses of the other, and often, vice versa.

The other benefit of having two or more sensors with different technologies is that one can be used to improve the performance of the other. An example occurs when a camera sensor makes use of a history file to determine a lane, vehicle or other object recognition. Credibility checking can then be used to resolve uncertainties and to reject false positives or to bridge over false negatives. Use of other sensors allow for more opportunities for credibility checking.

There are some strategies in which even more close interaction between sensors can be used. An example is the use of probability determination for object recognition, where the probability of a false negative is determined as the product of two recognition results.

9. Validation of the analyses

9.1 Approach to validation

A prime requirement of any hazard identification method is that it should be repeatable, or that any deviations from repeatability can be detected and corrected, for this reason, parts of the analysis were carried out both manually and automatically, using generic functional failure models.
Performance of the different approaches was made by identifying the range of accident scenarios (cut sets) determined at each level of detail in the analyses and comparing the results from different methods.
The ultimate test of a hazard identification is to compare it with actual accident experience. This was carried out by comparing the fully automated fault tree analysis using generic component models with a set of accident case histories taken from the literature.

9.2 AV controller failure cases

9.2.1 Case 1 March 2018 Uber collision with a pedestrian, Tempe California

In March 2018, Uber suspended testing of its autonomous Volvo XC90 fleet after one of its cars struck and killed a pedestrian in Tempe, Arizona.
The Self-driving Uber that killed pedestrian reportedly didn't realize she was human.
NTSB carried out an investigation and found that the probable cause of the crash in Tempe, Arizona, was the failure of the vehicle operator to monitor the driving environment and the operation of the automated driving system because she was visually distracted throughout the trip by her personal cell phone.

Uber were found to have inadequate risk assessment procedures, ineffective oversight of the testing and lack of adequate measures for monitoring operators' complacency or reliance on the automation

The ADS detected the pedestrian 5.6 seconds before the impact. The software initially classified the obstruction as a vehicle then as an unknown object. The software failed to predict the person's path or to reduce speed. 1.2 seconds before impact the software transferred control to the operator, but the time available was insufficient to carry out emergency braking.

9.2.2 Case 2 Collision between a shuttle bus and a reversing tractor trailer, Las Vegas November 8, 2017

NTSB carried out an investigation for this case. At about 12:07 p.m. on Wednesday, November 8, 2017, a minor collision, between a truck-tractor combination vehicle, operated by a 48-year-old driver, and a 2017 Navya Arma autonomous shuttle, carrying 7 passengers and a 38-year-old attendant. The shuttle, manufactured by Navya and operated by Keolis North America, was on a 0.6-mile designated loop The truck-tractor pulling a 2010 Utility refrigerated trailer, was backing into an alley when it struck the shuttle. The collision caused

minor damage to the lower left front of the shuttle's body and a minor abrasion to the truck's tire.

The shuttle operated on a fully mapped designated route. The operator was not intended to monitor the driving and to take over in the case of hazard. NTSB determined that the primary cause of the accident was the tractor trailer driver's backing into the alley, expecting that the shuttle bus would stop.

No information is given in the report on the reason for the vehicle's failure to stop. It can though be concluded that the lidar object detection and avoidance system function failed

9.2.3 Case 3 Collision Between a Sport Utility Vehicle Operating With Partial Driving Automation and a Crash Attenuator March 23, 2018, Mountain View, California

A 2017 Tesla Model X P100D electric-powered sport utility vehicle operating with partial diving automation crashed into a crash attenuator.
The driver approached the interchange consisting of:

- A single left-exit high-occupancy vehicle (HOV) lane
- A single HOV lane, the driver's intended route),
- Three mainline lanes (red arrows), and
- Two right-exit conventional lanes

The autopilot was active and the cruise control was set to a cruise speed of 75 mph.
Between 10 and 6 seconds prior to the crash, the SUV was traveling between 64 and 66 mph and following another vehicle at a distance of about 83 feet. The Tesla driver had set the cruise control to position 1, which maintained the closest possible following distance behind the lead vehicle immediately ahead (time-based following distance of about 0.9 seconds).
When the Tesla was about 5.9 seconds and about 560 feet from the crash attenuator, Autosteer initiated a left steering input (5.6 degrees at the steering wheel) toward the neutral area of the protected space (gore). At the time of the steering movement, no driver-applied steering wheel torque was detected by Autosteer

When the SUV was about 3.9 seconds and 375 feet from the crash attenuator and fully inside the neutral area of the gore, the cruise control no longer detected a lead vehicle ahead and the car began accelerating from a speed of 61.9 mph to the preset cruise speed of 75 mph. The forward collision warning (FCW) system did not provide an alert and the automatic emergency braking (AEB) did not activate as the Tesla approached the crash attenuator.

The Tesla's Autopilot lane-keeping assist system steered the sport utility vehicle to the left into the neutral area of the gore, without providing an alert to the driver, due to limitations of the Tesla Autopilot vision system's processing software to accurately maintain the appropriate lane of travel. The Tesla's collision avoidance systems were not designed to, and did not, detect the crash attenuator at the end of the gore.

The failure of the autonomous vehicle control here seems to be that the vehicle navigation strategy controller was simply not defined for the situation.

9.2.4 Case 4 Crash into a stationary vehicle, Culver City, California January 2 2018

At about 8:40 a.m. on Monday, January 22, 2018, a 2014 Tesla Model S P85 car was traveling in the high-occupancy vehicle (HOV) lane in Culver City, California. The Tesla was behind another vehicle. A California Highway Patrol (CHP) vehicle was parked on the left shoulder of southbound highway and a Culver City Fire Department truck was parked diagonally across the southbound HOV lane. The emergency lights were active on both the CHP vehicle and the fire truck.

When the vehicle ahead of the Tesla changed lanes to the right to go around the fire truck, the Tesla remained in the HOV lane, accelerated in response to the traffic aware cruise control, and struck the rear of the fire truck at a recorded speed of about 31 mph.

At the time of the crash, the fire truck was unoccupied. The Tesla driver did not report any injuries. The car was equipped with advanced driver assistance systems (ADASs), including Autopilot. Based on the driver's statements and on performance data downloaded from the car after the crash, the Autopilot was engaged at the time of the collision."

The car was equipped with advanced driver assistance systems (ADASs), including Autopilot. Based on the driver's statements and on performance data downloaded from the car after the crash, the Autopilot was engaged at the time of the collision."

The National Transportation Safety Board determines that the probable cause of the Culver City, California, rear-end crash was the Tesla driver's lack of response to the stationary fire truck in his travel lane, due to inattention and overreliance on the vehicle's advanced driver assistance system.

During the last minutes prior to the crash the autopilot was continuously engaged but issued several hand-off alerts, with audible warning and visual alerts. The automatic emergency braking did not activate. The owners manual stated that the traffic aware cruise control could not detect all obstacles especially when travelling at speed and if a leading vehicle moves out of own cars path and there were a stationary obstacle hidden by the leading car.

This seems to be a straightforward case of the vehicle safety software inadequate performance, with reliance on a driver unaware of the hazard involved in relinquishing control.

9.2.5 Case 5, Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck Near Williston, Florida May 7, 2016

On Saturday, May 7, 2016, a 2015 Tesla Model S 70D car, traveling eastbound on US Highway 27A inLevy County, west of Williston, Florida, struck the right side of a refrigerated semitrailer towed by truck-tractor. At the time of the collision, the truck was

making a left turn from westbound US-27A across the two eastbound travel lanes. After the car struck the right side of the semitrailer, it crossed underneath the semitrailer and went off the right roadside at a shallow angle. The impact with the underside of the semitrailer sheared off the car's roof.

System performance data downloaded from the Tesla indicated that the driver was operating the car using features of its Autopilot suite: Traffic-Aware Cruise Control (TACC) and the Autosteer lane-keeping system.

"The National Transportation Safety Board determines that the probable cause of the Williston, Florida, crash was the truck driver's failure to yield the right of way to the car, combined with the car driver's inattention due to overreliance on vehicle automation, which resulted in the car driver's lack of reaction to the presence of the truck. Contributing to the car driver's overreliance on the vehicle automation was its operational design, which permitted his prolonged disengagement from the driving task and his use of the automation in ways inconsistent with guidance and warnings from the manufacturer."

The failure of the Tesla to brake automatically to seems to be that there was no function to predict or detect vehicles crossing the and especially vehicles not ceding the normal right of way.

9.2.6 Cases 6 to 24 Incidents from Waymo testing

Waymo, with a clear transparency policy in their research, published an exhaustive register of crash data . Waymo reported 18 low severity (ISO 26262 severity S1) actual collisions in 6,1 million miles of on-road testing, and also counter-factual simulations for those cases in which the Waymo driver disengaged, and the human driver avoided a crash.

The Waymo Driver did not have any events (actual or simulated) in this data that involved road departure, contact with the roadway environment/infrastructure or other fixed objects, or rollover. There were also no collisions (actual or simulated) in which the Waymo Driver struck a pedestrian or cyclist.

Cases are:

- There were three events (one actual, two simulated) in which the Waymo vehicle was struck by a pedestrian or cyclist. In each instance, the Waymo Driver decelerated and stopped, and a pedestrian or cyclist made contact with the right side of the stationary Waymo vehicle while the pedestrian or cyclist was traveling at low speeds.
- There were two reversing collisions involving the Waymo Driver, one actual and one simulated (both S0 severity). In both scenarios, the Waymo vehicle was stopped or traveling forward at low speed and the other vehicle was reversing at a speed of less than 3 mph at the moment of contact to the side of the Waymo vehicle.
- Same direction sideswipe collisions are a more common vehicle collision mode, and are typically low in severity. These events are typically experienced during lane changing or merging maneuvers. The Waymo Driver was involved in ten simulated same direction sideswipe collisions
- Other vehicle straight, Waymo vehicle changing lanes collisions occurred in two simulated cases two. In both of these simulations, the Waymo Driver made a lateral movement in front of a vehicle traveling straight in an adjacent lane. In one of the events, the following vehicle was traveling over

30 mph above the posted speed limit. The other event involved a vehicle that had entered early into a dedicated left turn lane that the Waymo Driver was attempting to merge into.

- Rear End collisions are the most common collision type in human-driven collisions (Table 1), though they contribute to only approximately 5% of the human-driven collision fatalities in the US and in the Waymo ODD. The Waymo Driver was involved in fourteen actual and two simulated rear end collisions. In all but one of these events, another vehicle struck the rear of the Waymo vehicle.
- Two actual collisions involved the Waymo vehicle being struck on the rear bumper while traveling straight at a constant speed at or below the speed limit. In one collision, the Waymo Driver had slowed to a constant speed in the course of traveling over a speed bump. In the other collision (Figure 4, Event C), the Waymo vehicle, traveling straight at the speed limit, was struck by a vehicle traveling 23 mph over the posted speed limit. Both collisions were of S1 severity, with airbag deployment occurring in the striking vehicle in the latter collision.
- The remaining rear end struck collision involved a deceleration to a near stop by the Waymo Driver while making a left turn in an intersection with a following vehicle that was traveling at a speed and following distance that did not allow for the following driver to successfully respond to the Waymo Driver's braking. The simulated collision impact was estimated to be 16 mph, and this event is categorized as S1 severity.
- A rear end striking simulated event involved a vehicle that swerved into the lane in front of the Waymo and braked hard immediately after cutting in despite lack of any obstruction ahead (consistent with antagonistic motive). The Waymo Driver was simulated to have achieved full braking in response to the other vehicle's braking but was simulated to contact the lead vehicle with a relative impact speed of 1 mph (S0 severity).
- Angled collisions, those that are typically seen at intersections and involve crossing or turning vehicles, account for approximately one quarter of all human-driven collisions and a similar fraction of the contribution to all human-driven fatalities. The Waymo Driver was involved in fourteen simulated and one actual angled collision. Angled events with the other vehicle not yielding to Waymo right-of-way This grouping consists of the events (ten simulated, one actual) involve the Waymo vehicle traveling straight in a designated lane at or below the speed limit. In all scenarios, the turning/crossing other vehicle either disregarded traffic controls or otherwise did not properly yield right-of-way.

Angled events occurred with Waymo vehicle crossing another vehicle's path The collisions involve four simulated collisions, where the Waymo Driver was making a right turn from a rightmost lane that was either splitting to an additional lane, or had been the result of two lanes merging to one. In each event, a passenger vehicle attempted to pass the Waymo vehicle on the right while the Waymo Driver was slowing to make the right turn with the right turn signal activated. In each case, the Waymo vehicle's trained operator disengaged, while in simulation the Waymo Driver turned, resulting in simulated collision.

9.3 Summary of the cases

Table 9.1 Summary of all the cases recorded including a classification of the causes.

Case	Date	AV type	Scenario	Cause	Comment
1	Mar 2018	Uber	collision with a pedestrian	Pedestrian detected but with unstable identification. OP distracted by telephone	Included as a hazard type, Ch.6

Case	Date	AV type	Scenario	Cause	Comment
2	Nov 2017	Shuttle bus	Collision between a shuttle bus and a reversing tractor trailer	Function for an emergency stop not implemented or failed	Included as a hazard type in Ch 4 and 6
3	Mar 2018	Tesla Sport Utility Vehicle Operating With Partial Driving Automation	Collision Between a Sport Utility Vehicle and crash attenuator	The attenuator was occluded by a leading vehicle. When revealed the forward collision warning (FCW) system did not provide an alert and the automatic emergency braking (AEB) did not activate approached the crash attenuator	Included as a hazard type, Ch.6
4	Jan 2010	Tesla car with advanced driver assistance systems (ADASs), including Autopilot	Crash into a stationary vehicle	Crash into a fire truck, detected late when the occluding lead vehicle overtook the truck. Insufficient time for emergency braking,	Added in Ch 6 after validation data collected
5	May 2016	Tesla With Automated Vehicle Control Systems	Collision Between a Car and a Tractor-Semitrailer	Traffic aware cruise control failed to recognize a truck crossing due to the being no functional capability for this prior to truck entering the road	Included as a hazard type in Ch 4 and 6
6		Waymo driver	Reversing collisions (one actual one simulated)	Other vehicle impact, with low consequences.	Not included in the analyses. This is a difficult class for an autonomous vehicle, but humans can often detect a careless reversing
7		Waymo driver	Same direction side swipe	Six simulated and one actual collision cause by other vehicle moving into AV's lane or into stationary AV	Included in the analyses
8		Waymo driver	Sideswipe with AV changing lane	Following car was travelling at high speed (simulated)	This case is important because of the parameter value. When changing lanes AV checks for vehicles coming from behind but not well enough

Case	Date	AV type	Scenario	Cause	Comment
9		Waymo driver	Sideswipe with AV changing lanes	involved a vehicle that had entered early into a dedicated left turn lane that the Waymo Driver was attempting to merge into	Included in the analysis. Needs a deeper study and modelling of accidents
10		Waymo driver	Head on crash (simulated)	Other vehicle driving the wrong way in AV's lane	Included in Ch 7 The emergency response for this needs deeper investigation
11		Waymo driver	Rear end collision	13 cases of other cars crashing into AV	These accidents are not failures of the AV. Does AV need defensive driving?
12		Waymo driver	Rear end collision AV travelling slowly	AV slowed for speed bump	These accidents are not failures of the AV. Does AV need defensive driving?
13		Waymo driver	Rear end collision	AV travelling straight at normal speed, another vehicle speeding	These accidents are not failures of the AV. Does AV need defensive driving?
14		Waymo driver	Rear end collision while turning	4 actual cases of rear end collision while AV turning or filtering right	These accidents are not failures of the AV. Does AV need defensive driving?
15		Waymo driver	Rear end collision	rear end collision involved a deceleration to a near stop by the AV while making a left turn in an intersection with a following vehicle that was traveling at a speed and following distance that did not allow for the following driver to successfully respond to the AV's braking	These accidents are not failures of the AV. Does AV need defensive driving?
16		Waymo driver	Rear end collision	Another vehicle failed to stop a t a full stop junction	These accidents are not failures of the AV. Does AV need defensive driving?
17		Waymo driver	Angled collision	5 simulated collisions by other vehicles which failed yield right of way	These accidents are not failures of the AV. Does AV need defensive driving?

Case	Date	AV type	Scenario	Cause	Comment
18		Waymo driver	Angled event with AV crossing another vehicle's path	4 simulated collisions, where the AV was making a right turn from a rightmost lane that was either splitting to an additional lane or had been the result of two lanes merging to one.	Included in the analysis. Strategy for right turn needs to be studied

9.4 Lessons learned from the fault tree mark-up

Although the number of failures cased is small a degree of validity is demonstrated in the appendix. The fault tree was constructed independently of case data collection and not be the same author as the data collection. The actual cases are marked onto the fault tree with the case number.

Just 14 cases from the dataset can be attributed to the failure or lack of function in the AV .Of these cases all but two were "predicted" by the analysis. The historical completeness is assessed as 12/14, or 86%. The most interesting feature of this study is the ones which were not predicted, and the reasons for failure to predict. The case involved the navigation strategy in responding to changing road situations, especially other traffic, where the navigation tactics are non-optimal due to the speed of the changes, and the dynamics of right turns when there are following vehicles. These, in the analysis, were the result of inadequate modelling of the transition between tactics. These lead to the following improvements in methodology:

- All changes of navigation strategy and all situations necessitating such changes should be analyzed using cause consequence analysis and parametric brainstorming
- Event sequences from the CCDs can be integrated into the overall system fault tree.

10. Conclusions

The analysis and the comparison with the validation cases showed that the navigation planning and mode transfer is a critical component with a difficult design requirement. The original design was not adequate, lacking smooth mode transfer, and emergency time threshold as a parameter for strategy choice. This is an area where cause consequence analysis and timing analysis based on this can support improved design. Such an improvement is given in the appendix B.

From the examples carried out it is clear that detailed hazard identification can be made for autonomous vehicles. This can partly be done using traditional hazard identification methods, but that extensions are needed including software fault tree analysis, sequential and dynamic fault trees and hybrid fault trees.

Determining the probabilities of failure and the risk from failures depends in part on identifying errors and weaknesses in the design, and in part on determining the performance limits of the AV controller components. Many of the potential accidents involve the demands on the controllers being outside the domains of robust controller performance. Determining the limits of robust performance requires testing, preferably at the component level rather than on-road entire system testing. The probabilities of failure and the resultant risk will in many cases be the probabilities of circumstances arising outside the robust performance domains of the components.

A further conclusion is that detailed hazard assessment can be an important aid in determining the scope of controller component testing.

The most effective method for hazard identification was found to be fault tree analysis. Cause consequence analysis was found to be the most effective method for analysis accident scenario dynamics, but for the overall result, CCA results were incorporated back into the fault tree. FMEA, software fault tree analysis and STPA were found to be useful supplementary analysis methods.

One of the key findings of the studies described here is that safety and security analysis becomes much easier when a holistic approach is taken. There are many potential cases where individual controller components will fail, but where accidents can be avoided by other components taking over. This is particularly an issue where there is the possibility of a crash, and where visibility conditions are poor. In these cases, lidar and radar provide less informative but more robust detection of hazards.

Safety in AVs is not assured by hazard detection alone. It is not safe for example, to simply stop the vehicle when a crash potential is detected in fast moving traffic. Policies, strategies, plans and algorithms for safe state recovery are needed. The next challenge for us is then to be able to carry out hazard identification and risk assessment on these recovery plans.

References

Alex Krizhevsky, Vinod Nair, & Geoffrey Hinton. (2010). *CIFAR-10 and CIFAR-100 datasets*. https://www.cs.toronto.edu/~kriz/cifar.html

Banerjee, S. S., Jha, S., Cyriac, J., Kalbarczyk, Z. T., & Iyer, R. K. (2018). Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data. *Proceedings - 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018*, 586–597. https://doi.org/10.1109/DSN.2018.00066

Carlini, N., & Wagner, D. (2017). Towards Evaluating the Robustness of Neural Networks. *Proceedings - IEEE Symposium on Security and Privacy*, 39–57. https://doi.org/10.1109/SP.2017.49

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining And Harnessing Adversarial Examples. In *arxiv.org*. https://github.com/lisa-lab/pylearn2/tree/master/pylearn2/scripts/

Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, *37*(3), 362–386. https://doi.org/10.1002/rob.21918

Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *7th International Conference on Learning Representations, ICLR 2019*.

Kalra, N., & Paddock, S. (2016). Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability? In *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* RAND Corporation. https://doi.org/10.7249/rr1478

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2323. https://doi.org/10.1109/5.726791

Leveson, N. G., & Thomas, J. P. (2018). *STPA Handbook*. 188.

Lin, C. T., & Wang, M. J. J. (1997). Hybrid fault tree analysis using fuzzy sets. *Reliability Engineering and System Safety*, *58*(3), 205–213. https://doi.org/10.1016/S0951-8320(97)00072-0

Madry, A. M. ¸, Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards Deep Learning Models Resistant to Adversarial Attacks. In *arxiv.org*. https://github.com/MadryLab/mnist_challenge

Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard´, P. F., Polytechnique, F., & De Lausanne, F. (2016). *DeepFool: a simple and accurate method to fool deep neural networks*. http://github.com/lts4/deepfool

Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2015). DeepFool: a simple and accurate method to fool deep neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 2574–2582. http://arxiv.org/abs/1511.04599

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I. M., & Edwards, B. (2018). *Adversarial Robustness Toolbox v1.0.0*. http://arxiv.org/abs/1807.01069

Papernot, N., Mcdaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. *Proceedings - 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016*, 372–387. https://doi.org/10.1109/EuroSP.2016.36

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, 582–597. https://doi.org/10.1109/SP.2016.41

Schwall, M., Daniel, T., Victor, T., Favaro, F., & Hohnhold, H. (2020). *Waymo Public Road Safety Performance Data*. http://arxiv.org/abs/2011.00038

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2016). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*, *128*(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, *32*, 323–332. https://doi.org/10.1016/j.neunet.2012.02.016

Taylor, R. (1975). Sequential Effects in Failure Mode and Fault Tree Analysis. *Reliability and Fault Tree Analysis*.

Taylor, R. (2012). Lessons Learner from 40 Years of HAZOP. *Loss Prevention Bulletin*, *227*.

Taylor, R., & Kozin, I. (2021a). *Design for Emergent Safety Problems in Handbook of Engineering Systems Design*. Springer.

Taylor, R., & Kozin, I. (2021b). *Hybrid Fault Trees for Continuous systems*.

Varshney, K. R. (2017, March 27). Engineering safety in machine learning. *2016 Information Theory and Applications Workshop, ITA 2016*. https://doi.org/10.1109/ITA.2016.7888195

Waymo's Safety Methodologies and Safety Readiness Determinations. (2020). www.waymo.com/safety

Yin, D., Lopes, R. G., Shlens, J., Cubuk, E. D., & Gilmer, J. (2019). *A Fourier*

Perspective on Model Robustness in Computer Vision.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. El, & Jordan, M. I. (2019). Theoretically Principled Trade-off between Robustness and Accuracy. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 12907–12929. http://arxiv.org/abs/1901.08573

Zhang, J., & Li, J. (2020). Testing and verification of neural-network-based safety-critical control software: A systematic literature review. In *Information and Software Technology* (Vol. 123, p. 106296). Elsevier B.V. https://doi.org/10.1016/j.infsof.2020.106296

Zhang, J., Taylor, J. R., Kozine, I., & Li, J. (2021). Analyzing Influence of Robustness of Neural Networks on the Safety of AutonomousVehicles. *31th European Safety and Reliability Conference (Forthcoming)*.

Zhong, Z., Hu, Z., & Chen, X. (2020). *Quantifying DNN Model Robustness to the Real-World Threats*. 150–157. https://doi.org/10.1109/dsn48063.2020.00033

APPENDIX A A System fault tree for AV

Crash
- Crash due to car leaving the road >> 20
- Shunt into leading car >> 1
- Following car ear end shunt
- Head on crash
-- Own car in the wrong lane
-- Oncoming car in the wrong lane
- Leaving the road
- Rear end shunt >>
- Crash into oncoming car while in lane >>
- Crash into oncoming car while overtaking >>
- Rear end crash while overtaking >>
- Crash due to vehicle entering from a side road >>
- Crash at junction >>
- Crash into obstruction on the road >>
- Crash into person on the road >>

1 >> Shunt into leading car
-- &
--- Car in front
--- Distance become zero
---- &
----- Car in front brakes suddenly >>
----- Own car too close >> 1
---- &
----- Driving close [Normal condition when approaching a leading car moving slower than own car]
----- Fail to adjust speed to that of leading car
------ Fail to brake >> 2
------ Brakes applied too late >> 5
------ Inadequate braking >> 6

2 >> Fail to brake
- Brake hardware failed
- Air in the brake hydraulics
- No braking signal from ABS
-- Algorithm error in ABS
-- Criterion error in ABS
-- No braking signal to ABS
--- No braking signal from braking control
---- No braking signal from speed control
----- Inadequate avoidance strategy
------ Algorithm error in avoidance strategy
------ Criterion error in avoidance strategy
------ Erroneous input from navigation control >> 4
------ Interfering signal from navigation control >>
----- No input from object detection >> 3
- Inadequate braking signal from ABS
-- ABS suspends braking
--- Wheel rotation speed mismatch
---- Brakes blocked
---- Road surface slippery

5 >> Brakes applied too late
- Delayed signal from ABS
-- ABS performance failure ??
-- Delayed signal from braking control
--- Processor overload or deadlock
--- Delayed signal from speed control
---- Critically delayed signal from avoidance strategy >> 8

8 >> Critically delayed signal from avoidance strategy
- Erroneous avoidance strategy
- Algorithm error
- Criterion error
- ~ Speed equalization distance (SED) > separation distance (SD)
{SED = f1(Braking force, road condition, leading car speed, own car speed}
{SD = leading car relative distance at detection}
-- Speed equalization distance critically high
--- Braking force critically low
--- Road condition critically bad
--- Leading car speed low (or stopped) [in this case either the design safety distance is too low, or the detection distance is too large]
--- Leading car speed estimate critically high
--- ~ Erroneously high-speed estimate for leading car from object recognition >> 9
--- Own car speed critically high
--- ~ Critical combination of own car high speed and poor braking or bad road condition
-- Separation distance at braking critically low
--- leading car distance estimate critically high
---- Erroneously high-speed estimate for leading car from object recognition >> 9
---- Erroneously high distance estimate for leading car from object recognition >> 10
- Wrong prediction of behavior of object in front
- Delayed input from object detection

9 >> Erroneously high-speed estimate for leading car from object recognition

9 >> Erroneously high-speed estimate for leading car from object recognition
- &
-- Erroneously high-speed estimate from camera
--- Poor input from camera
---- Noisy background (Low robustness)
---- Noisy leading vehicle image
---- Poor lighting
---- Poor visibility
--- Recognition error
---- Deficient NN training
----- Vehicle type set too limited in training set
----- Vehicle coloring not in training set
----- Overlapping images
----- Vehicle pose not in training set
----- Leading vehicle not recognized due to curve in road
-- Erroneously high-speed estimate from lidar
--- Noisy input from lidar
--- Poor lidar reflection from leading car
--- Interference due to second car overlapping signal to lidar
--- Wrong lidar signal set used due to roadway slope
>> 20 Crash due to car leaving the road
- Car deviates spuriously from the lane
-- Deviation in the vehicle steering
--- Tire bursts
--- Vehicle mechanical linkage failure >>
--- Vehicle power steering failure >>
--- Erroneous steering value from the steering algorithm
---- Algorithm error
---- Error in the steering constants
---- Error in the straight lane output from the lane following function >>21
---- Error in the output from the navigation control >>22
- Car fails to follow a curving lane
-- Erroneous steering value from the steering algorithm
---- Algorithm error
---- Error in the steering constants
---- Excessive output from the curve following function >>23
---- Inadequate output from the curve following function >>24
---- Error in the output from the navigation control >>25
-- Car travelling at too high speed for the curve and road conditions >> 31

21 >> Error in the straight lane output from the lane following function
- Failure of lane detection
-- &
--- No lane marking input from camera view correction >> 26
--- Inadequate default correction
-- default in the lane detection algorithm
-- Algorithm error
-- Detection of a spurious lane marking
--- Old lane markings not removed
--- Other items recognized as lane makings [e.g. lighting poles laid on the ground]
--- Filter lane markings not detected as such

26 >> No lane marking input from camera view correction
- Lane marking problem
-- Lane markings deficient
-- Lane markings obscured
--- Snow
--- Rubbish, mud etc. covering the lane markings
--- Parked car [When the vehicle is not in the inside lane]
- Low contrast in the corrected image
-- Camera failure
-- Poor lighting conditions
-- Low contrast between the road surface and the marking
- Noise in the corrected image
-- Poor lighting conditions
-- Camera communications electronic noise

23 >> Excessive output from the lane following function
- Lane marking problem
-- Lane markings deficient
-- Lane markings obscured
--- Snow
--- Rubbish, mud etc. covering the lane markings
--- Parked car [When the vehicle is not in the inside lane]
- default in the lane detection algorithm
- Algorithm error
- Detection of a spurious lane marking
-- Old lane markings not removed
-- Other items recognized as lane makings [e.g. lighting poles laid on the ground]
-- Filter lane markings not detected as such
- Low contrast in the corrected lane image
-- Camera failure
-- Poor lighting conditions
-- Low contrast between the road surface and the marking
- Noise in the corrected image
-- Poor lighting conditions

-- Camera communications electronic noise

31 >> Car travelling at too high speed for the curve and road conditions
- &
-- Bad road conditions [mud, ice, heavy rain]
-- Bad road conditions not accounted for
--- Bad road conditions not detected
--- Bad road conditions not compensated in the steering algorithm
- Poor tire condition
-- &
--- Tire wear
--- Poor tire condition not detected
- Car speed simply too high
-- &
--- Car speed limit incorrect in map
--- Traffic speed limit sign detection error
---- Traffic speed sign not detected
----- Sign obscured [e.g. by trees or parked vehicle]
----- Sign pose makes identification difficult
----- Sign dirty or snow or frost covered
----- Low light conditions
----- Ambient light from signs, vehicles give wrong color
---- Traffic speed sign wrongly identified
----- Sign partially obscured
----- Sign dirty or snow or frost covered
----- Low light conditions
----- Ambient light from signs, vehicles give wrong color

APPENDIX B Example of Adversarial robustness testing

Here we present an example of adversarial robustness testing. We used the LeNet-5 model (LeCun et al., 1998) as the target NN model. The LeNet-5 model is trained on GTSRB dataset. Five attack methods (namely, FGSM, PGD, DeepFool, JSMA, and C&W) are implemented in the Adversarial Robustness Toolbox (ART) (Nicolae et al., 2018).
Table 0.1 shows the attack parameters and adversarial accuracy understand different attack methods.

Table 0.1 An example of adversarial testing result

Attack method	Attack Parameter	Accuracy_orig	Accuracy_adv
FGSM	norm=inf, epsilon=0.2, untargeted	0.983372922	0.112430721
PGD	norm=inf, epsilon=0.3, untargeted	0.983372922	0.005542359
DeepFool	maximum number of iterations =50,untargeted	0.983372922	0.013460016
JSMA	maximum fraction of features being perturbed= 0.3, untargeted	0.982	0
CW	norm=2, learning rate=0.01, maximum iteration=50, untargeted	0.982	0.534

APPENDIX C Examples of Saliency map explanation

Some of the threats (e.g., adversarial examples) are hard for a human to understand. Methods in the field of explainable AI (XAI) can be employed to identify the influence of threats on the NN performance (e.g., attack successful rate). It enables researchers to understand individual prediction, to peer into the inner workings of DNNs, and to trace the intermedia behaviour of NNs via visualizations. Some methods are able to provide high fidelity explanations, such as LIME (Ribeiro et al., 2016). In the case of an image classifier, it is also common to use gradient measurements and saliency map explanation (Simonyan et al., 2013) to estimate the importance value of each pixel for the final classification, such as DeepLIFT (Shrikumar et al., 2017), and Grad-CAM (Selvaraju et al., 2016). In Figure 0.1, we show an example of using Grad-CAM to identify the influence of input perturbation. The image is from CIFAR10 dataset (Alex Krizhevsky et al., 2010), and we use a CNN with two conv layer, the attack method is PGD. For a successful attack, the target NN model looks at the different features compared to the clean image, while for a failed attack, the target NN model looks identical to the clean image.

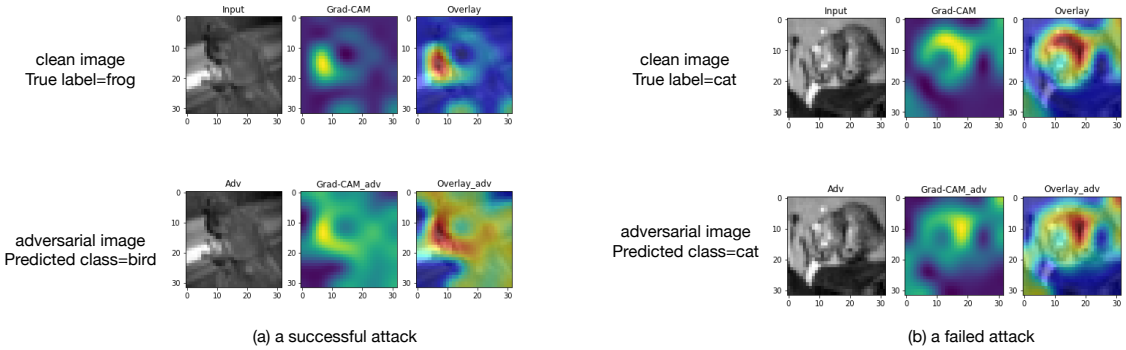
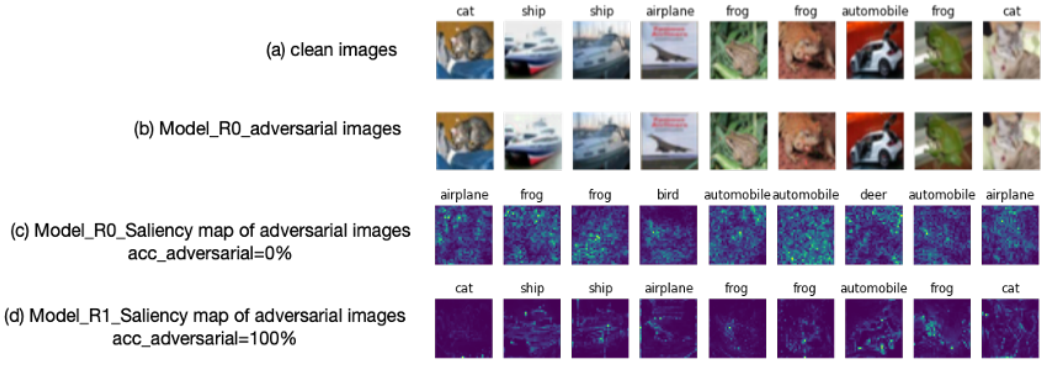


Figure 0.1 Grad-CAM visualization of successful attack and failed attack



Experiment setting: dataset-CIFAR10, two ResNet-based models, attack method-JSMA
Model_R1 is more robust than Model_R0

Figure 0.2 Saliency map visualization of the influence of adversarial attack