Master´s Thesis

Master in Electronics, Robotics and Automation Engineering

Safety check in critical safety scenario for self-driving vehicles

Autor: Raúl López Martín

Tutor: Daniel Limón Marruedo

Department of Systems and Automation
Higher Technical School of Engineering
University of Seville

Seville, 2020

Master´s Thesis
Electronics, Robotics and Automation Engineering

Safety check in critical safety scenario for self-driving vehicles

Autor:
Raúl López Martín

Tutor:
Daniel Limón Marruedo

Department of Systems and Automation
Higher Technical School of Engineering
University of Seville
Seville, 2020

Proyecto Fin de Carrera: Safety check in critical safety scenario for self-driving vehicles

Autor: Raúl López Martín

Tutor: Daniel Limón Marruedo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

Appreciations

My special thanks to Hitachi Company and Vulcanus Programme for allowing me to what can only be describe as the best experience any engineer may wish for.

To my family for always trust in me and always keep supporting me.

Special thanks to Daniel Limon, tutor of this project, for trusting in me and all the lessons he gave me.

Special mention to Rodrigo Alcubilla for making the whole stay in Japan a year I will never forget.

Resumen

En un Mercado emergente como es el de los coches autonómos una de las funciones esenciales es asegurar la seguridad del funcionamiento de dichos sistemas. La mayoría de los comportamientos del estado del arte son capaces de manejarse en un escenario sin anomalías. Sin embargo, las dinámicas del entorno, tales como las condiciones metereológicas o las oclusiones de los sensores, pueden comprometer el funcionamiento de estos. Para ampliar los escenarios en los cuales estos sistemas son capaces de funcionar, es necesario incluir nuevas funciones de seguirdad para una conducción segura en cualquier entorno. Esta contribución demuestra una validación para vehículos autónomos basada en las condiciones del entorno. Se propone una comprobación de los obstaculos y limitaciones de los sensores. Para ellos se define una Region de Interes (RoI). Combinando ambos conceptos se obtiene un valor cuantitativo del conocimiento de los entornos del sistema. La idea propuesta se basa en modificar el plan de actuación según dicho valor, mejorando el tiempo de reacción ante situaciones imprevistas. Los resultados de la simulación e implementación física en el coche autónomo muestran una mejora en los tiempos de reacción ante situaciones fuera del dominio operacional designado. Se considera que este proyecto resuelve una de las condiciones obligatorias para conseguir un coche autónomo con un nivel de automatización de nivel cuatro.

Abstract

Fully autonomous vehicles must guarantee safety. Most of state-of-the-art behaviors can drive safely on scenarios with no anomalies. However, Dynamics, *such as weather conditions or occlusions*, on the operational design domain might comprise the security. For further automation we need to enlarge the workbench for the technology allowing to work safely even on those situations. We contribute with a safety validation for AV based on the conditions of the scenario. We propose a check for sensor visibility and limitations. Additionally we create a definition of a Region of Interest (RoI). Merging both data we obtain a quantitative value for environment awareness. The proposed idea is to, based on that value, modify the acting plan, improving reaction time for unforeseen. The results from simulation shows that using the proposed idea, dangerous situations can be avoided. Henceforth, the fulfillment of the derived safety assessment validation can guarantee safety of the AV. The proposed idea is to, based on that validation, modify the acting plan, improving reaction time for unforeseen and endowing the autonomous vehicle of a safety check. This update is mandatory for self-driving vehicles that long to achieve a level 4 automation, as the human is no longer the responsible for safety check in unpredictable situations.

Index

TABLE INDEX

FIGURE INDEX

List of abbreviation

Abbreviation	Meaning
GPU	Graphic Processing Unit
CPU	Computational Processing Unit
NPU	Neural Processing Unit
NN	Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
OD	Object Detection
SSD	Single Shot multibox Detection. Framework for OD
YOLO	You Only Look Once. Framework for OD
DoG	Difference of Gaussian
LiDAR	Light Detection And Ranging
3D	3 Dimensions
Laser	Light Amplification by Stimulated Emission of Radiation
FPS	Frames per second
m	Meters. (cm=centimeter… etc.)
CCD	Charged Couple Device
CMOS	Complementary Metal-oxide-semiconductor
Gb	Gigabyte
GDDRx	Double data rate, a type of dynamic random access memory
GHz	Giga Hertz
CUDA	Compute Unified Device Architecture
AI	Artificial Intelligence
POMDP	Partially Observable Markov Decision Process
MPC	Model Predictive Control
FSM	Finite State Machine
RoI	Region of Interest
RT	Real Time
App	Application
ADST	Autonomous Driving System Toolbox. From Matlab.
BEP	Bird Eye Plot
ROS	Robotics Operating System
ID	Identification Digits

Table 1 Abbreviation table

1 INTRODUCTION

How do we understand the environment surrounding us and how do we act within it? For humans, and rest of the animals; once they reached their maturity, being able to extract information from the environment and react in consequence is a process that usually happens without much effort. A human does not have to think how to steep his feet to not fall while walking, like a lion does not have to think he has to turn right while running to not bump to a tree. But this process are not innate for us, yet we develop these skill since the moment we are born. We learn how to climb stairs, how to stand up, how to talk and how to read the surroundings to extract the key information from it. Still, the functionality of the brain is not fully understood, yet; as humans always do to create things, we can inspire ourselves in the nature to try to reproduce an artificial brain. If brain were fully understood and a perfect model of it could be simulate, then maybe a complete artificial brain could be created. As there is no complete model for the whole brain, we just focus on all members that are part of it. That is why using a similar structure than the one presented in our brains such a connection on simple links in a complex system, still much different from the one in human brains, can be the base to found an artificial brain. This artificial brain will undergo to a series of process to learn how to behave, depending on the applications which we want to carry out.

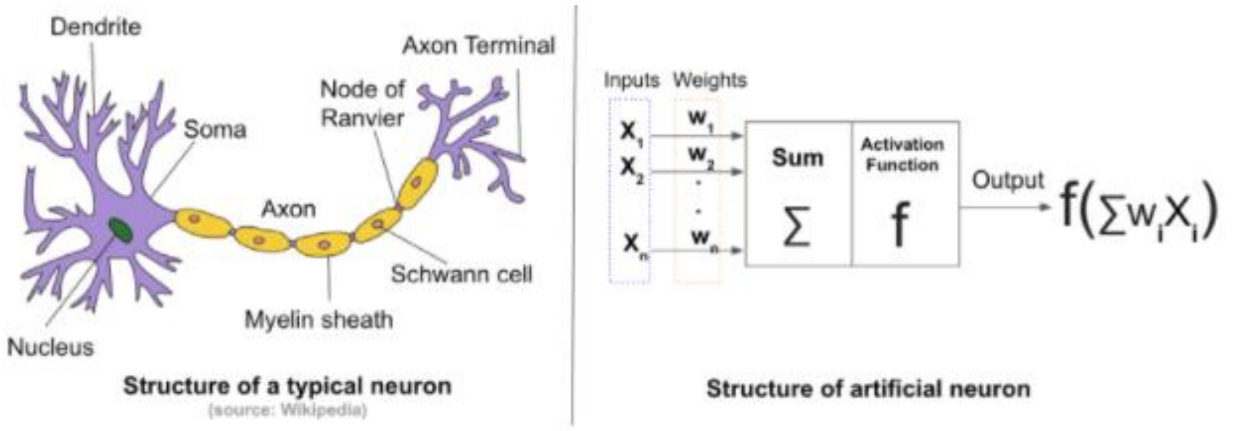


Figure 1 Similarity between biological and artificial Neurons

But this is not the only way to develop artificial intelligence. Since early in the 50s several attempts to design and develop intelligence have been tried out. From large states machines and tree behaviors that compiles all the possible situations that can happen, to mimic chatbots that tries to copy certain activity by being exposed for a long time to the correct answers. Any process that allows a machine to learn from its errors can be considered as an artificial brain.

What it would be use this artificial brain for? There are plenty of uses for artificial intelligence, every action that humans and animals performs can be imitate by a machine using AI, from how to talk to people, how to drive a car, how to behave in situations or even how to play board games. Of course, not all machines have to be humanoids or big servers[1] and most of them can make the simplest action and still use an artificial neural network. Not all the machines tries to reproduce the whole brain itself, but just a part of it able to perform the desired action. There are plenty of choices when these artificial brain become pretty handful, as early detection on cancer or diabetes related sight problems. Some of the functions have a road long yet to be traveled, as it

[1] As most Sci-fi movies and books like to imagine.

comes, for instance, to creativity process such as painting or coming up with jokes. Others are rather advanced and the applications are already in the society or soon to be released. One of the target for these intelligent machines is to perform a self-driving car where the driver is not needed anymore. And that is the topic we will be discussing over this document, how to make a car move by itself, which information should be recollected from the surroundings and how to react about that information.

How a car does drives itself? Or, another way of asking ourselves this, what kind of actions has to do a machine in order to be able to drive a vehicle? Of course it will have to connect with a GPS system and be able to trace a route from the initial place to the destination, but mostly, it has to make decisions from the information obtained from the sensors. One of the task we face in this problem is not only to be able to record the environment that surrounds the cars but also to be able to understand and to make choices from that information. Even for humans, get use to recognize dangers, pedestrians, animals and traffic lights, the first time we drive a vehicle can be overwhelming. Therefore, training a machine to be able to check the surroundings and recognize all the things that are going on around it, and with that information make a decision, it is a wide task. When it comes to recognize and make a decision, one of the most common way to perform this purpose is to use an Artificial Neural Network and train it to focus its optimality for the task itself. Although there are other ways to accomplish this task, as we will discuss later on this document, we shall focus on the NN behavior. Once our NN is already trained for this performance, it could be applied in real-life actuation. Once all of this has been done, the car would have a perfect understanding of the surroundings in that instant. But when it comes to human behavior not everything can be guessed by just one image and probably information from previous state is also required to make a prediction on their future movements. With all this information acknowledged, the system should make the optimum action.

Why to make a car drive itself? Surveys made among car users proves that around half of the population, in surveyed countries, is still feeling insecure about the fact of using an autonomous vehicle. Users and pedestrian are concern about traffic congestion, inefficient driving, but mostly, about the safety while using a car or walking on the streets. As a matter of fact, in early stages of NN, performance was not enough to these kind of applications. Lately, a lot of research have been made in NN and other machine learning techniques and its development, expanding fields of application more and more in the past few years. From the start of the first capable-to-learn systems in the early 60s, until the techniques for face-detection in your smartphone; the structure, training and computability skill for artificial intelligence have changed widely. In experiments performed in recent years, AI accuracy has overtaken human capacity for some applications, as it is image classification or detection for small spots in cancer detection, among many others. Nevertheless, most of the applications for AI in those cases are just used as a filter, to underline the cases that requires an expert attention. Same happens with the self-driving cars which are being sold at this time. Those cars, although they can drive itself, they require a driver placed in its position and focused on what is going on the street, able to react in case that some anomaly happens or a wrong operation has or is going to be taken by the car.

What have the future ahead waiting for us? Not all the scientific community are really willing to use or comfortable with NN and some of the applications which these are being used for. The way NN are used make of them as a black box where you cannot know what is happening there made them unpredictable, and that is the main reason not all scientist are wager to use this technology. Other technologies have still to grow to be able to perform a robust and fast performance. However, self-driving vehicles are already a fact, they are being sold already and improving their skills with their year. Will it come a future when fully autonomous vehicles can be used safely? Only time knows.

1.1 Context of this Project

This Project was developed during an internship in Hitachi Research Lab, placed in Oomika, Japan. Such internship was part of Vulcanus European Program that aims to merge Europe and Japan´s industries by funding bridges between companies and graduate engineer and scientist students. For the development of this project there were several resources available from the company. Starting with a high-technology computer able to create and computes NN as fast as possible, following an Autonomous Driving Vehicle, patent of the company. As this idea is meant to be used as an upper layer algorithm, ready to be included in any decision making module, none of the previously developed code were modify during the course of this thesis, only new modules were added to the previous ones.

2 GOAL

The first thing to do is to specified goals and search for the state of art in the desired field. The main purpose of this project is to create an additional safety layer to the decision making in autonomous vehicles. To accomplish that task it is needed to develop an Artificial Intelligence for decision making for a self-driving car. This car must be able to perform some task that also are crucial to be defined. These are the main tasks to perform by self-driving car:

- Being able to follow a route and reach a goal point in an optimum method.
- Detect other dynamic objects present in the environment and react safely for them and the car.
- Keep the track inside the lanes.
- Respect traffic rules.
- Perform the movement as smooth as possible

In order to achieve all those task, there are different ways to adapt a machine to learn how to safely drive and optimizing the route, reach from a starting point until the destination point (or at least one place nearby that last desired position). As we discuss in the previous section there are several ways to recollect data from the environment to use it on the car, so the ways to control the car may vary depending on how they take down all data. Likewise, some of the implementation were not aiming to create a full driving model for autonomous vehicles but few implementations as lane keeping and overtaking some slower cars, for instance. These are some of the different methods used:

- FSM: A machine with a finite number of state was one of the first ideas used to design autonomous vehicles. This method is as various and wide as the programmer wants to go. It can be used for a simple action as is parking the car, by riding the steps one by one; or it could implement big actions as a control for a whole plant. One of the biggest load it has the finite state machine is the need to program each one of all the possible situations that may happen in the road, which lays between a hard and risky thing to do. If one scenario is no declared in the FSM the car would not be able to react, but it is needed to be said that most of the other methods does not have neither the ability to react against undefined situations. This method have a bad reusability and the process must be rewritten for each situation that it want to be done.

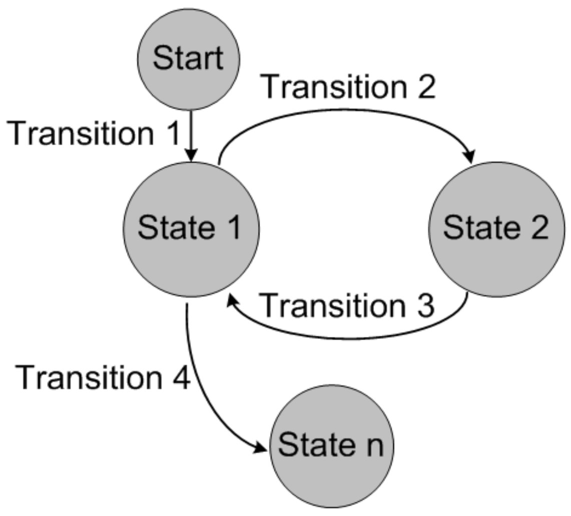


Figure 2 FSM Scheme

- Behavior Tree: Can be used to determine all the possible consequences or actions to be taken in a certain situation. It is not only applied to engineering-related topic, but can be also used in humanistic science and other fields. Once all the possible actions to be taking (we are going to focus on that kind of behavioral tree as it is the one it is used on autonomous vehicles) starting from the top (root node) and going as the European reading direction, all the cases are going to be considered. If some of the nodes (called children) which are ticked performs an action the exploration of the tree-like structure stops. There are different nodes and types of children which varies slightly the actuation of the program. All these behavior can be structured and implement by a high level FSM that may include other FSM (Hierarchical FSM) or other control loops for each one of the actions of the Behavior Tree.

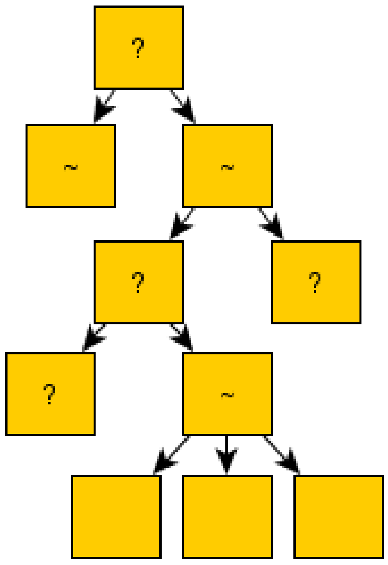
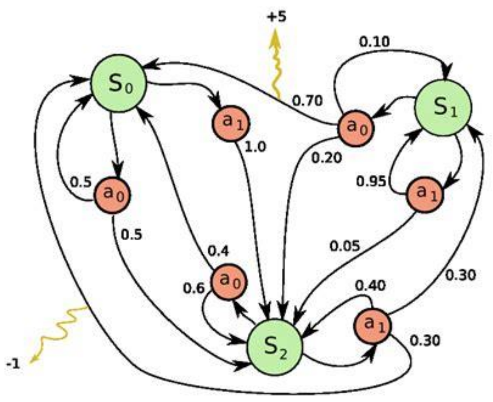


Figure 3 Behavior tree Scheme

- POMDP: With a several definition of states and actions in the system, with a Markov Decision process, each state has some changes among them caused by the actions and modeled by a probability of that change of state to happen. That probability may be fixed beforehand or change during the training of the machine. When referring to Markov Decision Process, Partially Observable means that not all the states from the important objects can be determined. Therefore, some presumptions have to be taken for those objects in order to establish the most likely state to be. The larger the number of states is, the larger amount of time the system will take to process all the information. A way of implementing this in a self-driving scenario is by creating some states car and other objects can be using at the moment, and search each one of the options it could be taken. Until here it would work the same way as a behavior tree works, but, instead of exploring all the options (that may take a considerable amount of time), Markov Decision Process run random simulation among all options and evaluates the result with a reward function entered before the test. All the branches who hasn't been explored enough or that have a high percentage of success (reward over some threshold value) are most likely to be explored in consecutive search. When this process has been done enough times, the action with the best reward function value. Even though this is not an optimal method result are quite accurate within those experiments. One of the main problem accounted to this method is the huge computational load that the server must take in order to perform all the task. As is not only for the egocar, but for all the other objects in the scene, the possible states must be reduced to a minimum. One of the current ways to include this was to create different driving policies and to assign to each car the most likely to be used. And by observing the move of the other objects in the past, determine change point in driving policies that can allow you to detect which route they are taking. For the ego car, two different controls with two different frequencies were used, one to determine which policy to use (at a rate of 1Hz) and other to determine the optimal action within that policy(at a much higher rate of 30~50Hz). This last policy will include also safe actions in case of anomalous behavior of the other actors. Although these changes were made, only a few car can be predicted at the same time if the actions was meant to be done in real-time operation, as it should be for a self-driving car.[2]



Figure

4 MDP Scheme

- End to End NN: As it was explained before, neural networks work in a similar way that human's brain do, but no analogies are made between those two. In a Neural network there are various layers with neurons in it. In those neurons there is a simple process carried out inside. All the inputs of that neuron are weighted and then summed up. At the end of that process, an output function is usually used, mostly a sigmoid or lineal one are the ones more common to be used. At the start, these values for weight the inputs of the neurons are chose either random or with a fixed values, as they are meant to be changed during the training. In order to train the network, several inputs are introduced in the neural network (which has to be designed in a proper way for the application) and the outputs are compared with the desired output (which also have to be provided). With different techniques, the weights of each neurons are modified to get the result closer to the desired ones. What these networks can use as input and output are not limited and there are plenty of ways of use them. One of the application used for self-driving cars is to train the network from the starting point until the control

[2] This method was implemented by Galceran et al. in the document reference []

signals. This mean that using all raw data from sensors and cameras and introduce each pixel or data into the network, and taking as the output the speed/acceleration of the car and the angle of the steering wheel. One of the main problems for neural networks is the incapability of extrapolate, or doing so with a poor result. For that reason a really big data is needed to train them. As this network has a big input size and the situations widely changes, the training requires even bigger datasets. Even with that size of data, if something which is not learn by the network happens, the machine will not know how to react. Other problems with Neural Networks is that work as Black box model after the training. Is not possible to know a priori what the machine will decide with a certain input, and that can always carry big problem in specific situations.

- Chauffeur Net: This method was designed by Google Research Subsidiary Group, Waymo. It works with a neural network but instead of installing it from end to end, as previously explained, it work, as they call it, Mid-to-mid. We already explained the problems related to neural networks, how untrusted they can become in certain scenarios and how big is the sack of data needed for the training. What Chauffeur Net provides is a more reliable way of introducing neural networks into decision making for self-driving cars. Instead of using raw data from the sensor as the input for the network, they processed those data into distinct maps with miscellaneous information such as traffic light or other dynamic objects. Those maps are fed into the network which has to output a safe and accurate answer. Instead of controlling directly the acceleration and steering wheel angle they chose to use the route, and accordingly, the next position for the ego car. For the training they used not only data from an expert driving sensor recording, but they also manipulated those data (thanks to the process of images, synthesize those data is easier) to create the worst case possible to train the network also in those stages. Also there is a dropout for imitation to avoid the car to learn just to repeat all that the car has been doing the previous states.

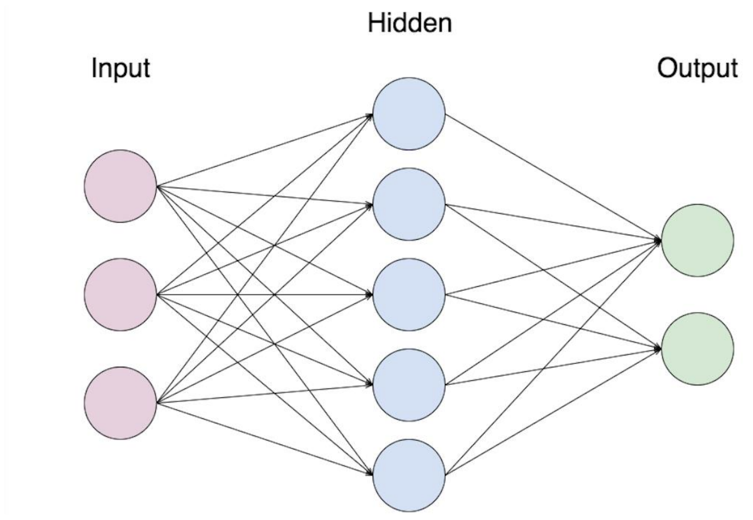


Figure 5 NN Scheme

- MPC: One of the most important thing when building a model predictive control is to acquire a precise model for the system we want to control, in our cases it would be for a car, either dynamic or only kinematic models can be used, depending on the application and purpose of the program. With a dynamic model of an Ackerman structure we would be able to control the position of a car giving the speed and steering angle. As the model gets better, so does the control.

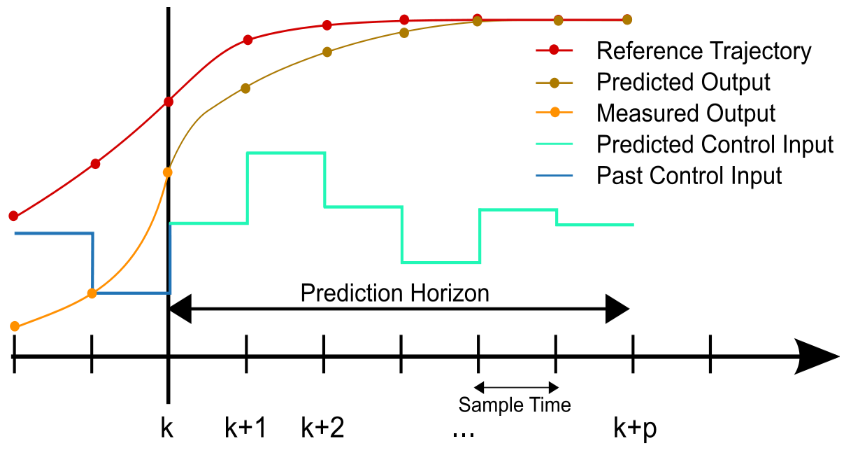


Figure 6 MPC Time representation

What we carry out on this project is a safety measure based on the sensor information and map topology, to validate how safe it is to perform a certain action. Within this document we explain the main idea to evaluate the security, and show some examples when this technique could prevent dangerous situations. What we aim to obtain is a safety validation on urban scenarios with low visibility whereas the car maintain its optimum velocity. In an intersection that presents obstacles on the corners, such as street display elements or other parked vehicles, where incoming cars are not easily visible, our safety validation improved the safety in the engage moment.

We designed a safety validation based on the occlusion on the sensors that measures how many hidden areas are left unchecked. Furthermore, tested the proposed idea in two simulation environments and in real-life situation showing the enrichment of safety for the ego vehicle. With this proposed idea we created a mechanism to evaluate the safety degree in any situation, endowing AV with a self-aware risk management that takes on the responsibilities that hitherto relied on human drivers.

3 VISION IN VEHICLES

For a machine to read the surrounding, are two main point which are crucial. How to get information from the analogic world to the digital region, and, once the information is there, how to interpret that information. Both field will be cover in this section, explaining the actual techniques to read, classify and detect object from the surroundings. All these parts will only be the ones needed for vision recognition task, but for a vehicle to be autonomous, several parts need to be attached all together to provide a full safe and reliable functionality. This process is crucial when looking at decision making in self-driving cars and other vehicles. How they percept the outer world and how the data is processed is an outstanding part for the decision making, as the decision must be taken based on the data received by these sensors.

3.1 Analogic to digital

3.1.1 LiDAR

Lidar stands for Light Detection and Ranging and it is a method of detection based on laser which provides a 360 degrees information (which can be amplified to a 3D approach by carrying out the laser detection in different layers and angles). The functionality is the following, a stream of a laser roam around the car from the LiDAR spot (usually on the top on the cars, as shown in Figure 7 LiDAR Working) doing a 360 turn and receiving the signal and calculating the time it took the signal from its departure to reach the object, so that it can be placed in a 3D map. This process is repeated in different angle of azimuth so it can take a third dimension the information. Each one of these repetitions would be each one of the circles shown in the following picture

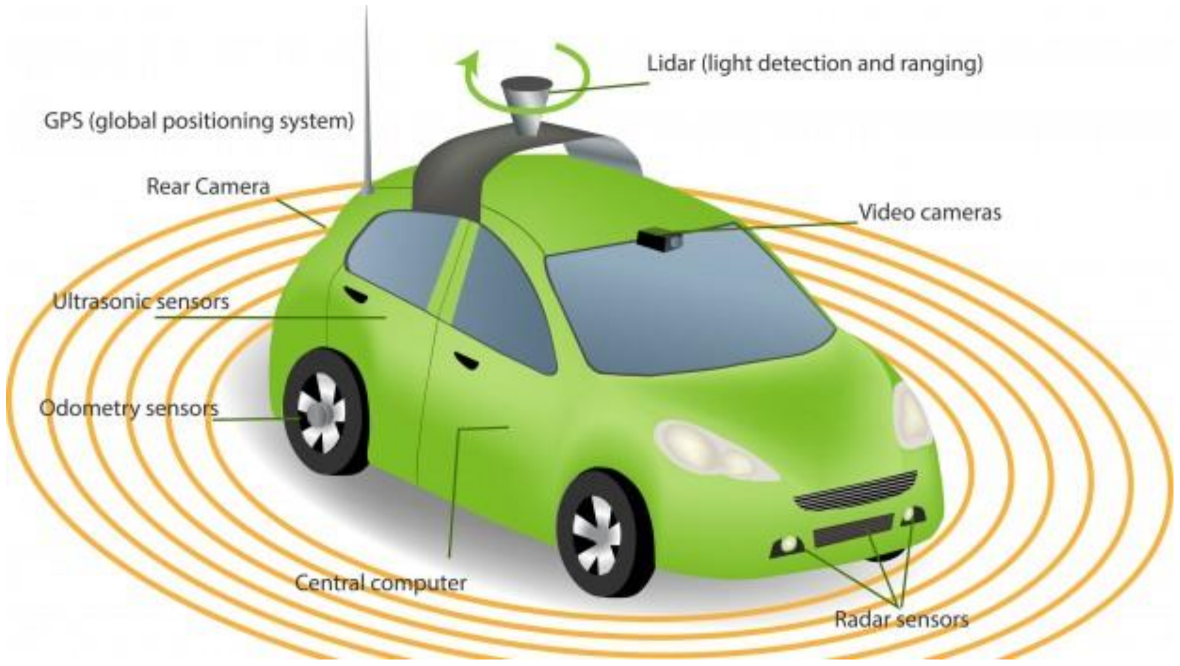


Figure 7 LiDAR Working

In the previous picture a typical disposure of sensors in an autonomous driving is dispatch. As it is better to have reliable data from surroundings, it is always an improvement to obtain data from more than one source in more than one method.

LiDAR has some really good specifications, which makes it really valuable when obtaining visual data for these kind of applications. One of its advantages is that it doesn't rely on the light ambience, making easier to detect object in the map created. It also doesn't have a range limitation when it comes to angles, it can cover all around the car, except from those angles pretty near the car that are not caught by the perspective of the LiDAR, if placed on the top of the car. Length range is also pretty good, covering up to 250m and with an accuracy of 0.5cm. Ahead, an example of a map created by LiDAR is shown.

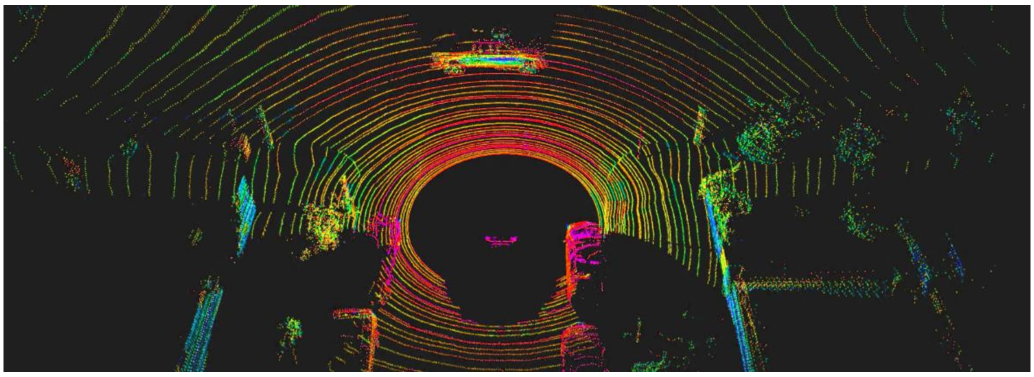


Figure 8 LiDAR Example of data

Once you get this image, you need to train your vehicle to be able to recognize other vehicles, pedestrian and other obstacles that may exist in the road. Also, lately, lines for keeping track of the road and the direction and orientation of objects have been also detected by this technology as the power of the laser steam changes depending on the surface they reflect on.

3.1.2 Camera

For the visual recognition with the camera, there are two main technologies that are being used. CCD and CMOS. Both of them works mostly in the same way, working by MOS transistors technology, but with a different way of recollecting the light from the target. The main, and simplified, scheme is the following:

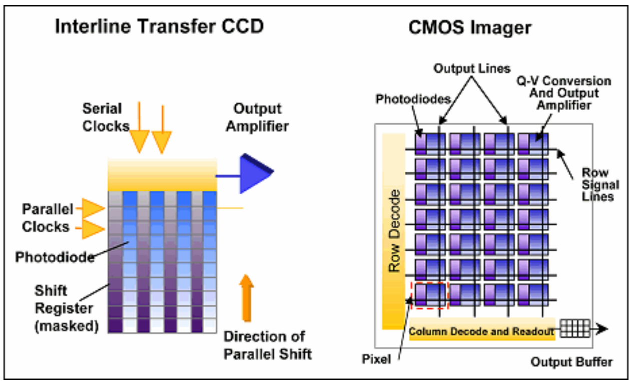


Figure 9 Scheme for CMOS and CCD cameras

Keeping the transistor functioning in the inversed zone, when the protons from the light enters in the well of the transistor, the energy they are carrying through their movement is transmitted to the electrons, creating a current in the device. By making an array of transistor for each pixel we will be able to distinguish how strong is the energy in each one of the pixel. For each pixel, one transistor is needed in case the picture is only in black and white. To make colored images a prefilter is needed in the transistors. Each one of the will only allow one the 3 primary colors to go in.[3] With that made, the system is allow to take a picture, and by repeating this process in a periodic way, videos are taken. This is the technology included in the cameras from self-driving cars. About which technology to use is up to the company and both of them have their unique weakens and strengths, which made them fit better for some applications or for others. The big challenge when it comes to vision by cameras is not the fact of obtaining the value of each pixel but to interpret them. The information that is available from a camera is wider than from any other sensor nowadays, although the process of that information is the hard task to accomplish. There are several ways to solve this, as CNN, and, even though the accuracy is quite well, they still have some room to improve.

3.1.3 Radar

Same technology that has been used since the 30s for detecting incoming vehicles, mostly flying, can also be used for autonomous vehicles. By sending an electromagnetic wave in a direction, and by catching the reflection from when this same wave hits with some object, you can determine the position and velocity of the objects:

$$f` = f * \frac{v_{wave} + v_{observer}}{v_{wave} - v_{source}} \quad [1]$$

Doppler Effect

$$distance = v_{wave} * \frac{t_{roundabout}}{2} \quad [2]$$

For EM waves, the velocity will be equal to the speed of light, 300.000 km/s. Based on these two physical effects, measuring the frequency of the wave and the time it takes for the wave to come back: knowing at which speed the cars is currently moving, it is possible to calculate those two values. Distance and velocity of obstacles ahead in the road. Different sensors are placed all around the car to add some safety to the vehicle by measuring distance and velocity from the obstacles in various angles. With a similar effect but for a closer range (up to 6 meters) ultrasounds can be also be used, perceiving the same effects than the radar. Usually radars are placed in the front part of the car while ultrasonic sensor are placed for backwards movement.

3.2 Visual Recognition

Each autonomous vehicles will use assorted sensors, with at least one of the previous list, depending on how they would like to proceed with the visual recognition problem. Once the data is collected, a process of those data is needed. When looking to a picture and trying to figure out important information there are different grades on how accurate you can determine the objects present in the frame. Depending on those grades there are different methods to extract the features and applications to be used.

[3] The three main colors for additive systems are Red, Green and Blue (RGB).

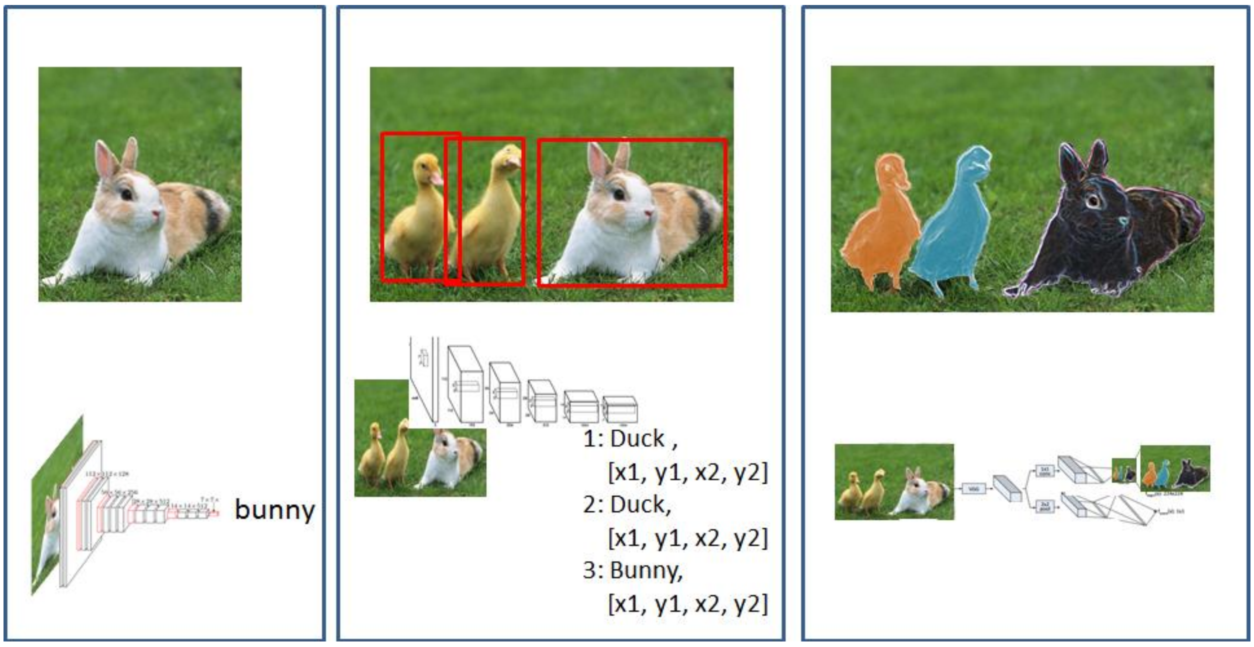


Figure 10 Image classification, Object detection and Instance Segmentation

The first step you can do with an image is to recognize the objects presented on it, even when there are more than one, and export that label outside. On the example we can see, how we got an answer of "bunny". This is called Image classification; this is not enough for the application we are trying to use.

The second step would be to surround the object with a square inside the image. As we are looking for more than one object and bounding this object with a box, usually the prediction of an object with its box comes with a probability of existence and probability of what kind of object it is. One of the things to take into account when carrying out this is when bounding boxes overlap, matching one object with more than one object. This is not the most precise one, but is enough for detection pedestrian, traffic lights, bikes, cars and others objects typical from the roads.

The third step, is to not only to select which kind of object it is, and where it is placed, but to limit its shape and differentiate each one of the objects, where are the boundaries and connections with other objects. It gives the biggest accuracy for problems of location and detection of objects, but takes much longer times to make this process.

4 SELF-DRIVING CARS NOWADAYS

The fever of self-driving vehicles is at its peak currently, as many of the main developer of automobile companies are investing considerable amounts of money on research on those topics to achieve as fast as possible a dully autonomous vehicle. Being a technology developed since early 80s, there has been many steps to reach where we are now. Self-driving cars improve in the way daily vehicles works, and is only the first one among several modifications until an SmartDrive system can be fully developed, Inter-connected (where it relies one of the main problems in the present, due to the funnel in wireless communications) cars with the capacity of reading the environment. This could reduce the risk of accidents in the roads, up to 60% by 2030 with the use of self-driving cars. However, this growth is unequable over the world, depending on the legislation and presence of R&D industry in each country. All those countries trying to get self-driving cars in the present world started a race to win the market. The race for self-driving vehicles has already started, but there are some struggles they have to overcome. Besides the technological problems to be solved, as it is a real change on the way transports are used, society needs to get used to these changes, therefore, as Violeta Bulc said: "Autonomous mobility will only real take off to the extent that is accepted by passengers and citizens as human beings and by society at large". You cannot get ahead the market, so these vehicles will only be profitable as long as the target group is big enough, and that start with optimizing the performance of them. Other problem to deal with is the ethical behind the programing in decision-making and the responsibilities in accidents, which are deeply different depending on the country. There are different opinions about who should be accountable for the accident (in case there was one), creating a gap in the law about insurance. Most of the cars released into the market by now are Level Three automated driving2, so pilots should never distract themselves from the road, and they have to react in case an incoming accident. This means that responsibilities from crash and pedestrian injuries are still pilots' responsibilities. This could be one of the main reasons that development of self-driving techniques are so uneven, as the laws adopted by each country varies, some circumstances does not support the implement of this raising upturn.

From now on, the driver will have helps on the direction, speed, and maneuver, which implies a change in mentality of the drivers to be willing to take this change. Alongside with this problem, comes the moral implied in programming death near experiences reactions for the autonomous vehicle. "What to allow to do to preserve driver and passenger's life?", "Has the car save a pedestrian or the driver?", "What if there were more than one pedestrian that could be injured?", "How many life has to be put in danger to sacrifice drivers´ life?" are some of the question being held by moral professionals. American Automobile Association studied that around 70% of American would be scared of to ride a fully self-driving vehicle. This fact states that society is still yet to adopt this new technology as a reliable one.

The three main zones were this technology is researched and they are making progress are Europe, Japan and USA. This does not mean that other zones are not working currently on this technology, but their short time dedicated to it or lack of resources make them to be outside the highlights on this reinvention of driving as we know in the present.

As for Europe, there is a different horizon for this technique. As Europe cannot be defined as an unique country when it comes to laws and insurances framework, all the efforts to try to create an standard law it has been harder. There is not a clear frame for all these situations, which involve self-driving vehicles. That makes uneven develop inside Europe, with each country applying laws separately. There were a restriction until midterm of 2017 in Europe that forbid any autonomous vehicle to move faster than 10 kilometers per hour [6]. This lack of laws and support to this researching, makes the market less attractive for the companies who are already working on this, such thing happens for Volkswagen. They have to release the new technology in other part of the world, because Europe is not the right market nowadays. As Johann Jungwirth, Volkswagen

Group chief, said in 2017 “And then comes Europe. We would love to come earlier since it’s our home market, but the legislation just isn’t there”, pinpointing that Europe´s laws are not yet created to accept this technology in the society. When it comes to this field, the variations made to the laws has been really slow during the last decade, delaying almost 5 years to catch up with other countries on the head of this race . All the procedures are trying to be done really slow and with the right terms so they can last longer, that is why they are doing some evaluations about ethical questions and how they should be answered. A team of experts that will discuss all these problems carries that study. Nonetheless, Europe is trying to catch up with the rest of the industry [8], by supporting a development of these vehicles. Several countries, such as German and England, they have changed the law in order to agree with the development already carried out in those countries. For example, Audi has already created several self-driving cars, which has been tested in closed circuits and they are expecting to release them to the market by 2020’s.

As for Japan, it has to been said that they raised a lot of positions on the “race” for the autonomous vehicle integration in the roads, having several companies researching in this field, like Toyota, Nissan and General Motors (which is one of the companies who has spent more money on this matter, with a budget of 2.25 billion $ for 2018) among others. Between Japan and USA are competing for the first spot on this race, both of them creating new vehicles with Level 3 Autonomous Driving for the market, and currently working in Level 4. Yet, not a single company has finalized developing a Level 5 Autonomous Driving vehicle. Waymo (USA) and General Motors (Japan) are the companies with the biggest amount of self-driving cars currently working in the world. Japan is now outlining laws about these vehicles, and how the responsibilities is hold by the pilot in all Level 3 vehicles. There are new policies for this incoming fleet of cars, which covers the main scenarios that could happen [1]. In addition, Japan is experiencing an uprising economy for the last decades that has position Japan as the second largest economy nowadays, which implies a larger amount of resources available for the market. Incoming large event such as Olympic Games is also a notable impulse for Japan to release this new technology as soon as possible. Prime Minister of Japan, Shinzo Abe, said this year that by 2020 (year of the event) self-driving cars will be try to be released to the market, including a fleet of self-driving cars dedicated for public transport . With this goal set in the future for Japan, all the research and development about this field is being speed up, with a lot of companies trying to get in the head of the revolution of the self-driving vehicles.

As a sum up for this section, a list of the biggest companies participating on AV technology nowadays:

Zone	*USA*	*Europe*	*Japan*
Companies	MobilEye	Audi	Nissan
	General Motors	BMW	Toyota
	Microsoft	Volkswagen	Honda
	Nvidia	Delphi	NTT Data
	Waymo (Google)	Renault	Fuji Heavy Industries

5 REGION OF INTEREST

This idea´s main contribution is a new layer, easy to implement and scalable, on ADAS, to increase safety in decision making, especially for level 4 autonomous systems. Up until now the decision making was based in 3 main process: Sensor information, recollecting and processing all the data from the sensor obtaining a clear scene of the surroundings; Plan planning, with all the information from the sensor to decide an optimum path to follow; and lastly to convert those decisions from the plan planning module to actuation signals, which are usually acceleration, braking and steer angle. With this technique we included two more steps in the process.

In sensor information it would need not to only get all the data from environment and to detect all the objects present in the scene, but also to check the occlusion those objects are carrying out in our vision. With that information we create an ideal plan the same way we were doing without safety check module. With that ideal plan we can enter in the safety validation module, where will need information from the previous two modules. In this module we will calculate the Region of interest required to check for the selected path in the plan planning. Obtaining this region of interest is hold on debate and can be determined differently by each user.

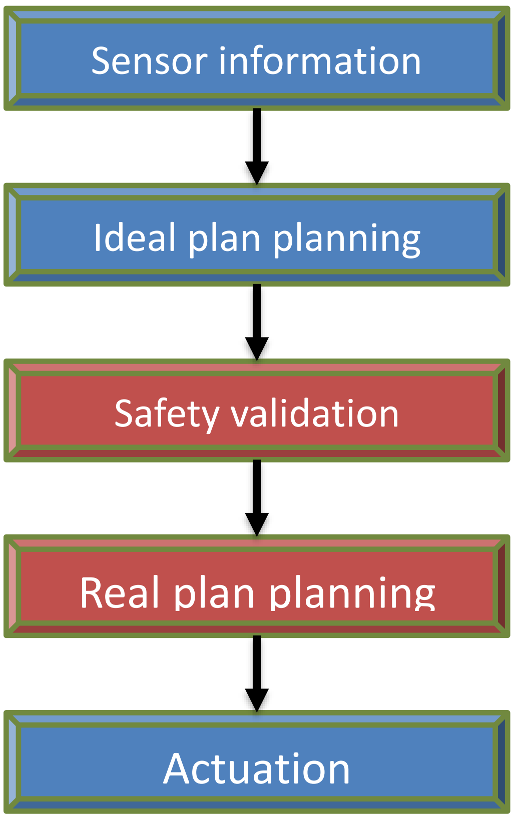


Figure 11 Safety check scheme propose

To explain what a region of interest is and how we would check that space, is better to establish some background scenario. One of the most interesting scenarios to apply the region of interest is those where the visibility is reduced. We shall take an intersection with some obstacles in the corner that blocks our sight for the incoming cars or the next step of the route. Within this scenario we shall restrain from the whole map which are the areas of this zone we need to check before taking any action into course. From the map of the area we can check where are the cars allow to come to our position, which is maximum speed those car can be travelling at and other features from near stances that may be useful for accident prevention actions.

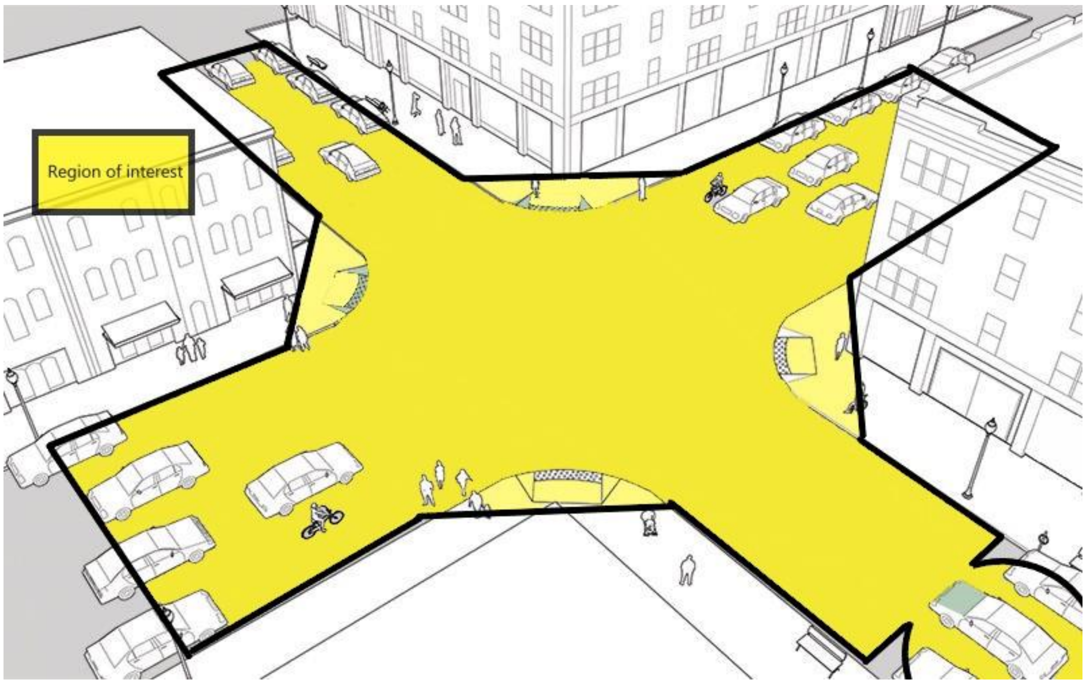


Figure 12 Region of interest required. Intersection scenario

From that scenario, with the new information obtained by the sensors in the first module (their range of vision and occlusions) we can establish how sure the ego vehicle is about its surroundings. Not only to check whether we can see a car or other vehicle coming to the intersection, but also to see how much of the scene we are being able to check. With all that information we can obtain the region of interest feasible. In other words, the part of environment the car is actually checking in that moment.

Looking at Figure 13 Region of interest feasible. Intersection scenario there are three different colored areas able to see. The **yellow one**, as it was already written in Figure 12 Region of interest required. Intersection scenario is the RoI required for that moment. The **green area** in the figure represents the area of the RoI that has no occlusion, therefore, it can be checked. That is going to be called the RoI feasible. As the green represents the RoI feasible, the areas that remains yellow in the figure are not only areas that are required for the action in that instant yet areas that are not covered by the sensors in that current moment. The **salmon color areas** in the figure are the spaces that are being covered by the sensors, but the information from them is no longer required by the decision making module. How to proceed with this areas is up to the user, as they can contain information that may be worth in the future such as tracking objects in those areas that might be moving into the RoI. Another action to be perform due to this information is the selection of the most important sensors, putting to sleep those sensors which are no pointing to the RoI of that moment, speeding up the processing of data.

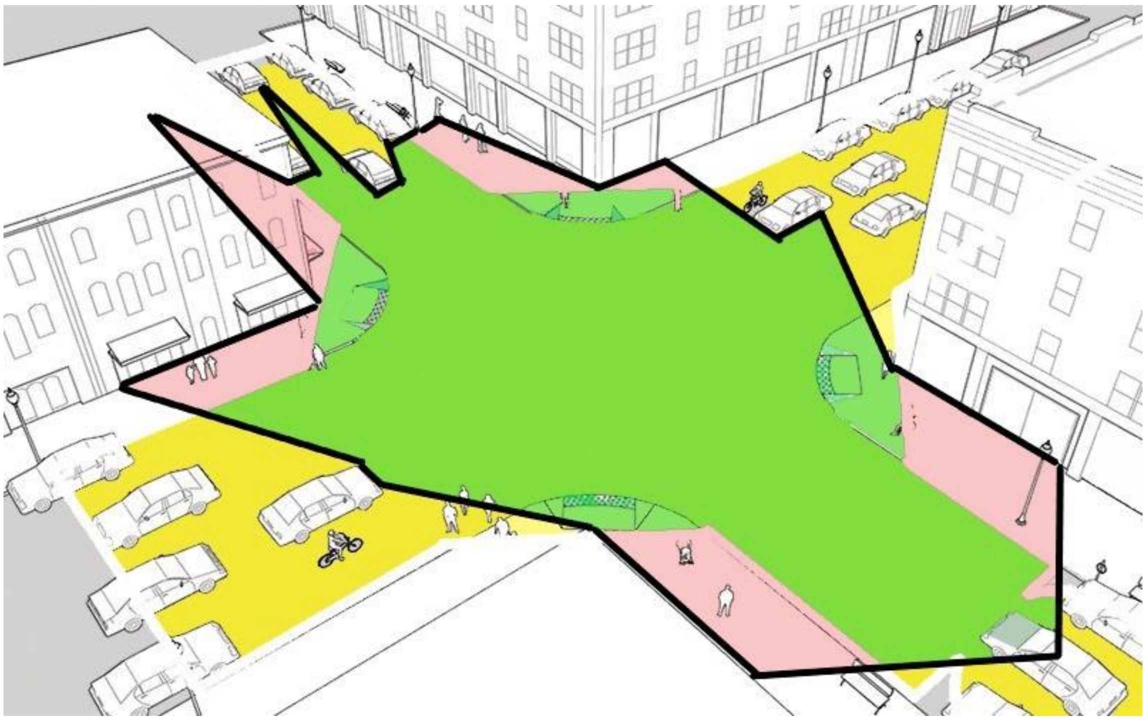


Figure 13 Region of interest feasible. Intersection scenario

After obtaining this feasible RoI, it is time to set up certain constrains as the last step in Safety validation. Based on the areas of the RoI which are in the sight of the ego vehicle and the initial plan planning done, there are certain actions to be taken. These actions can be chosen at ease, for instance, in case we have not enough visibility the speed can be reduced, or when doing a turn or an incorporation, when there is not enough area in sight, we can carry out a small move to increase our feasible RoI. One example of this situation would be a traffic jam, where the cars in front and in our back blocks the sight. In this situation is sometimes recommended to move the car slightly to the side to increase the visibility before starting any action.

Once this constrains are settled, a new plan planning can be carried out. This new plan increased the safety of the behavior of the car. Then, we send the information of the desired path to the actuators.

Something crucial to emphasize is the selection of this region of interest. One of the basic selection, based on the scenario, which can be done for the RoI is to check ahead for the maximum braking distance, based on the comfortable braking deceleration and current speed. On a more complex way, the RSS model [5][25] checked all the possible incorporations in the road, always calculating the needed braking distance, $d_{brake,max}^{ego}$. That distance can be obtained by using a comfortable braking deceleration, which was established to be under $3.4\frac{m}{s^2}$ [15], a_{brake}^{max}. If the car started braking with an initial speed of $\bar{v} = v_o^{ego}\frac{m}{s}$ then:

$$t_{braking}^{ego} = \frac{v_o}{a_{brake}^{max}}$$

$$d_{brake,max}^{ego} = v_o^{ego} * t_{braking}^{ego} - \frac{1}{2} * a_{brake}^{max} * {t_{braking}^{ego}}^2$$

To perform all the test in a safety way and to endow the system with a loose threshold to detect incoming dangers, we enlarged the distance by 25% to always stay on the safety side, $d_{brake,max}^{ego}$. Within this Region of interest we may encounter static obstacles in the ego vehicle path, other vehicles following the same lane than us, and vehicles and pedestrian crossing through our region of interest, yet, cars circulating on the opposite direction are not considered in this approach as it is an illegal behavior. With a definition of speed as following $\bar{v} = v_o * \cos(\gamma)$, with V_{LIMIT} as the maximum speed allowed in the current scenario, γ from the system of reference the road, being zero radians circulate on the direction of the road. $v_o \in [0, V_{LIMIT}] \subset \mathbb{R}\ and\ \gamma \in \left[\frac{\pi}{2}, -\frac{\pi}{2}\right] \subset \mathbb{R}$.

What it is widely defined in [5] about safe distance and keeping AV cars from dangerous situations, is deeply touched in [25]. Using the same nomenclature as of [25], ρ is the response time, and $a_{accel,max}^{ego}$ as the maximum acceleration for the car during the response time. As it was already presented in [25], to assure safety for the ego vehicle is needed to keep a safe distance, both longitudinal and transversal, enough to avoid any dangerous situation that may occur. We applied this same definition for safety to calculate our RoI.

With other vehicle on the scene with v^{ext} in our same road, r_{main} with the same speed limit than the ego vehicle, V_{LIMIT}^{ego}, as defined in [25] a safety distance for braking is:

$$d_{brake,max}^{r_{main}} = \left[v^{ego}\rho + \frac{1}{2} a_{accel,max}^{ego}\rho^2 + \frac{\left(v^{ego} + \rho a_{accel,max}^{ego}\right)^2}{2\left(a_{brake,max}^{ego}\right)^*} - \frac{v^{ext\,2}}{2a_{brake,max}} \right]$$

Note that we use the maximum braking for this formulation, differing from [25]

$RoI^{main}$is the extension of the road topology, with a width $\delta_{RoI} \in \left[-\frac{\delta_{lane}}{2}, \frac{\delta_{lane}}{2}\right]$, with a longitude of $d_{brake,max}^{r_{main}}$.

For roads incorporating to r^{main} , $r_{int}^1, r_{int}^2 \dots r_{int}^n$, with an incorporation angle γ^n, and a speed limitation of V_{LIMIT}^n, with a region of interest of RoI^n, and a time for the ego vehicle to cross safely the intersection of $\rho_{cross}^{r_{int}^n}$, and intersection point of roads ξ_{safe}^n, the region of interest was described as an extension of the road topology with the same width of the lane, $\delta_{RoI^n} \in \left[-\frac{\delta_{lane^n}}{2}, \frac{\delta_{lane^n}}{2}\right]$, longitude from the intersection to the look ahead needed distance with orientation γ^n, $RoI_{distance}^n = \left[0, d_{cross}^{r_{int}^n}\right]$, defining such distance as:

$$d_{cross}^{r_{int}^n} = V_{LIMIT}^{r_{int}^n} \rho_{cross}^{r_{int}^n}$$

With $\rho_{cross}^{r_{int}^n}$ as function of the speed of the ego vehicle, v_o^{ego}, and the distance to the first space in r_{main} for the ego car to be safe after crossing the intersection or incorporation, $d_{safe\,place}^{current}$:

$$\rho_{cross}^{r_{int}^n} = d_{safe\,place}^{current} \Big/ v_o^{ego}$$

As a result for our region of interest we merged all the results hitherto $Z \equiv RoI_{ideal}^{current} = \left[RoI_{ideal}^{main} \cup RoI_{ideal}^n\right] \Leftrightarrow \exists RoI_{ideal}^{main} \ni \xi_{cross}^n \forall n$ with n being all the roads that have an incorporation or intersection on main road in the nearby states. If $\nexists RoI_{main} \cap \xi_{cross}^n \forall n \Rightarrow RoI_{ideal}^{current} \equiv RoI_{ideal}^{main}$.

For the feasible RoI, we need to check the occlusions from the different dynamics ODD that block the perception for the sensor, for a sensor i with a range $\sigma_{range,max}^i$ exposed to some occlusions from different natures,$\sigma_{occlusions}^i$, the true range for that sensor is $\sigma_{range,real}^i = \sigma_{range,max}^i - \sigma_{occlusions}^i$. Thus, $\sigma_{range,real}^{ego} = \bigcup \sigma_{range,real}^i\ \forall sensor\ in\ the\ ego\ vehicle$is the addition of all the perception of the ego vehicle.

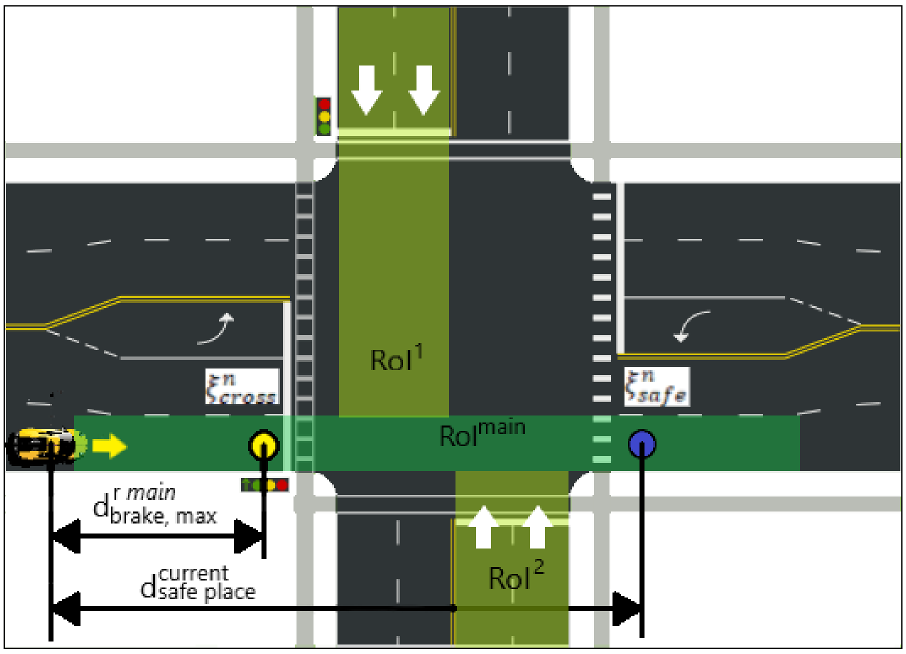


Figure 14 RoI Definition

Feasible RoI is defined as the intersection of the ideal RoI with the real capacity from the sensors, $Z_f \equiv RoI^{current}_{feasible} = RoI^{current}_{ideal} \cap \sigma^{ego}_{range,real}$. Define action contracts from here depends on the nature of the object and situation.

When weather conditions or static ODD block the sensors preventing from checking the feasible RoI, the speed is reduced, until the feasible RoI becomes safe again. Hence, from $RoI^{current}_{feasible}$ we obtained the maximum distance visible from the sensors $d^{ego}_{visible}$, and calculate the maximum speed to be able to brake on time in case any obstacle appear.

$$\frac{Weather}{Scenario} : v^{ego}_{ideal} = \sqrt{2a^{max}_{brake} d^{ego}_{visible}}$$

When a moving obstacle constrain the vision, and, in turn, the feasible RoI, it also implies that the $d^{r_{main}}_{brake,max}$ is not being respected, therefore, the speed should be reduced consequently.

$$Moving\ object : v^{ego}_{ideal} = \sqrt{2a^{max}_{brake} d^{r_{main}}_{brake,max}}$$

Lastly, in the cases of lack of visibility in another road incorporation or intersection. When the feasible RoI is not safe enough in an intersection or incorporation, the car should be able to stop before reaching any dangerous position.

$$\frac{Intersection^n}{Incorporation^n} : v^{ego}_{ideal} = \sqrt{2a^{max}_{brake} d^{ego}_{cross}(\xi^n_{cross})}$$

In all these cases, we considered that the feasible RoI was not safe enough if:

$$RoI_{\%} = \frac{RoI^{current}_{feasible}}{RoI^{current}_{ideal}} \% < Safe\ threshold$$

In those cases the speed constrain was activated as a risk assessment measure. If$v^{ego}_{o} = 0$ $and\ RoI_{\%} < Safe\ threshold$, an improving feasible RoI behavior should be implemented to react properly on those situations, avoiding any dead ends that may come.

6 Matlab Simulation

On the first stage of the testing for the Decision Making algorithm several Toolbox from Matlab will be used. All the programming will be carried out in the programming language of Matlab and the test will be represented by the different ways Matlab allows users to represent data. For the investigation purpose a professional license of Matlab was acquire with following Toolbox:

- Matlab version 9.5 (R2018b)
  - https://ch.mathworks.com/help/matlab/index.html
- Automated Driving System Toolbox
  - https://ch.mathworks.com/help/driving/index.html
- Computer Vision System Toolbox
  - https://ch.mathworks.com/help/vision/index.html
- Deep Learning Toolbox
  - https://ch.mathworks.com/help/deeplearning/index.html
- Image Processing Toolbox
  - https://ch.mathworks.com/help/images/index.html

Through those links all the applications, examples and functions provided by Matlab that can be used can be searched and understood.

Even when Matlab is not going to be implemented in the real scenario due to big computational load required for the program, the access to this kind of Toolbox accelerates the testing and the scenario creation. As it was explained, the main goal of this research is to improve the safety of the decision making in critical scenarios. Therefore, creating scenario with critical circumstances is an important part of this process. Matlab uses an exclusive language that may be found similar to C++, for that, converting Matlab Scripts to a ROS understandable language (C or Python) may take some time, but can be done with no difficulties beforehand. Also Matlab includes a Toolbox that allows connection with ROS that grants faster and easier test in a real environment.

As Decision Making includes different behaviors and several problems to be solved, the developing of the algorithm have been chopped down in various stages. All those stages have different upgrades for the general code. Not all block are connected and/or use in the final version of the program. Some of these functionality were included for debugging purposes or were not improving the performance of the whole. The following picture serves as a blueprint of the developing of the Matlab code from the bottom to the behavior to be implemented in the next stage of the research.

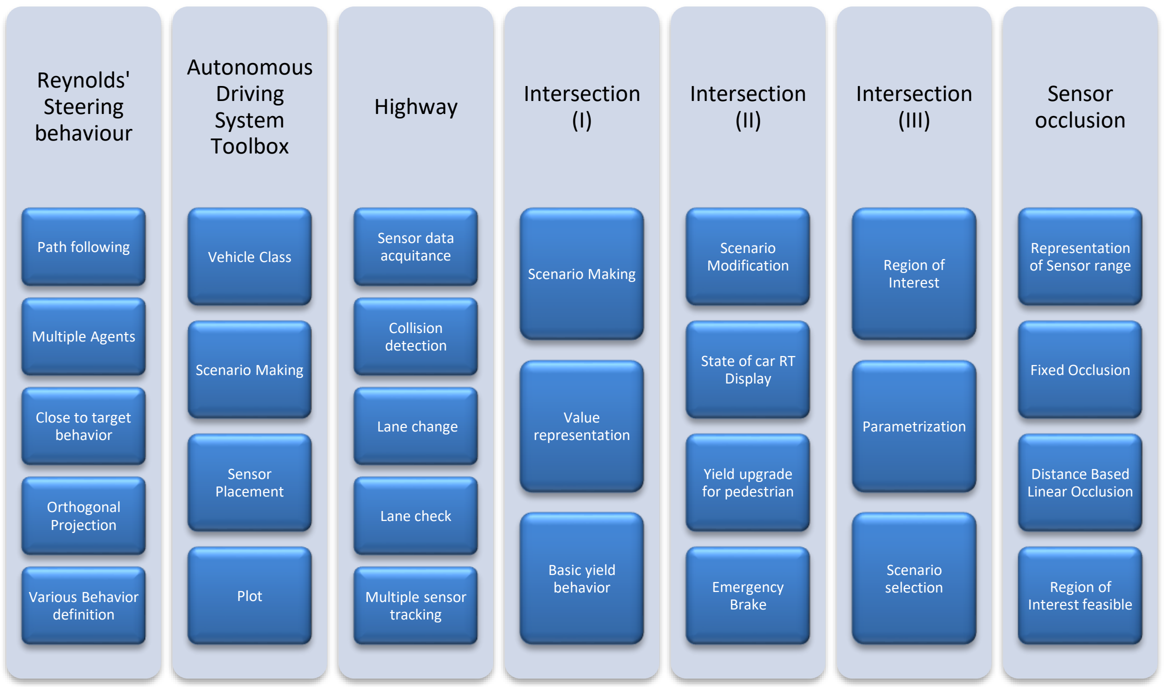


Figure 15 Matlab program blueprint

As it can be noticed, there is only implementation for two scenarios (highway and intersection), and only one is implemented with all the features (as it can be seen in the coming sections) and this is because the main goal of this investigation is to improve decision making by adding an additional layer of security to check the safety in a decision and thus choose the optimal choice for that situation, based on the information we are actually able to recollect. So for each one of the situations you want to test this program, you also need a base of decision making for the background, which is going to decide how the vehicle is moving, to which objects is paying attention the most, among other options to be taken. For that, we chose to focus on one of the most complicated scenarios and base all our test there, leaving the window open for other scenario's testing in the future. Hence, focusing on one scenario as all the function are done by parameter that can be easily modify, made the development of the code faster than it would have been to develop a fully functional decision making, which may take years to perfect.

One thing to take into account is that, even though all the code of this examples is based on a Finite State Machine, as all the functions were made to be implemented in various systems, this code may be used in a decision making neural network, for instance, as long as the system is able to identify all the information from the environment required by the code. The code may need some adaptation from one system to other depending on sensor configuration, behavior definition and traffic rules (side of the road for driving, maximum speed in Km/h, priority in intersections… etc.) and can be also personalize by changing some parameters free to change by the user as it is the threshold of security or braking distance, among others.

With all this information checked, all the different modules of the code can be studied independently in order of creation, as the results are being put together towards the final destination.

6.1 Reynolds Steering behavior

When starting from scratch this problem, there are innumerous flanks to cover, and too much work ahead to do. Nevertheless, there are some of those problems who need to be attached first.

The main problem to be treated at the start is to create a simulator of the movement of the vehicle. How is going to move and what kind of inputs it need to move? In this case there was chosen an acceleration in each sample time that updated the velocity and therefore, the position. Even when the car does not use this kind of inputs; basically they use acceleration, braking and steering wheel; a transformation between these inputs is easy to do, so the easiest implementation was chosen.

Once we have a simulator of the movement for the vehicle (the vehicle does not have a body yet, but a coordinates in space, thence, is more a particle than a vehicle right now) we need to choose a behavior for it. One easy-to-implement management for the acceleration of the vehicle is the Reynolds' steering behavior. In this technique there are three disparate attitudes for the particle to take: flee, seek and pursuit. Flee is used when running away from some point, seeking is when trying to get close to that point and pursuit is seeking a moving target predicting future positions. Based in our situation, the demeanor which fit the most is seeking a target which would be a destination point in a map. At this early stage of development we may consider our vehicle as an omniscient particle that knows its position and all the positions from the other actors and beacons. In this case we can have our seeking to the destination point from the current position until we reach the desired state. In the next figure, Figure 16 Reynolds' steering behavior there is a scheme of how this technique works.

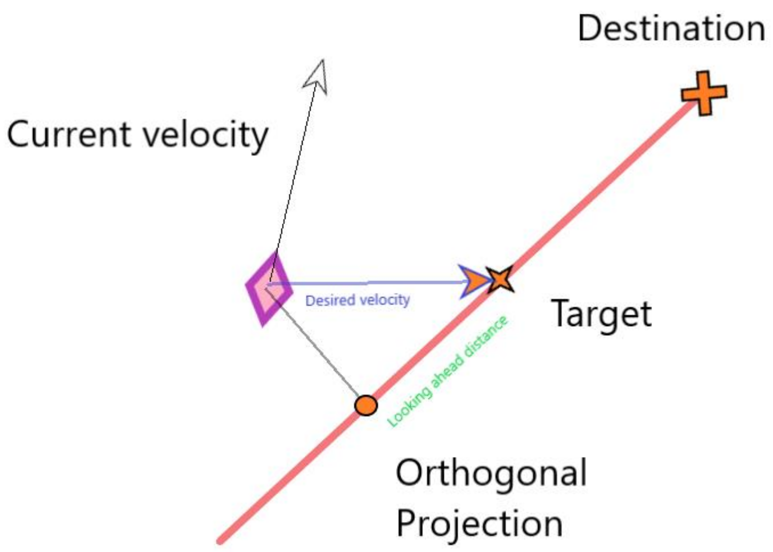


Figure 16 Reynolds' steering behavior

We can see an object, purple rhomboid, with a vector of its current velocity, $\overrightarrow{v_c^{ego}}$, based on egocartesian reference system. Egocartesian axes are the ones place with the origin on the position of the vehicle and with the X axe pointing to the car moving direction and the y axe at 90 degrees.

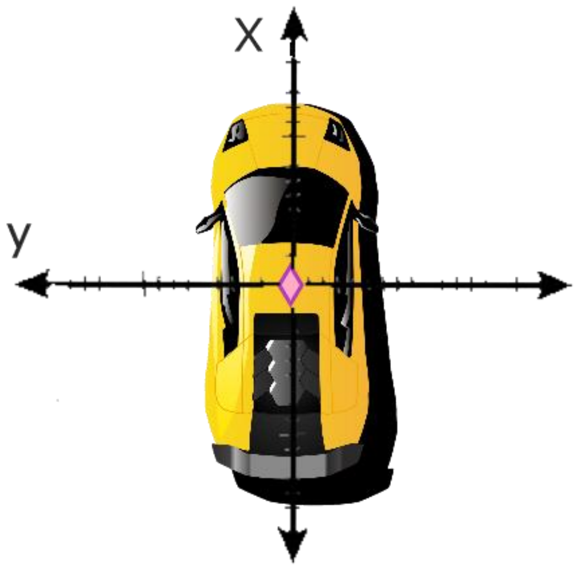


Figure 17 Egocartesian representation

There is a salmon colored lane which represent the intended trajectory of the object, with a final pint defined as destination. The direction of that vector is represented by $\overrightarrow{traj}$ and is represented in absolute axes. Perpendicular to that vector in the position of the vehicle we can acquire the orthogonal projection of the vehicle position in $\overrightarrow{traj}$. In case we want to seek destination point we can just go ahead to that point, but that will not follow the route but just assure we reach the last point.

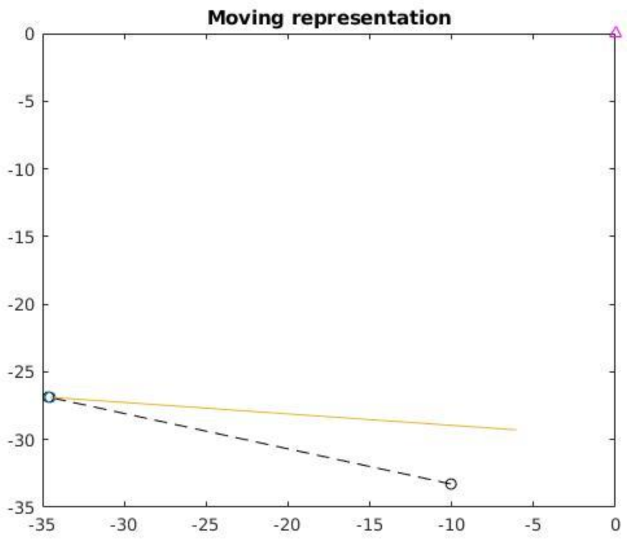


Figure 18 Seeking destination without mid-way targets

In the previous figure you can see how the vehicle, represented its trajectory through the experiment with the yellow line, moved from the initial state to the destination without getting closer to the trajectory, represented as a discontinuous black line. To avoid this we use the orthogonal projection of the position in $\overrightarrow{traj}$, $\overrightarrow{OP}$. If we were going to chase the orthogonal projection, the vehicle would never move forward, that is why you need to choose a target in the vector of the trajectory.

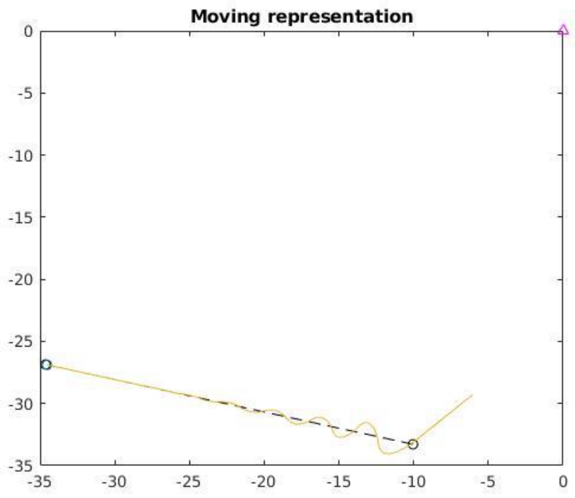


Figure 19 Seeking with few ahead vision

As you can see in the previous image, with a close target in the trajectory, our movement becomes unstable and takes longer for the trajectory to adapt to the route. Also reduces the smoothness of the movement. By choosing the right parameters a smooth and stable path can be followed through the route, and depending on the situation those parameters can be change to optimize the behavior. In the following picture there a current speed linear variable was used to keep the route.

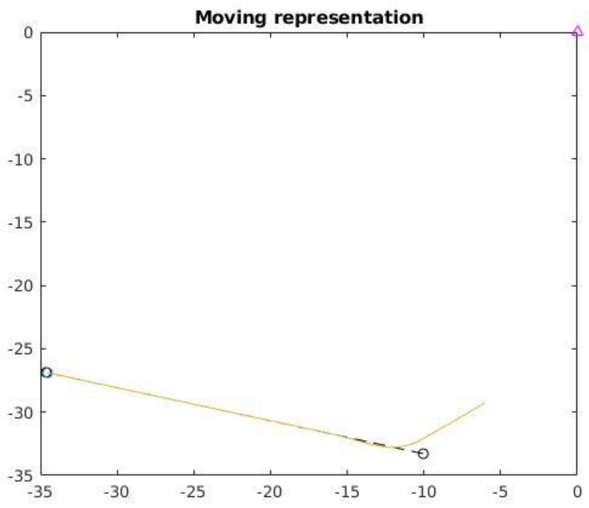


Figure 20 Seeking target looking ahead

With that all the bases to follow a path between a starting point to a destination, and the way to do it is by using the equation of Reynolds for autonomous particles. Looking in the next picture all the important values can be seen with a graphic demonstration. From those values, performing some operations, the acceleration can be calculated.

$$\overrightarrow{Target} = \overrightarrow{OP} + V_{MAX} * T_{Sample} * \frac{\overrightarrow{traj}}{|\overrightarrow{traj}|}$$

$$\overrightarrow{dir_{desired}^{ego}} = \overrightarrow{Target} - \overrightarrow{p_{ego}}$$

$$\overrightarrow{v_{desired}^{ego}} = V_{MAX} * \overrightarrow{dir_{desired}^{ego}}$$

$$\overrightarrow{Steer} = \overrightarrow{v_{desired}^{ego}} - \overrightarrow{v_{ego}}$$

$$\overrightarrow{Acceleration} = Torque_{MAX} * \frac{\overrightarrow{Steer}}{|\overrightarrow{Steer}|}$$

With these simples operations we can calculate the acceleration to apply for the vehicle. In the last operation we use a parameter called maximum torque for the vehicle. With that we can restrict the speed increase and the running of the car. This is not applicable for cars when it comes to lateral sliding, but with a simple restriction in the code we can keep this movement as one similar for vehicles. If we choose a big value for the maximum torque, we would get an ideal movement.

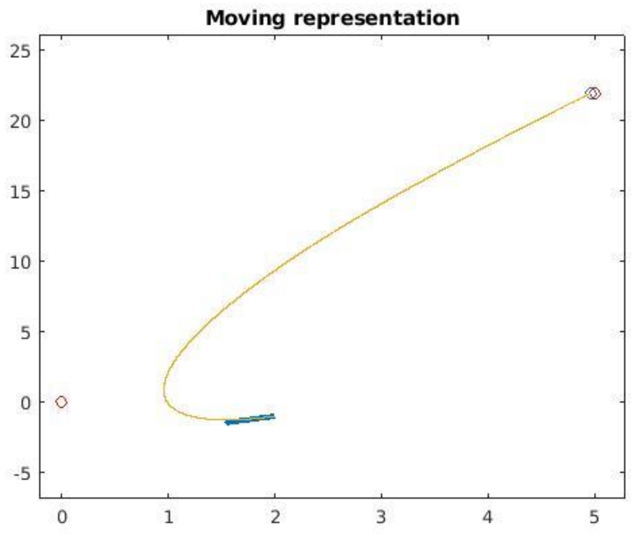


Figure 21 Ideal target seeking

In the previous picture, a vehicle with an starting velocity, represented by the blue arrow in the opposite direction than the destination, with a big value of maximum torque can turn with a really obtuse angle, unlike a vehicle would do it. To accurate this movement to one more realistic, an experimental parameter was chosen by submitting the vehicle to several experiment. Even when in a static picture may seem that the following picture turn is more obtuse, the speed is lower in the moment previous to the turning.

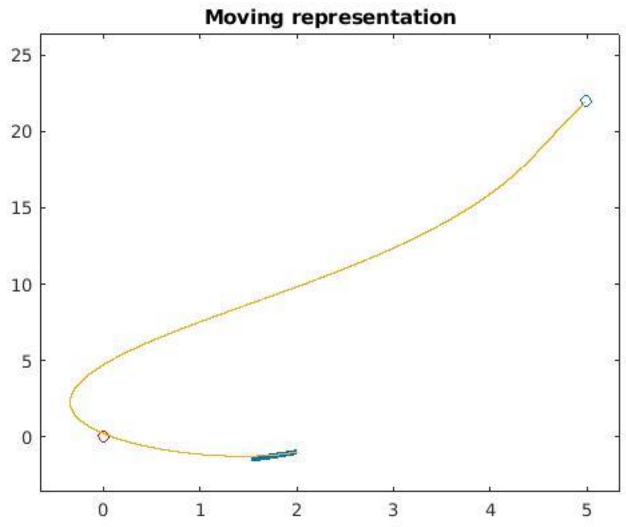


Figure 22 Realistic target seeking

From this point, the next step to be performed by the car is not to follow only a line from to points, but to keep going through several waypoints, and as we cross them, change for the next one. One of the main things to take into account here is that this technique is good to approach to the surroundings of the point, but after we reach the point, there is nothing that make car stop in that point. For the previous experiments the simulation was stopped when reaching some area nearby the destination point, but when working with different points, these event may trigger instabilities in the corners, as it can be seen in Figure 23 first try: Total failure.

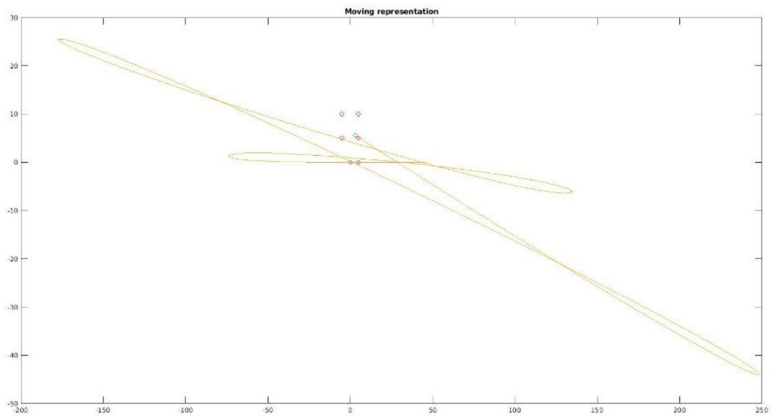


Figure 23 first try: Total failure

To avoid this phenomenon, once our prediction in the trajectory, target, have reached the destination point of the first stretch of the route, we start looking to the next stretch of the route, so the car star turning before reaching the corner and not once it is on the corner, which indices oscillations on the movement. Another arrangement done for the improvement of this movement is to reduce the speed near the corners, as a human driver would do, and not to take turn at maximum speed, but at 25% of it.

Once we have one particle/vehicle capable of going through different waypoints, instead of choosing random spots in a blank space and make the vehicle to run them, we decided to use one map from the surroundings of the Oomika Station, in Ibaraki, Japan. But to use the map in Matlab as a series of points there is a need for another package from internet. Using the Open Source Package of functions from OpenStreetMap, the following map could be turned into various nodes, being aware of cross and lane directions, obtaining a route.

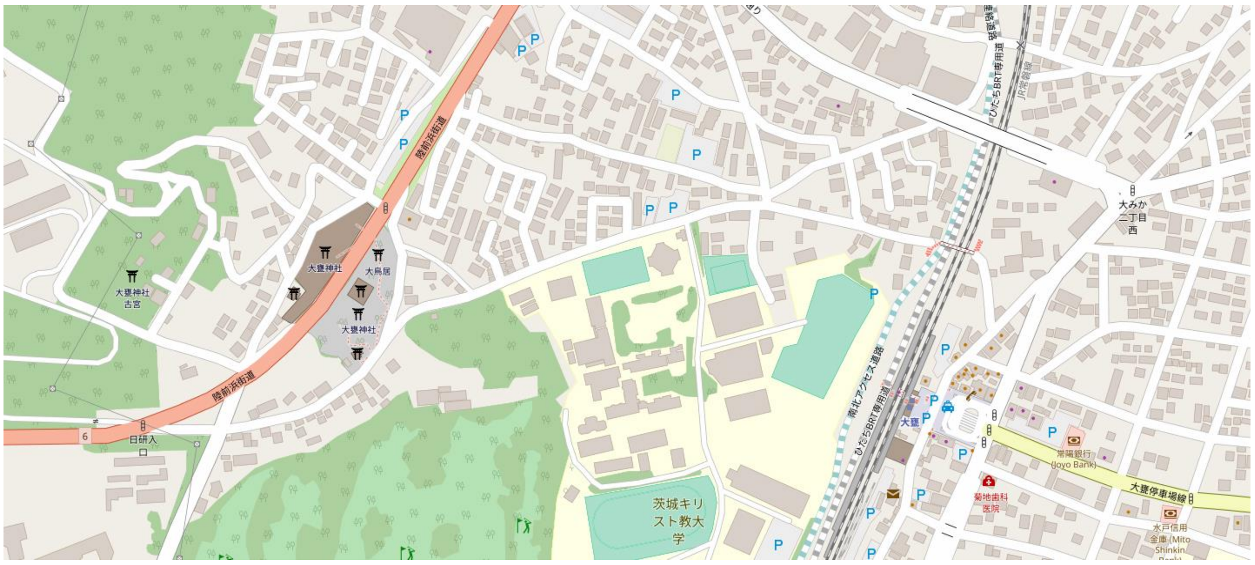


Figure 24 Map of Oomika Station surrounds

From those points, after doing a transformation to be used by the vehicle (as the points were measured in latitude and longitude, some of the accuracy was lost in calculations by the declaration of the variables), the vehicle followed the route, represented by the discontinuous red line, moving as show in the black line. The sample times of these experiments were no chosen accordingly to the real one needed.

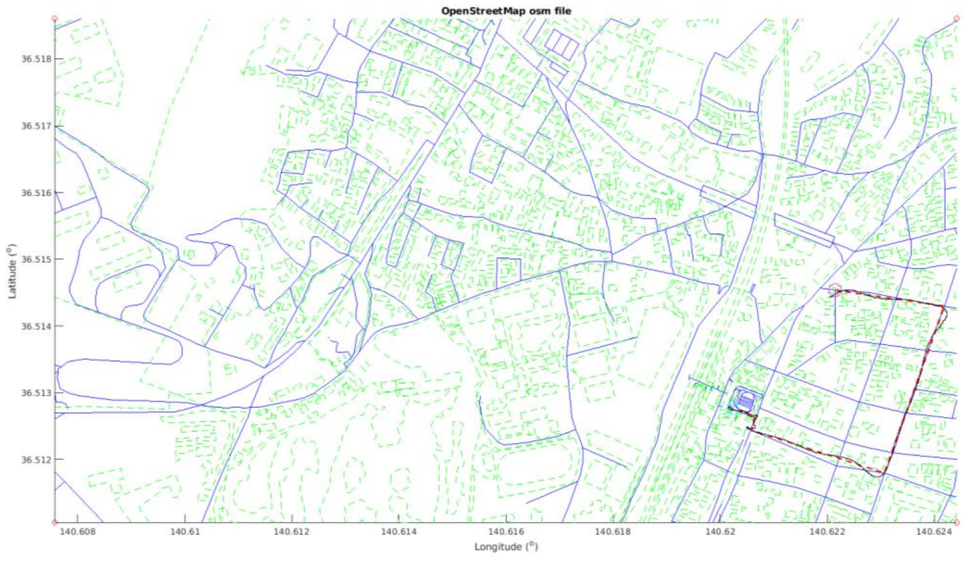


Figure 25 Real map based consecutive target seeking

From the same map, other routes may be taken for different vehicles in a scenario. When setting up the scenario, all the car or vehicles must have a behavior determined, but it does not have to be the same. Therefore, two demeanor have been chosen. One of them, aggressive behavior, is avoiding bumping into other actors in the scene by turning the car to the opposite side of the incoming car. The other one, passive behavior, will yield when crossing with another actor. All the vehicles in the scene are suppose as all-knowing actors, hence, they all know at all tie when are the rest of vehicles. As showed in the Figure 26 Multiple vehicle path following. Steering Behavior. a test was run, were all the vehicles were repeating in loops their routes. All the routes have a various length, creating differences in the scenario in each loop. This would repeat for a fixed amount of time for the purpose of the testing. [4]

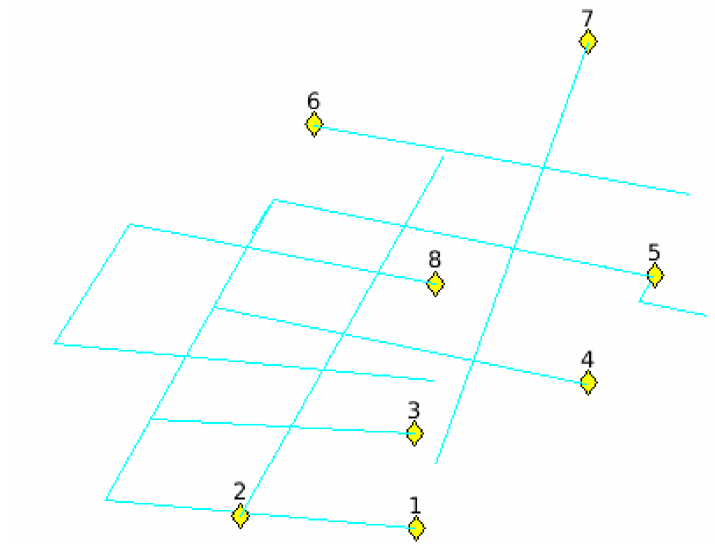


Figure 26 Multiple vehicle path following. Steering Behavior.

In this experiment, the vehicles 1 and 5 have to cross path, therefore both of them are using aggressive behavior as there is only one lane to drive through. The rest of the vehicles are using the passive behavior. For reality purpose, there is an invisible area surrounding the vehicles that acts as their bodies, and if two of them get in touch, a collision error pops up and stop the reproduction of the experiment. Without this feature is not possible to debug problems in the behavior as there is no way to check if two vehicles have bumped into each other.

[4] **The whole video of this experiment is disposable in the attached files by the name of "".**

Besides that experiment focused on intersections, another test were run in diverse scenarios. For example, in Figure 27 Incorporation experiment. Steering Behavior an incorporation to a highway from a secondary road, including some time to check the lane to see if there is any car coming toward our designed spot and some prediction of the car to see if there is time to incorporate the road when a car is coming slow or from a far distance. In these experiments, we will be focusing on a single car, denominated ego vehicle, and the other actors in the scene will be attach to a fixed route. In the same way we did in the previous experiment, in this experiment also all the vehicles return to their initial state after reaching the destination point, creating anomalies in the original situation that allows us to prove extra situations.

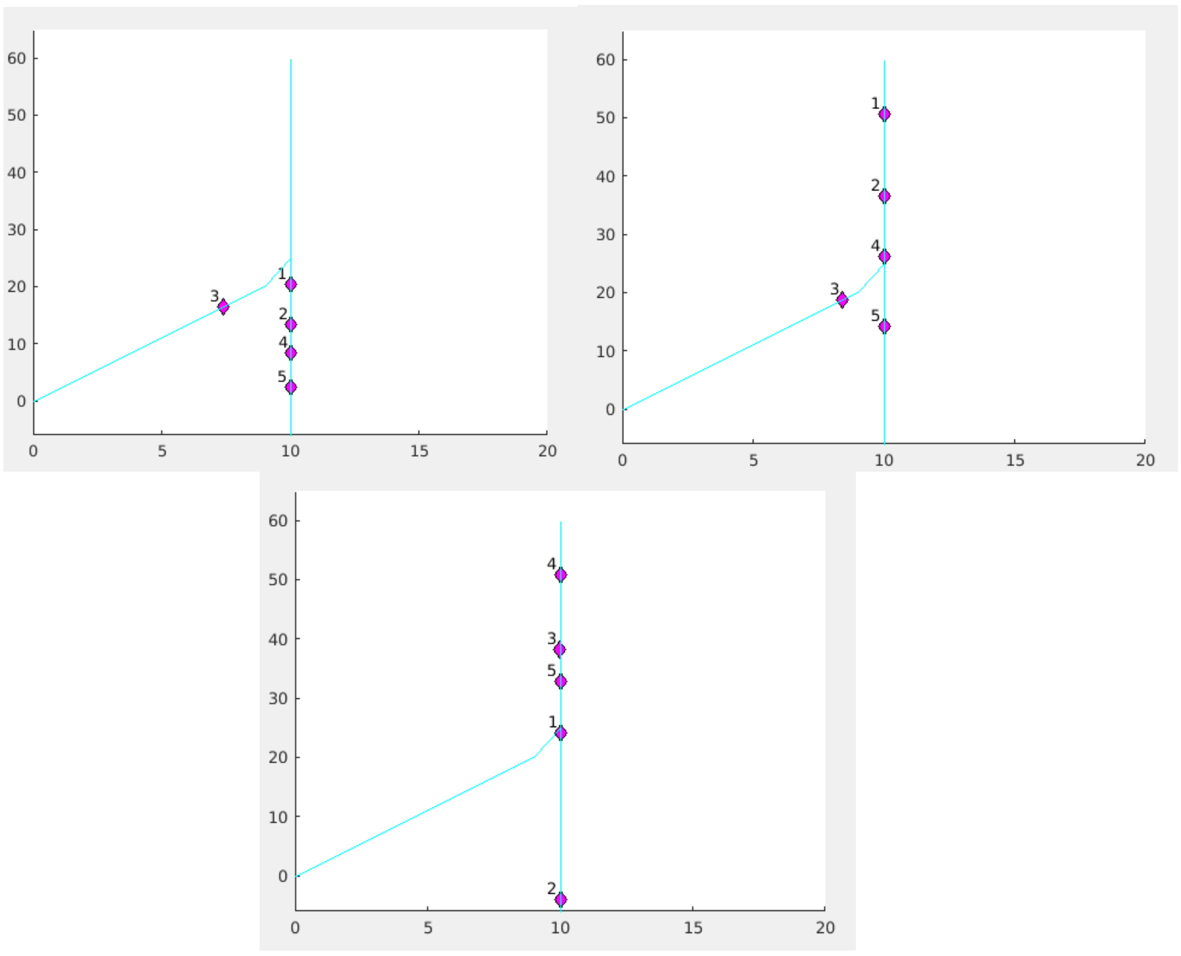


Figure 27 Incorporation experiment. Steering Behavior

In this experiment there are several car going through a highway (1, 2, 4 and 5) and there is road that incorporates in that highway. From that secondary road there is our ego vehicle coming. Our ego vehicle is going to slow down as he is approaching the incorporation to the road. If when doing so captures any car incoming from the right that might enter the cross before him, will stop. To model the lack of visibility of the sensor, a variable was created. This variable was increase as close the car were to the corner. In case this variable was not beyond a certain threshold the car will not incorporate into the main road. In this experiments we can see how the car stops for the first two loops (upper images) as there is no space enough, but on third one it enters between the vehicles 4 and 5 (lower image). In case this maneuver was considered too risky, by changing the threshold and prediction of other vehicles, it could be reduce the aggressiveness of condition of our ego vehicle.

With this it finish the first part of the structure in the development of the program. Until now we have manage to obtain an autonomous particle seeking a target, which is able to evaluate the situation in the surroundings and react in consequence in a simple basis. Starting from that a much more developed and critical safety environment focus system can be researched.

6.2 Autonomous Driving System Toolbox

For deeper studies in this area a special Toolbox of Matlab is going to be used. Autonomous Driving System Toolbox is specially designed for self-driving vehicle, proving a wide range of scenarios thank to all the utilities, functions and applications. For further information check the previous section were all the links to the toolbox are placed.

The first thing were this toolbox come in very handy is when it comes to scenario making. Along with all the functions included in the toolbox there is a scenario maker with a series of features in it. In the following picture, Figure 28 ADST: Menu Bar, we can see all the objects and actions available in the App. It allows us to design roads and different actors, include sensors on those actors, run simulations on the own application and export data. The exportation data may be two different nonexclusive sets, one of them give us a script in Matlab with the declaration of all the actors, objects, roads and trajectories defined in the application. The other set can only be obtained after adding some sensors to the ego vehicle and running one simulation. After that all the data taken from the sensors can be exported to be used for later processing. It also allow to change some stings of the simulation such as sample time or simulation time among others. It can also be executed stopping in each sample time to observe more in detail

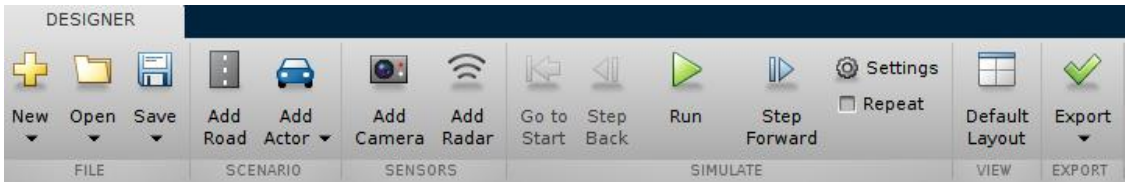


Figure 28 ADST: Menu Bar

As for the actors is not only made by one ego vehicle, there are various kind of vehicles and other objects to be set in the scenario, granting more scenarios to be created with miscellaneous characteristics.

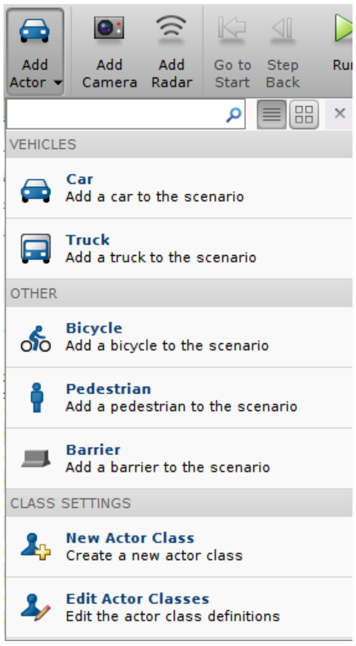
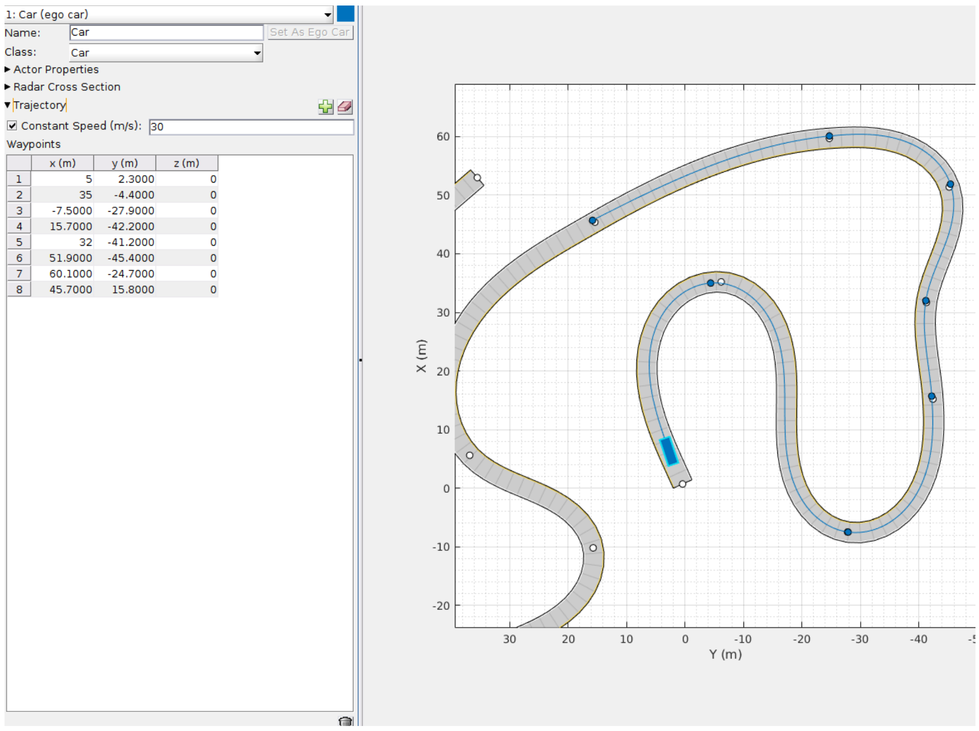

Add Actor
Add Camera
Add Radar
Go to Start
Step Back
Ru
VEHICLES
Car
Add a car to the scenario
Truck
Add a truck to the scenario
OTHER
Bicycle
Add a bicycle to the scenario
Pedestrian
Add a pedestrian to the scenario
Barrier
Add a barrier to the scenario
CLASS SETTINGS
New Actor Class
Create a new actor class
Edit Actor Classes
Edit the actor class definitions

Figure 29 ADST: Actor creator Menu

All of the objects shown in Figure 29 ADST: Actor creator Menu can be modified in several ways. In nonmoving objects such as barrier you can only modify the shape. In other objects alongside with the shape you can also change its velocity, trajectory, starting point and radar and camera errors when being detected. Apart from this you can also create your own actor with unique properties.

To create roads and trajectories for actors the same process is carried out. You select the waypoints or road centers and then an interpolation among them is done to create a smooth line to be followed for the vehicle. All the position, velocity and space values in this simulator uses a 3D axis, but to simplify all control signals, in the test only x and y axis were used (as explained in the section Reynolds Steering behavior). When choosing the same points for both the road and the ego vehicle, the trajectory of the car fits with the road center for the whole path.


1: Car (ego car)

Name: Car

Class: Car

Actor Properties

Radar Cross Section

Trajectory

[✓] Constant Speed (m/s): 30

Waypoints

	x (m)	y (m)	z (m)
1	5	2.3000	0
2	35	-4.4000	0
3	-7.5000	-27.9000	0
4	15.7000	-42.2000	0
5	32	-41.2000	0
6	51.9000	-45.4000	0
7	60.1000	-24.7000	0
8	45.7000	15.8000	0

Figure 30 ADST: Trajectory Definition

Once the car and other objects are settled, with their trajectories define and road is fit for the path they are going to follow (in this simulator there is no possibility of adding some control to the vehicles such as lane keeping algorithm, hence the trajectory needs to be adjusted to fit in the road center, otherwise experiments would look strange) it is needed to place and define the parameters of the sensors. In this Toolbox there are two types of sensors accessible: radars and camera. The camera works as a real camera with the data from the experiments and situates the objects in the middle of it. Techniques for the camera can be also be used in real camera data to detect objects and obtain bird eye plot. The radars in turn detects in discrete ranges among a certain angle. Thereupon same object may obtain multiple detections from a single radar, each one separate from the others a certain minimum angle that the radar is capable to disclose. Each sensor can be place in any part of the car (even some part outside the range of the own car) pointing to any direction desired. Hence, azimuth, yaw, maximum angle, range and position can be changed at ease. In Figure 31 ADST: Sensor placement a basic two front radar and one camera structure can be seen, and can be tested in early stages of this simulations.

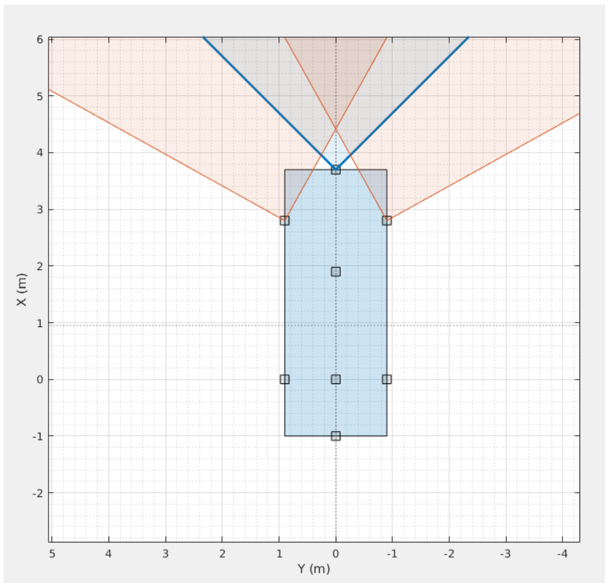


Figure 31 ADST: Sensor placement

With those sensors placed as shown, a scenario with other actors inside a road can be structured, to start testing how data are collected by those sensors. In Figure 32 ADST: Sensor Data Acquaintance there is a simple test being done with this sensor configuration. Two vehicles, ego vehicle and a fixed-trajectory actor, were arranged in a circular loop road, having the ego vehicle running in the opposite direction than the other vehicle.

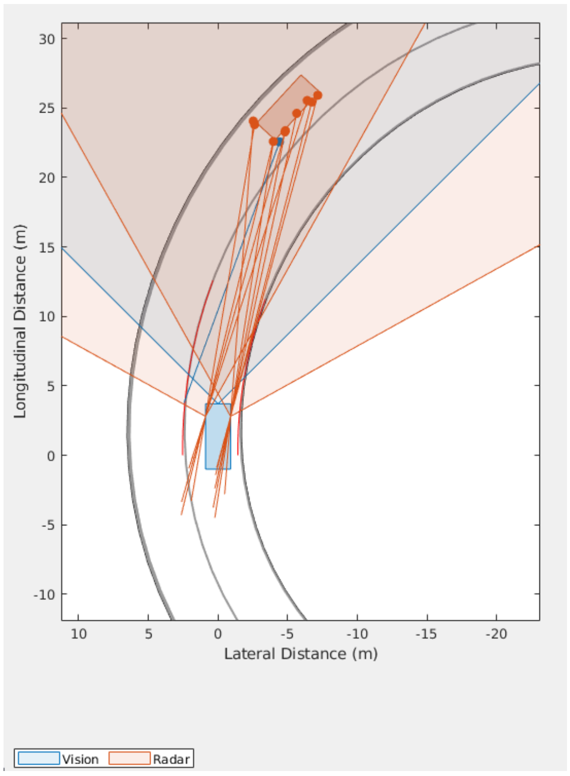


Figure 32 ADST: Sensor Data Acquaintance

In the figure there are 8 detections from the radar and one made by the camera. From now on, all the experiments will use the same color pattern, all the detections from radars and range of them will be painted by a light red and for cameras will be painted by a navy blue. The camera is only one detection allocated in the middle of the object as it is seen by the ego vehicle and does not give us any other information but the coordinates of that objects at first. It can be changed to be able to detect type of object, lanes boundaries and also speed. For radars, in the other hand, there are multiple detections and there are lanes coming from them. Those lanes are graphic representations of the speed of those points by the egocartesian axis. As we have multiple data from the same object but the radar repeatability is not reliable, we can conciliate all data from sensors to have one more accurate acquisition.

With all this information, we are ready to start developing a fully equipped environment for test and behavior planning. Starting for the easiest one, to increase gradually its difficult. We have chosen highway driving as a starting point as it is reduced the directions from where the danger may come from. Also, to reduce difficulties in programming the behavior a straight lane have been chose so the first steps were easy to reach. With this a full directional sensor equipment was allocated in the ego vehicle to start the testing. Exporting all this information to a script in Matlab made possible to start working and modify some aspects in order to create more experiments to test the algorithm in. In Figure 33 ADST: Representation of experiments you can see a preview of how the sensor were placed and how the experiments were represented during testing and debugging of the code.

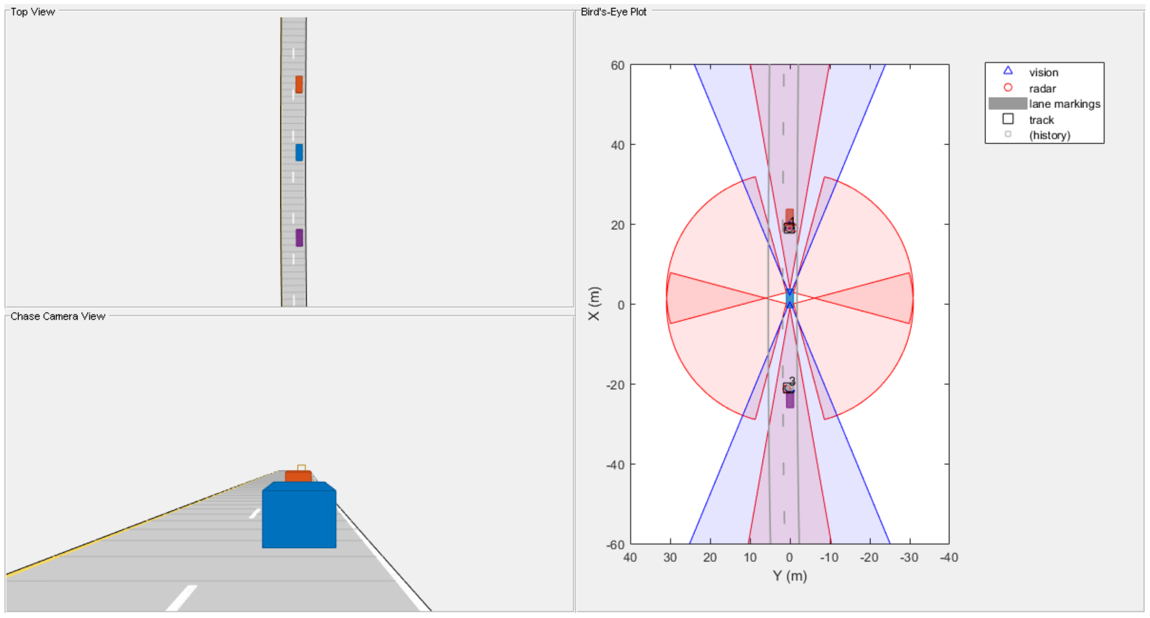


Figure 33 ADST: Representation of experiments

6.3 Highways

In the beginning of the experiments, a double lane straight road was designed as to move only in one axe in the absolute reference system, so all the movements were easier to program, as to lane change and acceleration and brake. In this moment one first challenge, not problem but lack of feature, was found. In the ADAS Toolbox there were no feature designed for collision among objects, hence, while doing the experiments everything could happen and the program will still keep running as usual, with the only exception of some measures in the sensors. While the sensors were inside other object a warning was displayed by Matlab but did not stop the execution of the code. Thus, a collision detection function was added to the code. As the only object without a fixed trajectory in the scene is the ego vehicle, only the collisions between this actor and the other objects have to be taken into account, as if the other actors collapsed is not a code problem but a scenario bad design. In a first instance an exhaustive point by point checker was designed but the simulation became slow as there were too many comparisons to be done in each sample time. Therefore a simpler check was done. From the center of the vehicle, a radius was traced with the highest distance inside the vehicle. Inside that area there were check all the position of the other objects to see if there were inside. One problem with this coding was the lateral crash, as they were obtained even when the ego car and the other object did not touch. For that, the radius were cropped to a certain percentage. This give us the collision warning that stop the running of the code for the situations, allowing a faster and more accurate debug of the code.

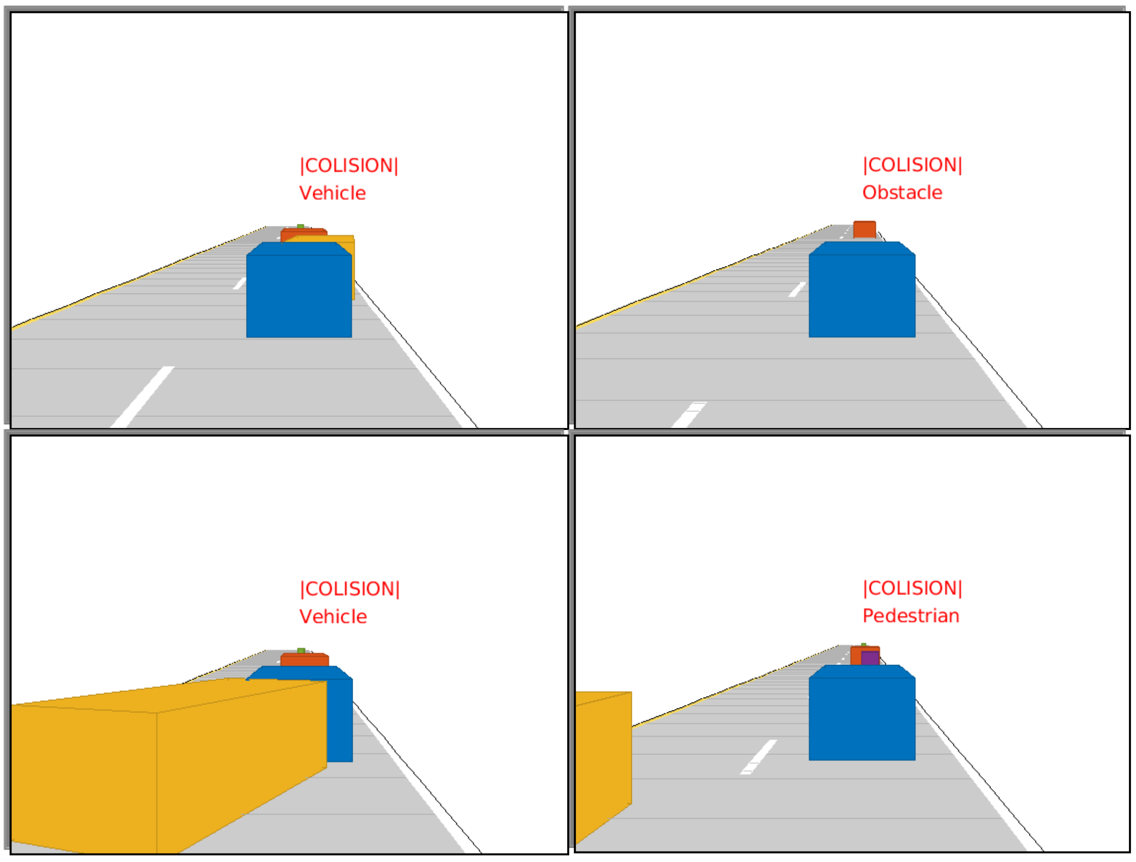


Figure 34 ADST: Collision

As it can be seen in Figure 34 ADST: Collision detection, keeping the trajectory of the ego car and placing some other objects in that trajectory it can checked that the code worked for collisions from different angles and in various heights. Also, to determine what kind of crash happened in the simulation, as sometimes can be varied reasons for a crash, also the class of the object that enter the collision radius is also displayed. This thing too helps to explain crash from blind spot as it happened in the right top picture in that figure.

One of the main changes made in the code while performing this development is the change of the sensor disposition. Up until now, the sensor were placed as explained in the section Autonomous Driving System

Toolbox, and there were a function from the same Toolbox used for the conciliation of the data. This function were given a smooth multitracking with a filter of all the data received from the sensors. Only, with the sensor error some of the objects were not being detected. This could be avoid in real life with the last sensor it was going to be included and it could not be used in simulation, the LiDAR. To add some feature similar to a LiDAR, or at least with the same range, a radar of 360 degrees and 100m range were place on top of the vehicle. As this sensor was enough to check all the surroundings, all the other sensors but the frontal camera were withdrawn. This would lead us with a configuration as shown in Figure 35 LiDAR-like radar configuration

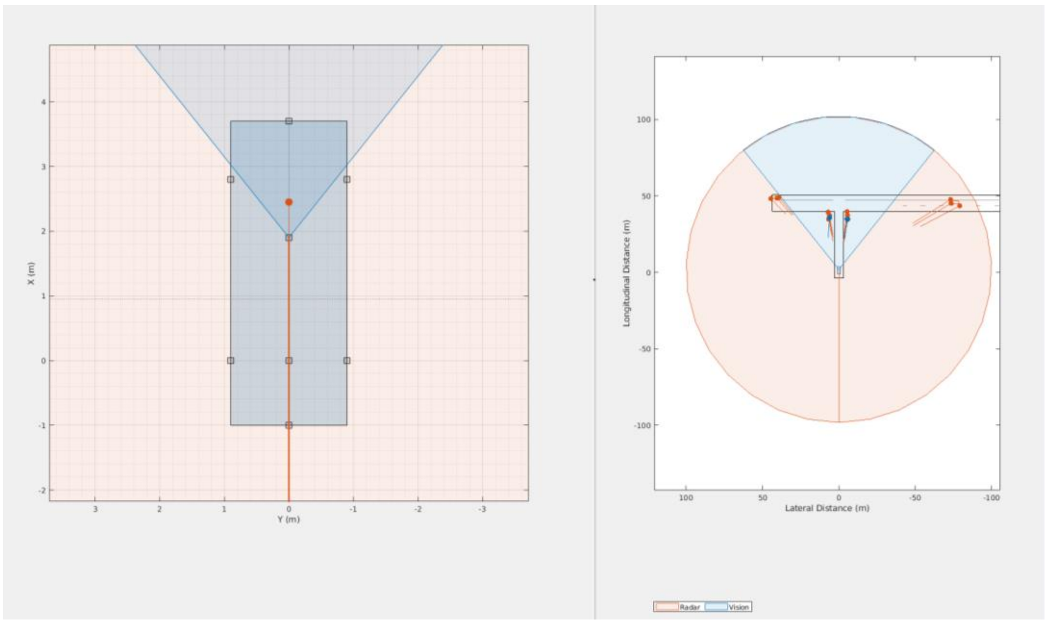
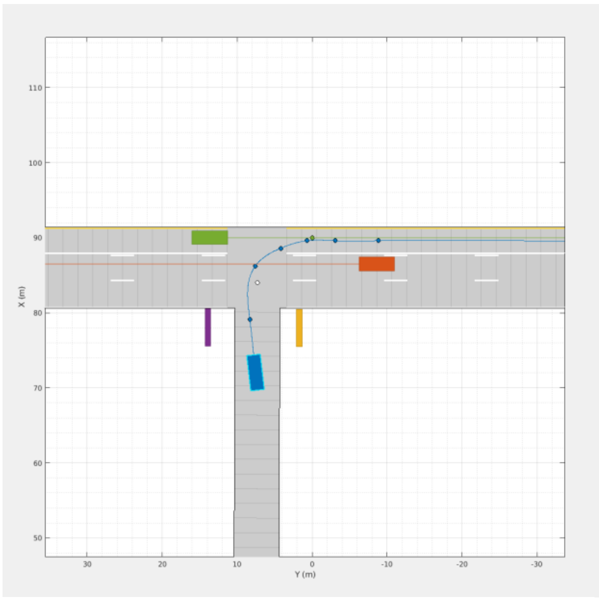


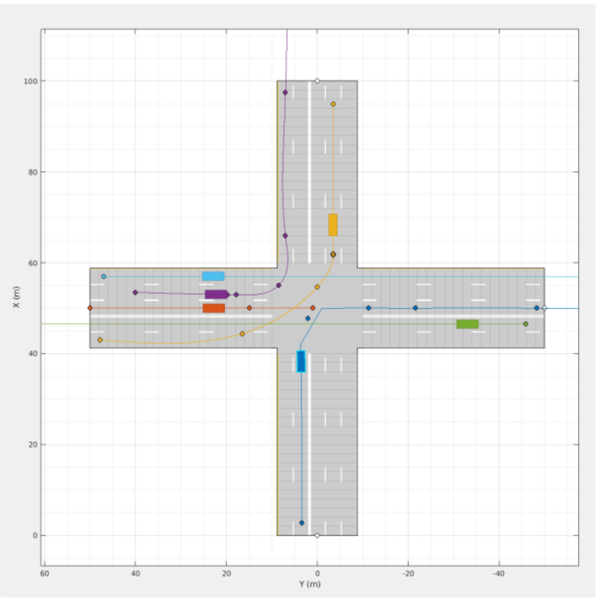
Figure 35 LiDAR-like radar configuration

As there were only one straight lane to follow, the Steering behavior was not included yet in these experiments. This code is the most primitive one in all the coding used in this research. A rotation matrix and a specific time were defined to do a change of road, working only in the designed scenario and in straight roads. As it has no scalability is not applicable to other scenarios but sets the background of the main code, what kind of loops to use and how to make decisions.

6.4 Intersection

Once we establish some background in the code, we can focus on our main targets. From now on our scenarios to be tested are intersections with different number of lanes in each direction and a T section as an





incorporation with low visibility.

Figure 37 T section initial trajectories

Figure 36 Big Intersection initial trajectories

In the T section there are two fixed obstacles with 2 meter height to block the detection from the sensors near the corner, testing decision in low visibility scenarios. As for the intersection, one with only one lane each direction and other bigger were designed to check the behavior with moving obstacles from different directions. The ego vehicle in those figures is the blue placed in the bottom part of the image, while the other colors represent the other car in the scenario. The line departing from each vehicle represent its trajectory, and the points are the waypoints. Even though a blue line is also represented for the ego vehicle, that is not the one it would follow, but the original trajectory, where the possible problems may appear. We will try to follow that trajectory as long as it is safe and feasible.

To check all the values of the variables in the car states and inputs such as movement in Y axe, speed or acceleration after the car have some problems (mostly bumping with other vehicles) these variables are shown at the end of the experiment, making easier debugging. Depending on the kind of experiment or part of the code that it is under testing, these variables to be displayed can be changed at ease. For some of the experiments would be attached to this document.

Firstly in this situation the steering behavior already designed was implemented in this code to follow the path made by the waypoints of the trajectory. Besides the change of variables and parameters, such as looking ahead sample time parameter or change of waypoint distance, to be treated with that algorithm, no further problems were encountered. Now we have a way to follow a route. Some of the trajectory needed some extra waypoints to smooth the movement alongside the route.

From that we can start creating a basic yield behavior. As for the start, we just check if there is any detection from the sensor near the cross which is moving with a speed higher than a threshold. To accomplish this in an adequate form inside society rules we need to reduce speed when getting close to the cross. Any speed below this threshold would consider as a static object in that sample time. As the velocity is obtained from egocartesian axis there is need to make some calculations before watching out about speed of objects. This behavior is not accurate enough when trying to catch pedestrian as the radar sometimes is not able to detect the pedestrian based on various reasons. Also for fully stop objects this demeanor is not acting properly.

To improve some of these flaws in the behavior, several adjustment were done. An emergency brake situation was added, checking 10 meters ahead of the vehicle to see if there was any obstacle, and in case there is, the vehicle will reduce its velocity, turning to stop the vehicle when the obstacle is close enough. Another thing to take into account is to add some border near the cross to add possible actors that may join the scene in the following moments than the one being hold in sight in that instant.

All this recognition around the cross are done based on the center of the cross and some information about the road, such as road width and ego car length. This parameter, cross center position, which is added by hand can be obtained by interpretation of the map or by some image detection in camera data. Although that process is not included within the limits of this project.

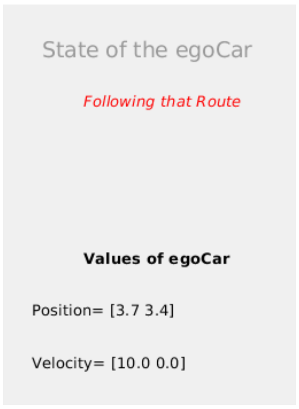
Other problem affecting our code is the lack of detections from the sensors in certain moments of some of the actor, giving us a blind spot where it should be one, a false negative situation. Usually this happens with objects smaller than usual or moving in a low speed, such as pedestrian or kids. Until now the car reduce its speed until some critical value when is almost stopped and check whether a car is coming or not. When the car is not detected within the range, the ego vehicle proceed to continue its route. To avoid this false negative detections, instead of taking that decision in only one sample time, some consecutive number of sample times without object detections. By testing its performance a proper number of sample times were determined to be able to be precautious while not losing so much time yielding in the cross. Other problem that may occur with sensors is false positive inside the range of yielding, implying as much useless waiting time as sample times were chosen for being forehanded.

	When there is an object	**When there is NO object**
When there is a detection	***True Positive***	***False Positive***
When there is NO detection	***False Negative***	***True Negative***

Table 2 Detection slang explanation

Also the parameter to stop the car near the cross were adapted to the map topology, as the distance from the cross center where the car should stop varies depending on the number of lanes present in the scenario. All this information is again hand inputted as there is no map recognition designed yet and that task lays outside the limits of this research.

Hitherto there are 5 different states that the car may be in each moment. The car can be either following the route, braking for an intersection, yielding, being cautious or activating the emergency brake. To help debug and visualization of experiments, there was a visual description added to the experiments in real time in the right part of the graphics. Likewise, other values were also added to the representation, such as velocity of the car or distance for the emergency brake.


State of the egoCar

Following that Route

Values of egoCar

Position= [3.7 3.4]

Velocity= [10.0 0.0]

Figure 38 Display of state of the car

From here, with a car that is able to react in a simple way to an intersection we start developing the idea expressed in the previous sections.

As mentioned starting this section, all the programs are going to be focused on two types of scenario, leaving any extension for the rest of scenarios up to future works. These scenarios we are going to be working with are the intersection-like scenarios, varying number of lanes and trajectory of the ego car. Now we need to develop some kind of region of interest for those kind of scenarios. For intersections we need to check the lanes that are coming to the cross. For that a simple condition-based code to create a region in space to show this section was created.

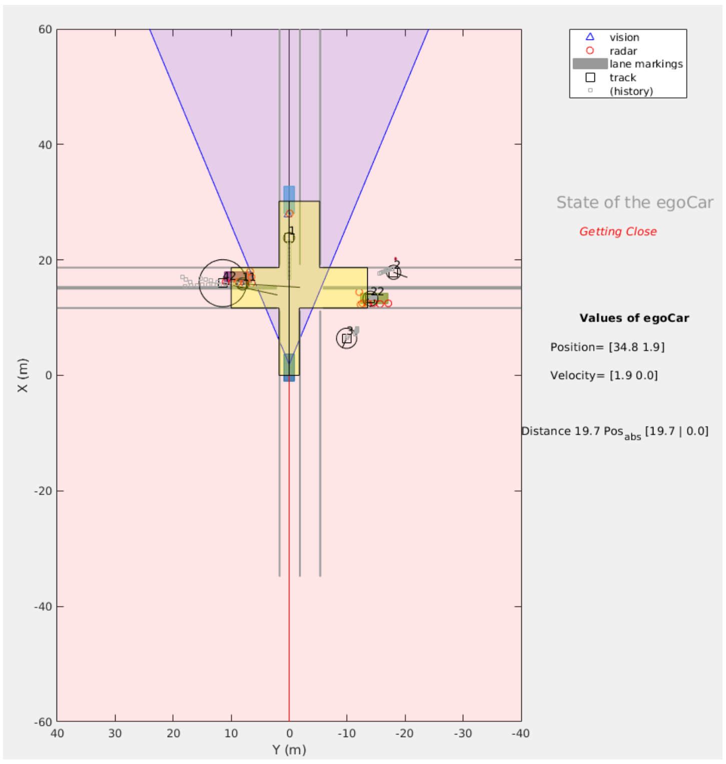


Figure 39 Small intersection. RoI

In case we are not close enough to the space in the cross, we were be checking a certain distance in front of us. This distance would be a safety distance for braking, calculated depending on the current speed of the vehicle. The faster the vehicle is moving, the larger the area should be.

The area for the Region of Interest in this scenario and the distance for the Emergency brake state to start is the same, so as soon as a car enters in the RoI, an emergency brake would start. It is important to note that all the decision taken around the RoI are user-defined and can be changed for an optimal actuation in certain situations, based on the opinion of the designer. In this case, the direction of the RoI is focalized on the desired direction of the car, also denominated as $\overrightarrow{dir_{desired}^{ego}}$. A minimum distance was added to the safe distance calculation, as if the car reaches really low speed or stops, it would reduce Region of Interest to a really small space or even null, which is not a desired act.

When creating the RoI definition, only one lane of each direction was taken into account and the direction of the ego vehicle was not used to determine the RoI needed. Also there was not paid attention to the fact of axis change between egocartesian and absolute. As the car start the turning the RoI also start turning to keep fitting the road, or so it should. In the next example it can be seen these problems with the first version of the Region of Interest needed for the car.

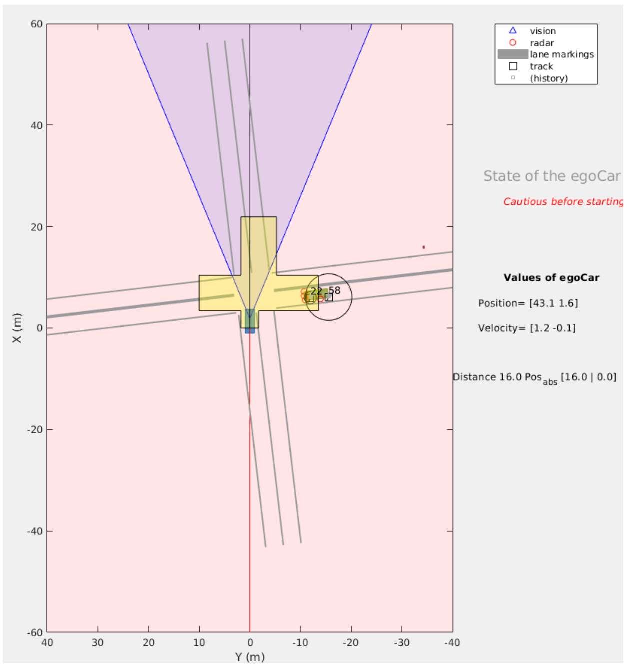


Figure 40 Small intersection. RoI bug

For a better parametrization of the approaching problem to the intersection, all the lane information was also input to the system so only the lanes where cars and other vehicles can come from, following the rules of driving, and a change of axis so the region moves with the car as this one turns. It is important to notice that all the programing involving this regions have been done by following the rules established in Japan, such as maximum speed, side of driving, yielding situations… among others.

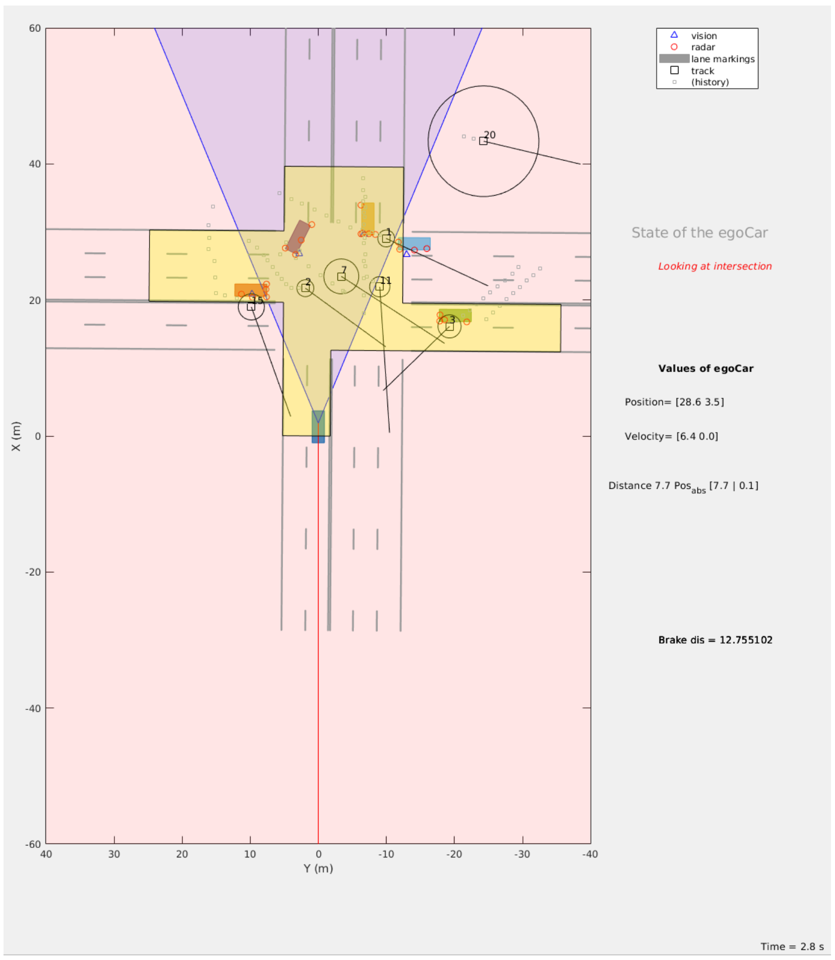


Figure 41 Big intersection. RoI parametrized

With this implementation it is possible to check which parts of the map we need to get view, yet we still cannot check which percentage of that area is it occluded for our ego vehicle. In this moment there were already several scenarios the program could work on and each one have its own and unique difficulties for the code. Changing among all the scenario required changing some parameters, the map loading, the actors update, maximum speed of the vehicle, the route and the initial position of the ego vehicle and the information about the intersection. All these changes implied a lot of time to change the test among scenarios, and more important and dangerous, a lot of change in the code which could imply in some typo mistakes that could lead to mistakes on the performing. To avoid further problems of these kind, a selector of scenarios was made between 4 different scenarios. Those were the scenarios selected as the most critical scenarios to test the behavior on. Those scenarios are the following: Small intersection with a car stopped in front of us, a T section with obstacles in the corner reducing visibility, small intersection with a pedestrian crossing and a big intersection.

Like a first instance for this problem a condition-based program was created to determined which part of the RoI was feasible. In other words, it determines what part of the RoI was inside the range of the sensors. This programs give us a first stage to work with, although, the lack of scalability, bug appearance outside the scenarios which the program was focused to work on, and the alienation with reality made this work line not worth it anymore. As you can see in Figure 42 Big intersection. RoI feasible Y/N there are several fails. Not all the points are being taking into account when calculating the RoI feasible. The points that are being taken into account are not describing a realistic area that would be occluded by that object. These are some of the several errors that the code produced inside the simulation that it was designed to be tested on. The problems outside this close simulation grow bigger.

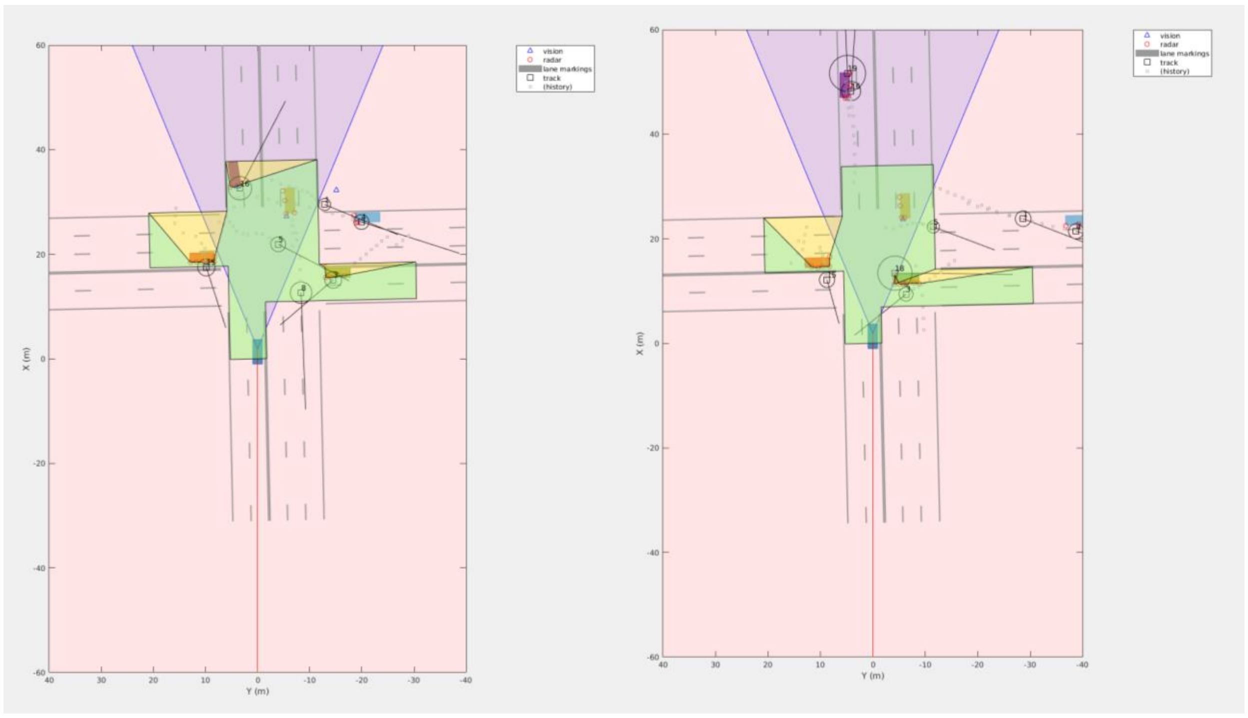


Figure 42 Big intersection. RoI feasible Y/N

Therefore, a new method was implemented. This new method, how it was implemented and the characteristics it adds to the program functions are discussed in the next section, Sensor Occlusion.

6.5 Sensor Occlusion

The previous method was not working for a real implementation, therefore, a new method had to be develop. In this case, as we want to see the occlusions happening inside our sensor range, and our sensor are reduced to just a 360 degrees radar and a camera, our initial range is a circle with the area that the LiDAR covers. When it comes to create this implementation with a real LiDAR obtaining data it would be as simple, a priori, as joining all the point cloud obtained from the sensor by layers. By doing so we will obtain a 3D map of occlusions that could be overlapped on the map so it would obtain a range of sensors capacity. As for the simulation, as we do not obtain a LiDAR information but radars data, there is a need to simulate additionally the occlusion from the sensors.

One first idea that may come to your mind is to apply the same reasoning that was carried out with the LIDAR data. To create an area that surrounds the vehicle in a circular shape, with the radius of the sensor range. Then connect all those points with the ones obtained from the sensors. But as the sensor data is not recorded by some order, yet all the information comes and is saved as soon as it is received, it is not possible to carry out this technique. Another way to perform this simulation of the occlusion from the sensors is the one that end up being implemented.

In this technique we make a circular shape and place all the points obtained from the sensors. Starting from the center of the circle, which is the same position where the ego vehicle is positioned, we obtain the direction of the vector to each one of the detections. Then, we scale that vector to point in that direction up till the end of the circle. In the ending point, the one laying on the circumference, we trace a perpendicular line to the direction from the center to the detection. In that perpendicular line we place two points separated by a distance δ. To determine δ we used a simple linear equation. The longer is the distance from the circumference to the detection, the bigger the value of δ becomes. Making a triangle with these two points separated δ with the detection, we obtain the blind side made by that detection. In the next figure, Figure 44 Object occlusion measurement, there is a representation of this effect.

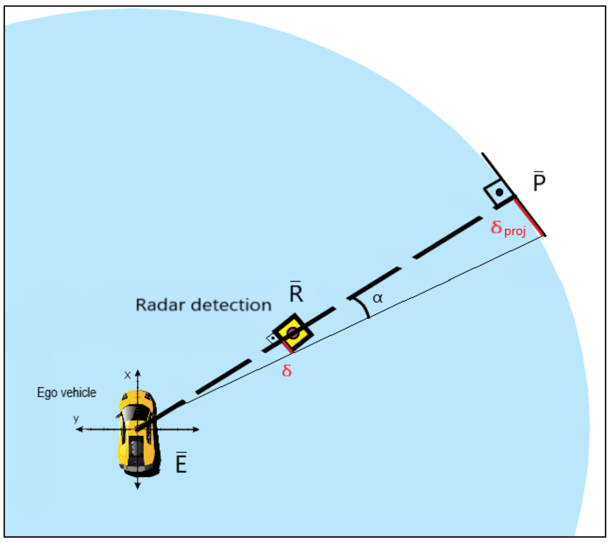


Figure 43 Sensor occlusion basis. Matlab

Within a recreation of the range of the sensor in a circular shape, we place the ego vehicle, $\overline{E}$, the radar detection, $\overline{R}$, and the projection of this detection at the end of the circumference, $\overline{P}$; and with a sensor resolution of 2δ we simulate a square shaped object in $\overline{R}$ with a side size of 2δ and facing $\overline{E}$. We calculate δ_{proj} for both directions.

$$\alpha = \tan\left(\frac{\delta}{\overrightarrow{ER}}\right)^{-1}$$

$$\delta_{proj} = \overrightarrow{EP}\tan(\alpha)$$

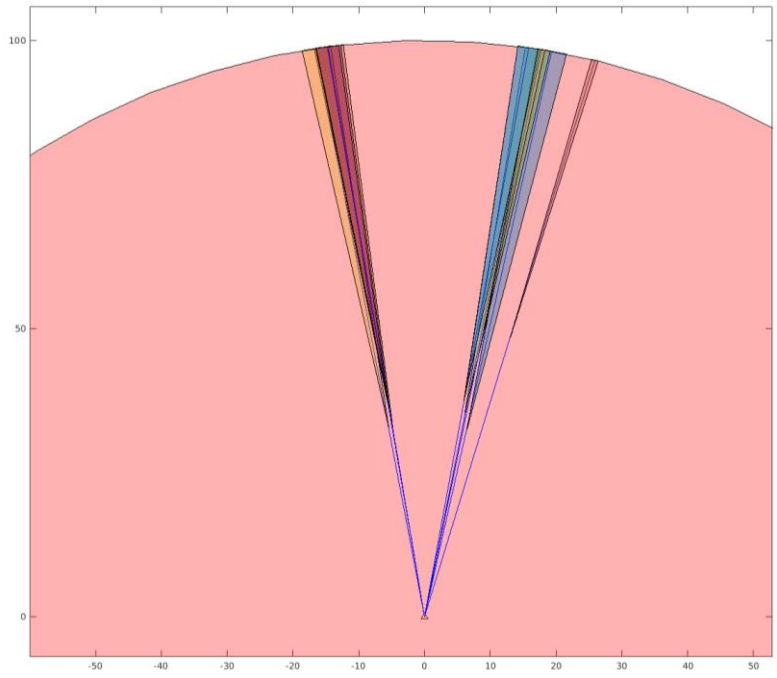


Figure 44 Object occlusion measurement. Matlab

In the previous figure a red area represent the range of the sensors (LiDAR and camera, both mixed) and the rest of the areas painted in various colors are the occlusion from the sensors. Merging all the areas and subtracting the occlusions from the range of the sensors we obtain the feasible range of the sensors, as it is shown in Figure 45 Sensor capacity after occlusion. This is the results from applying the previous idea into Matlab simulation.

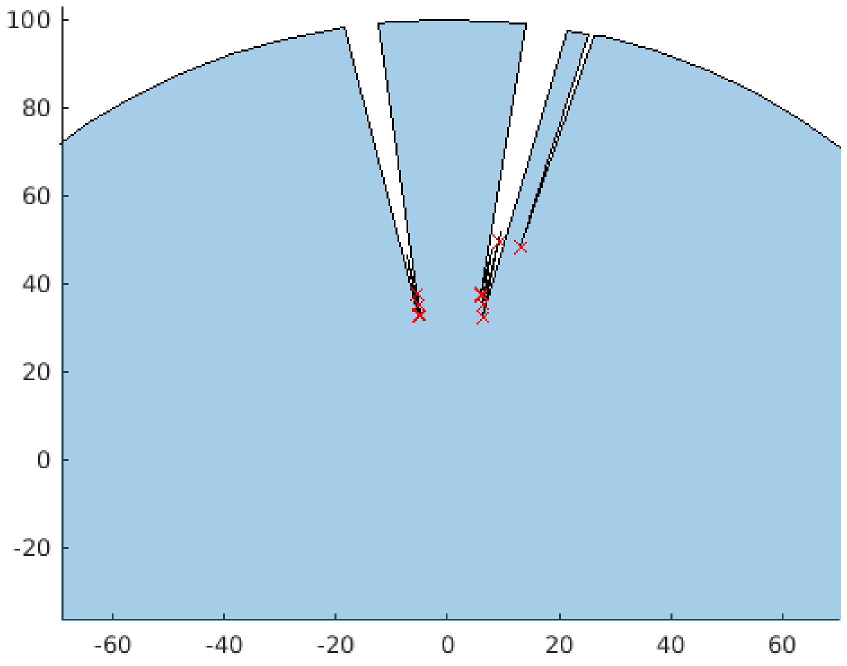


Figure 45 Sensor capacity after occlusion. Matlab

This figure represents how far the sensor are able to see, starting with the assumption that we have a circular ideal surface as the capacity of the sensor.

With an ideal sensor range $\sigma_{original}$ from where we subtract the occlusions, $\sigma_{occlusions}$ as the area between $\delta\ and\ \delta_{proj}$ we obtain σ_{real} shown in Figure 45 Sensor capacity after occlusion. Matlab. If we take into reference the first sensor placement we discussed in this research, Figure 31 ADST: Sensor placement, and we merge the areas of occlusion obtained with a circular shape with the area that all the sensors are covering, we can obtain a simple but reality likewise simulation of the occlusion for the objects. In the next figure, Figure 47 T section. Occlusion in sensors, a bird's eye plot representation with the occlusion of the sensor can be seen, checking how it works during the test.

When we have an area that covers all the space that the sensors are able to cover, and we have already obtained the Region of Interest that we need, we can check the percentage of that region that we are able to see. By merging those two areas into the intersection of both of them and comparing both the area of the intersection, Z_f, and the area of region we need to check, Z, we can value how sure we are about the environment that surrounds our ego vehicle.

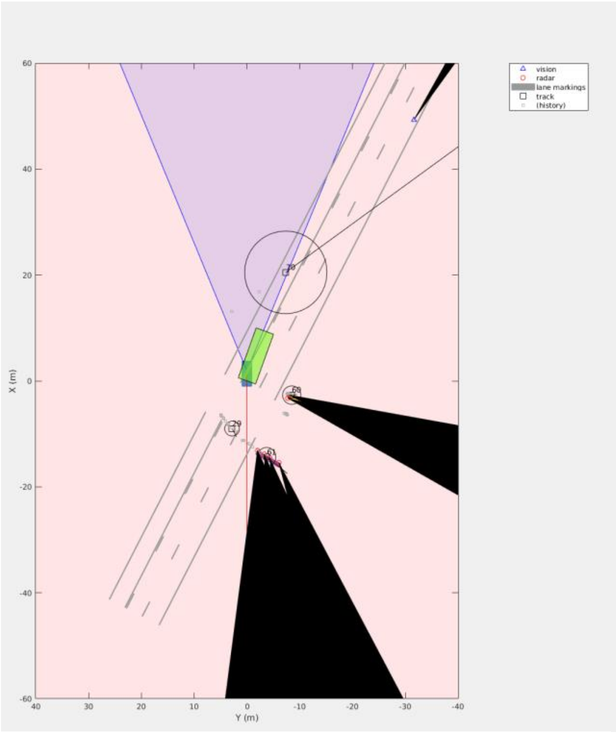


Figure 47 T section. Occlusion in sensors

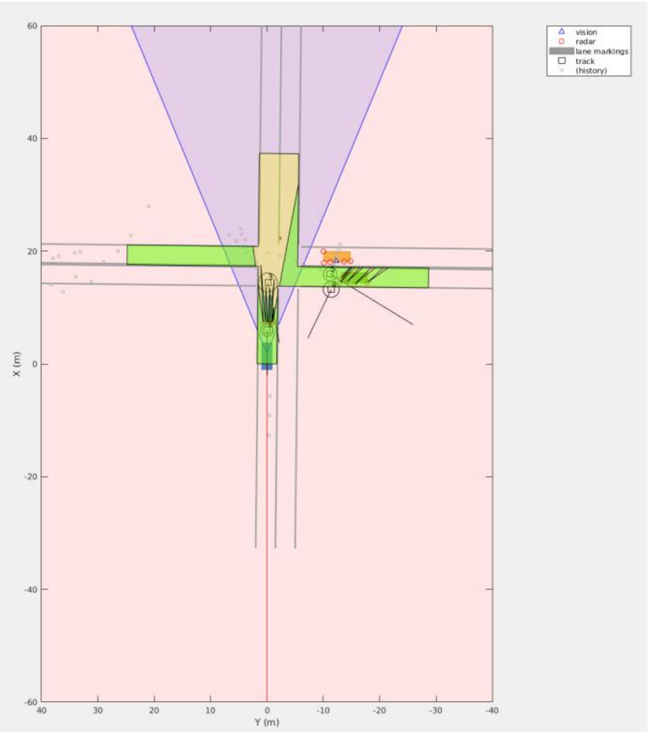


Figure 46 Small intersection. RoI feasible

Once we reached this point, the last step is to test this algorithm in the critical safety scenario. We designed a T section of a road, where a high speed road incorporation is need to be done. Besides that initial problem, we added obstacles in both sides blocking the sight for the road. As a human we would act in the following way, we would reduce speed as we reach the intersection, until we stop completely the car, and as we see there is not enough visibility, we would move forward in small steps until we find ourselves with enough visibility to decide to incorporate. We designed a code that would decide the same as the human should without declaring step by step how to proceed. We declared a vehicle that arrives to the intersection from the blind side right in the moment to crash with our ego vehicle if he moves reckless. The result of the experiments before adding the feasible region of interest can be seen in Figure 48 T section. Crash due to lack of visibility.

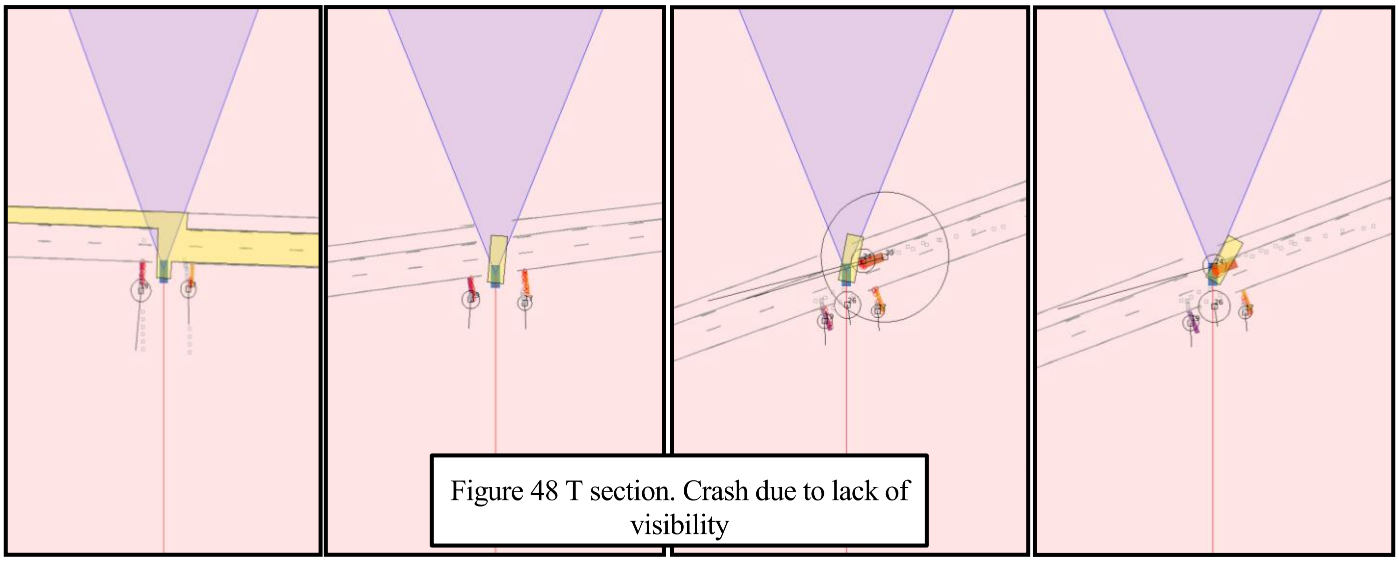
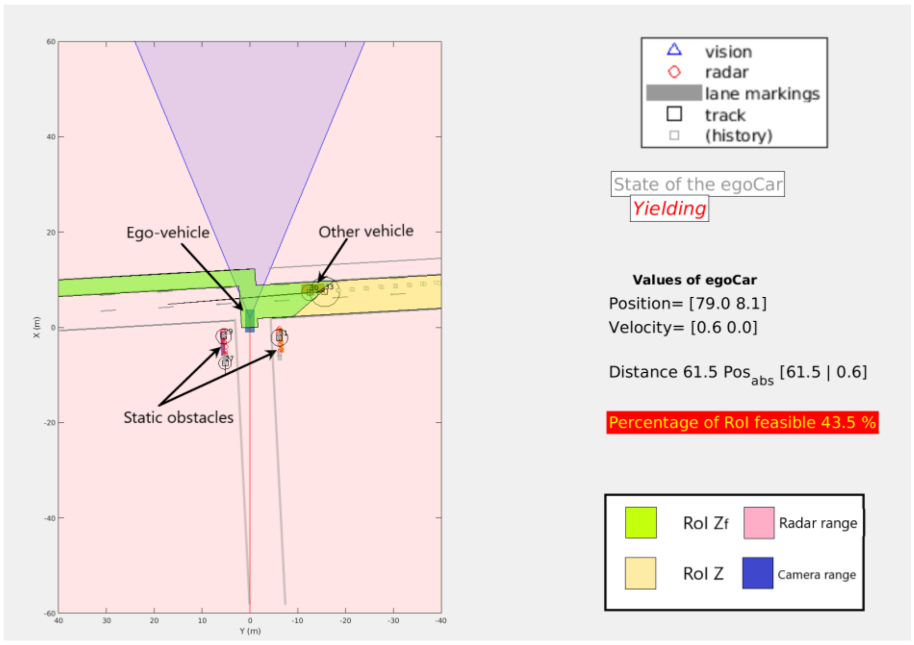


Figure 48 T section. Crash due to lack of visibility

After implementing the code for the feasible RoI, we need to establish a threshold. When the car travels with a value of their surroundings below that threshold will slow down its speed so it have more time to react for unforeseen situations. Also when decision making time such as intersections or overtaking in highways, when the value is under that threshold, smooth movements to acquire more vision would be taken until being sure up to that verge value to make the action. As it can be seen in Figure 49 T section road. RoI Feasible, the ego vehicle is moving slow, 0.6 m/s, as the percentage of RoI feasible is not enough due to the obstacles that are blocking the sight. We calculate the percentage of area visible, $RoI_{\%} = \frac{RoI_{feasible}^{current}}{RoI_{ideal}^{current}} = \frac{Z_f}{Z}$, and based on that value perform the proper safe reaction for the ego vehicle. We implemented this idea in the critical safety scenario with a simple FSM for decision making, adding a behavior to improve the $RoI_{\%}$ value if there is not enough visibility on the turn. Choosing the correct value for the threshold we manage to make our ego car to wait for the car in the blind spot to cross before taking the intersection, or to go first if there is time enough to merge into the new lane.



State of the egoCar
Yielding

Values of egoCar
Position= [79.0 8.1]
Velocity= [0.6 0.0]

Distance 61.5 Pos_{abs} [61.5 | 0.6]

Percentage of RoI feasible 43.5 %

Figure 49 T section road. RoI Feasible

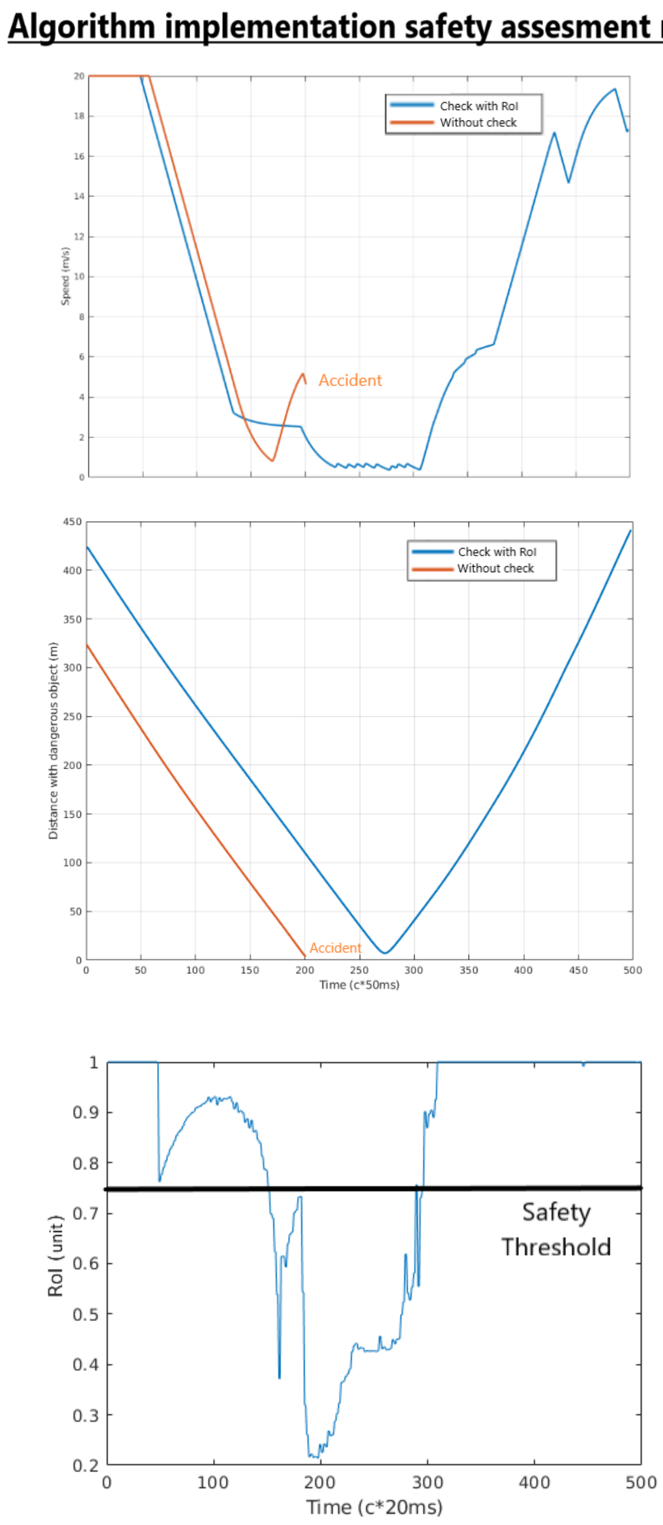


Figure 50 Speed and distance check with and without our proposed safety layer implemented. We can see how the accident is prevented by improving situational awareness

When using our proposed idea we not only avoided the crash, yet performed a smoother action around an intersection with low visibility, as we show in Figure 50 and Figure 51. In the following picture we deepened in details explaining the car state in each moment during the simulation. We show that the dangerous situation for the car is avoided by slowing down when visibility is not enough. We also limited the actions on the decision making module based on the value $RoI_{\%}$, giving a safer and more consistent behavior.

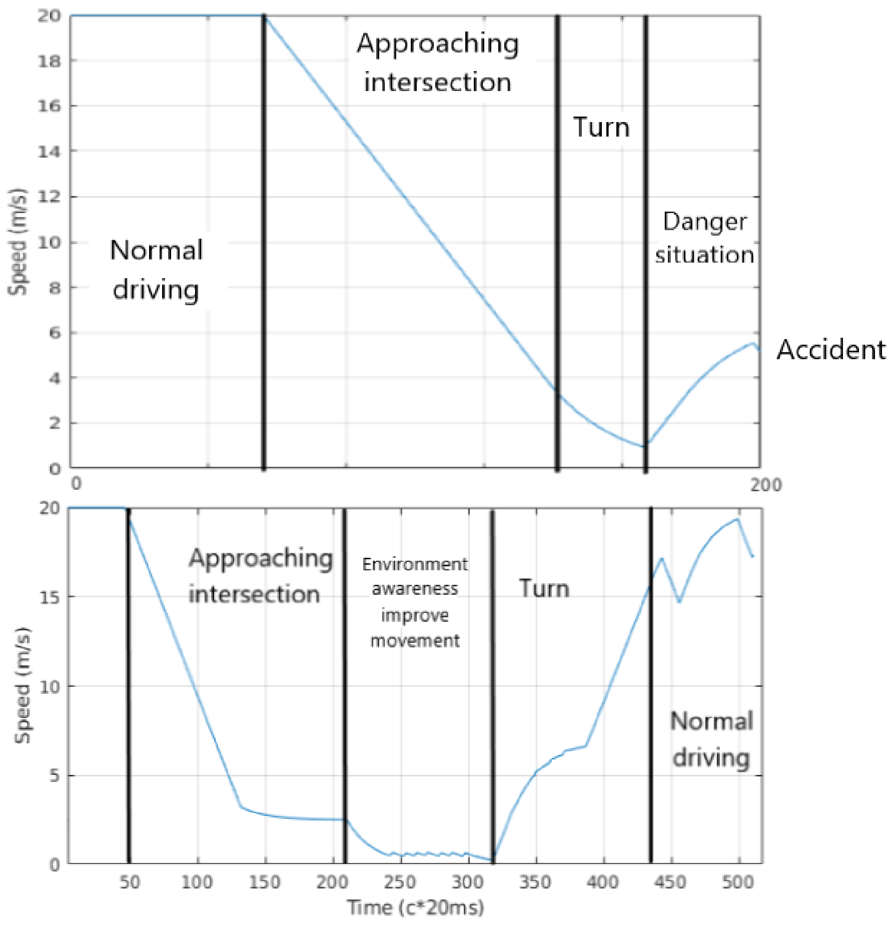


Figure 51 Comparison between FSM with and without our proposed idea, in detail. On the first picture there is the result from the Figure 8 with the accident, on the second picture the speed using our safety assessment management.

7 AUTOWARE IMPLEMENTATION

Autoware is an open source package for ROS for autonomous vehicles, which includes several packages for the processing of data, motion and mission planning for the movement and several algorithms for obstacle avoidance. ROS can be worked either in Python or C/C++ languages. As a basic of ROS is needed to understand the whole program, a simple explanation of ROS performance is giving ahead.

There are several terms required to understand in depth how ROS works, besides the relation among them:

- Node: are the backbone of ROS architecture. In ROS there is no one big program that handles all the functions on it, but there are several nodes that connects through topics to perform defined task each one. There is one main core node that is going to be the handler of all the other nodes address and topics links. Each node has to have a unique name so the core node is able to identify it from the list. Each node can be program in different languages and the it has to be built so the program know all the dependencies needed in order to create an executable that will work as one or more node. One of the main differences between the two languages available in ROS is that when programming in Python the executable just point to the .py file so all the changes made in the code does not need to be rebuilt in the program. Just the opposite than with C/C++ where each change has to be applied and rebuilt to create a new executable that will launch a node. All the code in the nodes can be launch more than one time as long as the nodes have different ID.
- Packages: is the connection of one or more nodes with other packages and among them. All and each one of the nodes needs to belong to one package. By building all the packages in your projects is the only method to launch the different nodes in a successful way.
- Topic: reserved space for nodes to publish messages and receive the data. Instead of sending the information to one of each of the nodes in the system, the message are published in the topics where other nodes can request to obtain that information. Each topic has to define a message type for the topic, so all the nodes can know how the information is being uploaded. Same as with nodes, each topic has to have a distinct name.
- Message: are the information published on each topic. One type of message can be published in various topics, yet, each topic must always publish the information in the same type of messages. Inside the own ROS libraries there are several messages types, which are the most common.
- Publisher/subscriber: as it was mentioned in the previous terms, each topic has messages published on it. When a node wants to upload information to a topic need to sign up as a publisher in the core node. When a node wants to receive the information from a topic needs to sign up as a subscriber in the core node. Each node can subscribe and publish as many topic as it needs.

In the next picture Figure 52 RQT from rerunning experiment, there is a simple representation (carried out by one of the different nodes in ROS libraries called RQT-Graphs). The rounded text box are representing nodes and squared ones represent the topics of the program actually running. Not all the nodes has to be link among others, but may be isolated zones for specific functions, but as long they all connected to the core node, even when the node are not publishing or subscribing, they are going to be placed with this functionality.

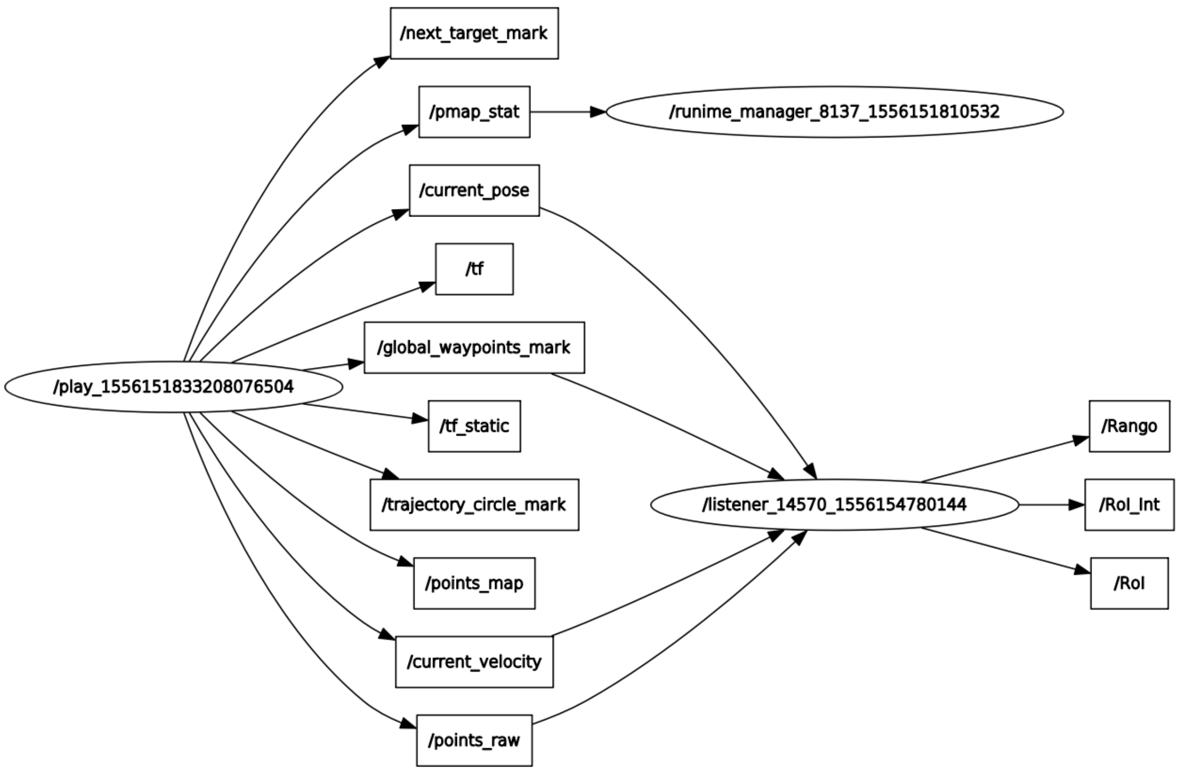


Figure 52 RQT from rerunning experiment

Autoware works on top of this structure and it can be used for different purposes. The tree main purposes that we will be using are to rerun simulation from a ROSbag[5] with data from a previous experiment recorded, simulate the behavior of synthesized experience based on a real map and, lastly, to run a whole autonomous system on a real car. The complexity of the process increases as we move toward the final destination of real life implementation. In Figure 52 RQT from rerunning experiment there is a representation of all nodes while rerunning an experiment, adding some features, later discussed. In Annex A there is a representation of nodes when running a simulation in Autoware environment. The step between this and real-life implementation would be including all the read from the sensors in real time, and publishing this information on the different topics.

As a method of visualization for the experiments there is one module of ROS that come in handy, RViz. With RViz there a various type of figures, lines and models that can be represented in a 3D space, with different represent axis. As it is displayed in Figure 53 Autoware graphic representation of experiment. Moriyama ROSBAG., we can see all the rings (layers) from the LiDAR, a model for the car, and object distinction in the image, by varying the color of the points from the LiDAR. This image belongs to one of the maps distributed by the same company from Autoware, and it is created from a place in Japan, Moriyama. From Moriyama ROSbag we do not only get the map, but also a crucial element inside Autoware conception, a vector map. In order to perform the task of the autonomous driving faster, Autoware relays on a vector map loaded offline with the key points from the map, such as lane centers, lane width, light traffic position, stop signals among others.

[5] A ROSbag is a type of storage data for ROS environment. All the data from a certain experiment or for a database can be saved in it, and the access it publishing its data in the attached topics.

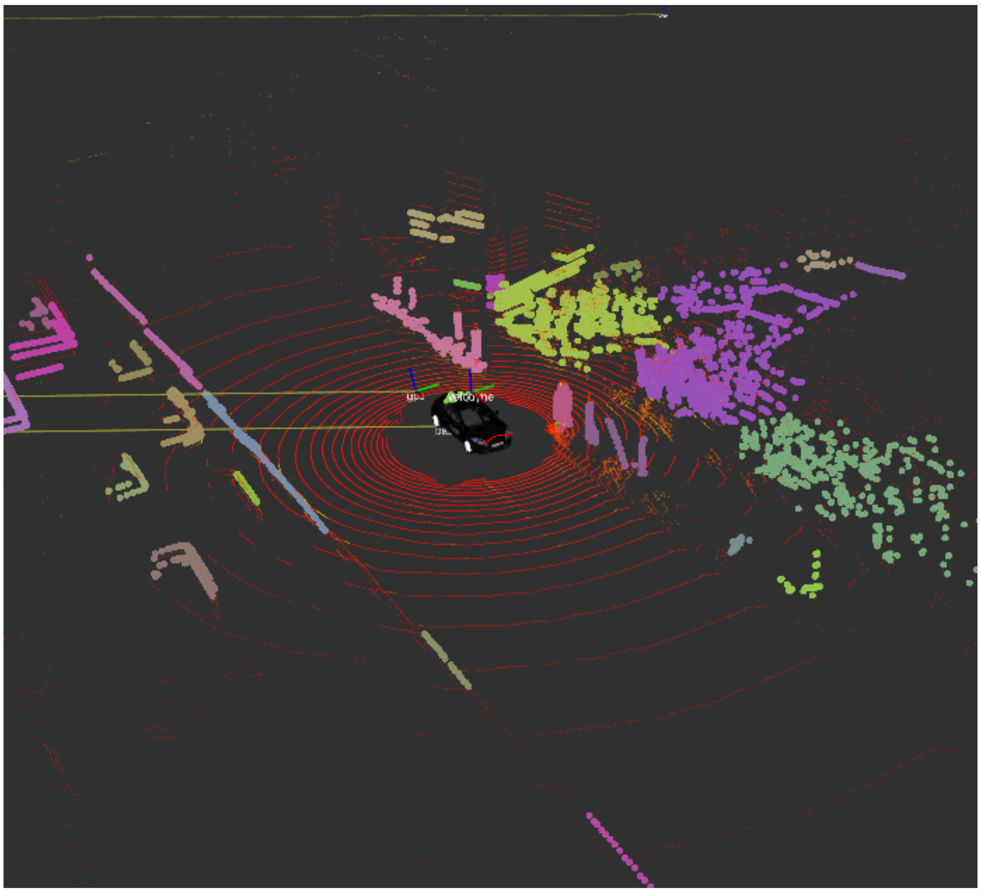


Figure 53 Autoware graphic representation of experiment. Moriyama ROSBAG.

Other thing needed for localization and orientation, besides object detection, is the point cloud obtained from the LiDAR, with a matching between the point cloud from the map and the ones obtained in that moment, the position of the car is determined. Apart from the one in Moriyama, Hitachi Ltd. owns another ROSbag with information from inside Hitachi Research Lab in Oomika. Inside this labs there is one path already determined where experiments can be performed. Upon the data from this experiment, we will include the safety verification we designed in the section Region of Interest to check how sure we can be about our surroundings and act in consequence to it. As Autoware already have decision making modules, obstacle avoiding, local planning …etc., there is no need to create any of this for this part of the project. To make a change in decision making you need to change a big part of the Autoware program, which may causes many bugs. For that, safety approach will be affecting speed of the car on each moment based on the percentage of feasible Region of Interest.

7.1 Region of interest

Same as we did in Matlab, we need to design a space which we define as needed what would be called Region of interest. The map from Hitachi Ltd. does not have a vector map associated to it, so the lane width and other characteristic are not available. To overcome the lack of those data is needed to create a RoI for the map. Creating the RoI as the car is moving may require computational load that slows down the car. To avoid it, using the global waypoints from the map, an entire sheet of data with the limits of the RoI was made, and the used during the simulations. By doing all the calculations need for the RoI offline, we improved the performance and speed of the code.

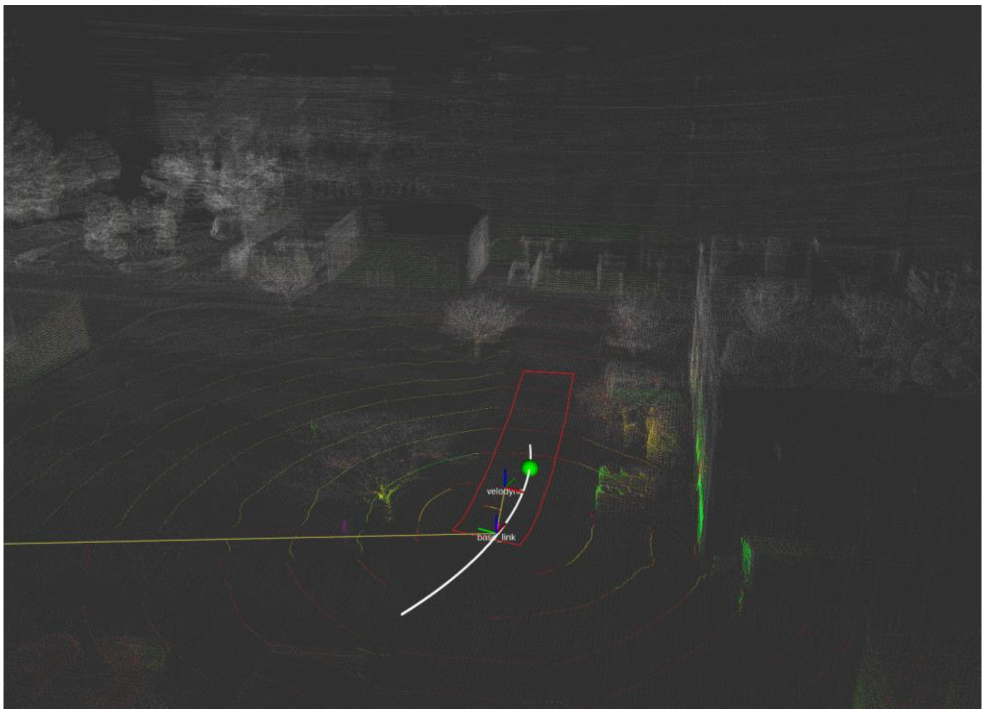


Figure 54 RoI Waypoints. Autoware

The resource that we had available was the waypoints of the trajectory of the vehicle during the experiment. Those waypoints were referred to the world frame that does not move with the vehicle. Knowing where all the waypoints are, we can draw a line that connects two waypoints coordinates and create the perpendicular line to it. Moving on that perpendicular line, leaving the space of a standard lane width we can create a file with all the points of right and left border for the region of interest. By creating a file to access during the experiment we take out the computational load from calculating the perpendicular on each point of the trajectory. This region of interest is only for the straight line (even though sometimes is not straight but goes with the curves) and does not take into account all the other topologies important from the map, like intersection, incorporations, among others. With the vector map we could take those cases into account in a generalized way, but as the tool to create vector map from Autoware web pages is having some bugs, there is no possibility to work with that[6]. In our map there is only the presence of 4 intersections, but one of them does not have problem of visibility so it was subtract. In the Figure 54 RoI Waypoints. Autoware we can see with a red line bordering the region of interest of this straight line. In the figure there are also one green dot that represent the pure pursuit destination of the vehicle to move, and the white curve represents the curvature of the car on that instant. As there is no model for the car in the simulation, the position of the vehicle is only represented by both frames of the base link and the LiDAR Velodyne. The distance of each waypoint, as specified by Autoware, is 1 meter. Based on this information, we can look ahead in the trajectory doing some calculations with the current speed. As both the waypoints and RoI borders share the same index, what it was done in the program was to check the position of the car in one instant and attach a waypoint as the one the car is place in that moment. From that point, based on the speed of the car, the number of the following index is determined and all the points from the border are saved in one variable. One thing important to pay attention to it is the fact of how the information is being saved in Autoware. What now we are implementing is a graphical method, although there is no need for representation of it in any case. That said, all the polygons showed in the figures are saved with their borders in a variable type defined in ROS as a polygon.

For the implementation of intersection in this case there were a need to take some decisions offline and then apply them to the program, which makes the program only work in the current specifications. As it was mentioned already in the document, this can be easily modified to work with vector maps that contain the information. Checking in that information the position of any road that may interfere with our current

[6] As of the date when this document was written, May 8th 2019.

trajectory can be added to the RoI without larger modifications in the code. What it was implemented on this case, which can be also used for vector maps, is to check if any of the points inside of the RoI straight have a link to others roads. In this scenario, the links were added manually in the three intersections, alongside with the RoI needed for the intersections, in the points were the car is turning the most on the curves, thus any other point of the curve could have been taken. Same as it happened with the straight RoI, in this case also a polygon type variable was used. As it was manually input, the polygon only requires four corners to work simplifying the number of data for the polygon. One of the problems that can be seen in Figure 55 RoI Intersection. Autoware is the overlapping between both representations. As the representation it there is no problem to check the RoI, but, when counting the percentage of RoI that is feasible, that duplicity among areas could bring some inconsistencies during the performance. There are several ways to try to avoid this kind of problems. One would be to determine the percentage of area that is overlapping and subtract that area from one of the areas. Other would be to determine the RoI of intersections without the area of the RoI straight, yet, this would require more specification and we are working on the opposite direction for this project. One last option, and what finally was implemented was to create a polygon that was the union of both other polygons. This actions produces certain delay when processing data but creates and accurate polygon with desired boundaries. Even when this programing was made, there is no possibility to represent it on the same program that plots the whole point cloud as it crashes without any deductible reason.

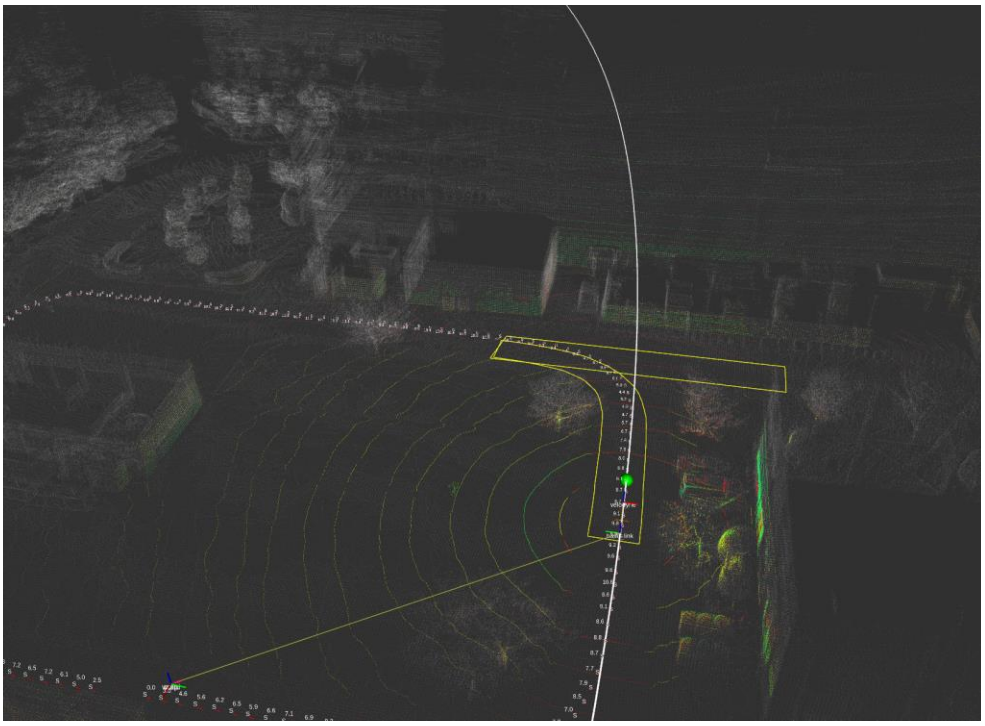


Figure 55 RoI Intersection. Autoware

For that reason, both polygons are represented in the previous figure, Figure 55 RoI Intersection. Autoware instead of only one polygon with the union of both areas. In that figure, same as the previous one, there is no model of the car, so its position is plotted by the frames that moves alongside with it. The change of color of the RoI does not have further explanation that visual facility for the user to check the areas. Once the RoI of this experiment is created and tested in the whole scenario is necessary to determine the range of the radar and sensors from the car. In the current experiment, the only data available is the one from the LiDAR. Here we will retake and continue the idea mentioned in Sensor Occlusion of taking all the points of one layer LiDAR creating a polygon that will represent an approximation of the range. In one layer LiDAR would be as easy as taking all the data and order them by angles regarding Velodyne frame. In our case, LiDAR have either 32 or 64 layers, and the points are stored as soon as they arrive from the reflection. Therefore, there is no order and the height may vary in each layer. What we did to filter the points is to take all the points restricted between two heights and split all the points in sectors inside a circumference.

7.2 Polygon range

The idea described here to solve the problem of the range of vision for the LiDAR is as following. First you need to split all the points raw from the point cloud obtained from the LiDAR to only focus on the ones truly important for occlusions Ideally we could use all the points from the point cloud and create some kind of selector that indicates which one are creating an occlusion, but this would consume time enough to make impossible to work on real time. Instead we cropped all the points based on their relative position with the car. If $0 \leq x, y \leq \max\left(d_{brake,max}^{r_{main}}, d_{cross}^{r_{int}^{n}}\right)$ & 0.5m ≤ z ≤ 1.5m the points were included in $points_{raw}^{imp}$. For the height crop we chose 1 meter range around the height of 1 meter as most of the other actors has their gravity center around those heights. Once we have all the important points, we decided to split all our coverage area in different sectors. Depending on the number of sector we decide to create, we will obtain different resolutions. Therefore, resolution can be expressed as a relation between the whole range and the number of sectors.$resolution(°) = \frac{360°}{number\ of\ sectors}$. The number of sectors will highly increase the computational time needed for the processing and also making unavailable to work on real time. Working with those two parameters, resolution and processing time, we had to choose a tradeoff that gave us enough resolution for our functionality while still working on RTO. By splitting all the area in sector we obtained two major improvements. The first one, we shared the computational load among all the processing units present in the AV processing core and GPU, if any. Alongside with this, gave us a method to control the resolution of the LiDAR, and in a certain way, to crop the number of samples that we obtain from it One of the problems after having a circumference with all those points is to order them. The number of sectors will highly increase the computational time needed for the processing and also making unavailable to work on real time. Working with those two parameters we have to choose a tradeoff that give us enough resolution for our functionality while still working on RTO. Deciding a number of sectors for our original range as it is shown in Figure 56 Polygon range. Figure 1 we need to represent all the data from the LiDAR.

By splitting all the area in sector we obtained two major improvements. The first one, we shared the computational load among all the processing units present in the AV processing core and GPU, if any. Alongside with this, gave us a method to control the resolution of the LiDAR, and in a certain way, to crop the number of samples that we obtain from it.

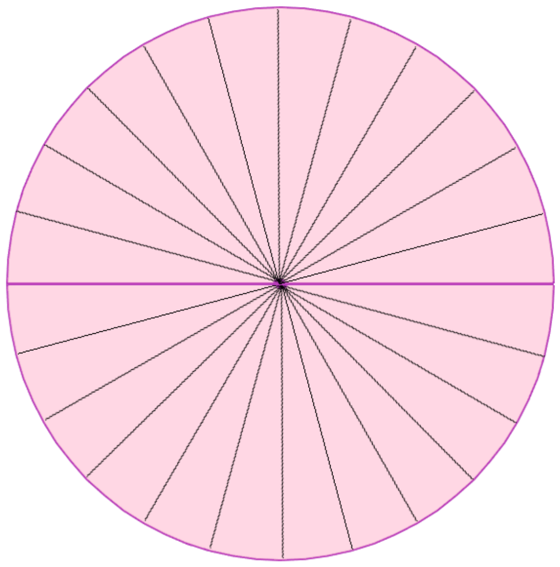


Figure 56 Polygon range. Figure 1

Another problem to discuss is where and how to place all the points from the point cloud. Right now we are working with a 2D area of coverage for the sensors, so place all the points in a 3D space would be using resources in an inefficient way. To avoid this waste of resource, all the points were projected to the ground level were the processing will take place,$p(x, y, z) = p(x, y, 0)$, meaning by 0 the ground level. With all the points in a 3D space, there could be some processing left to do to improve the accuracy, such as grouping points per object and creating 3D bounding boxes. As this would take some time of processing, all these improvements were discarded, using a worst case scenario.

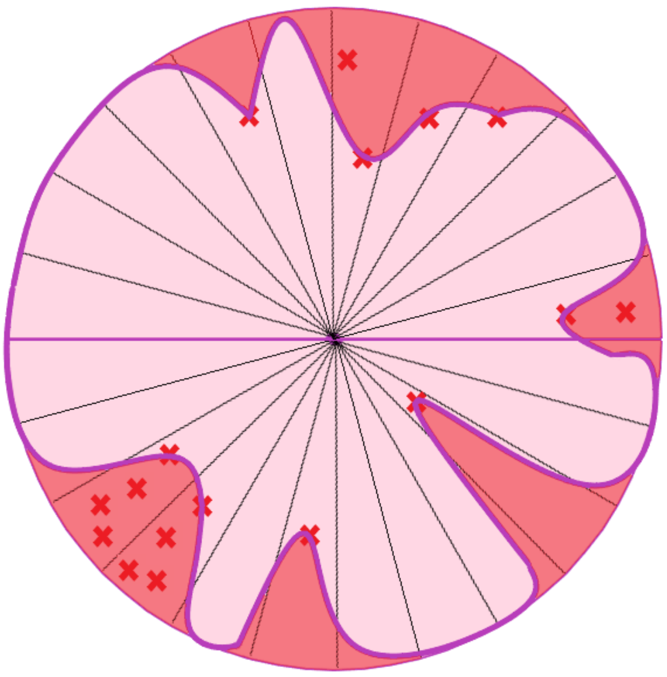
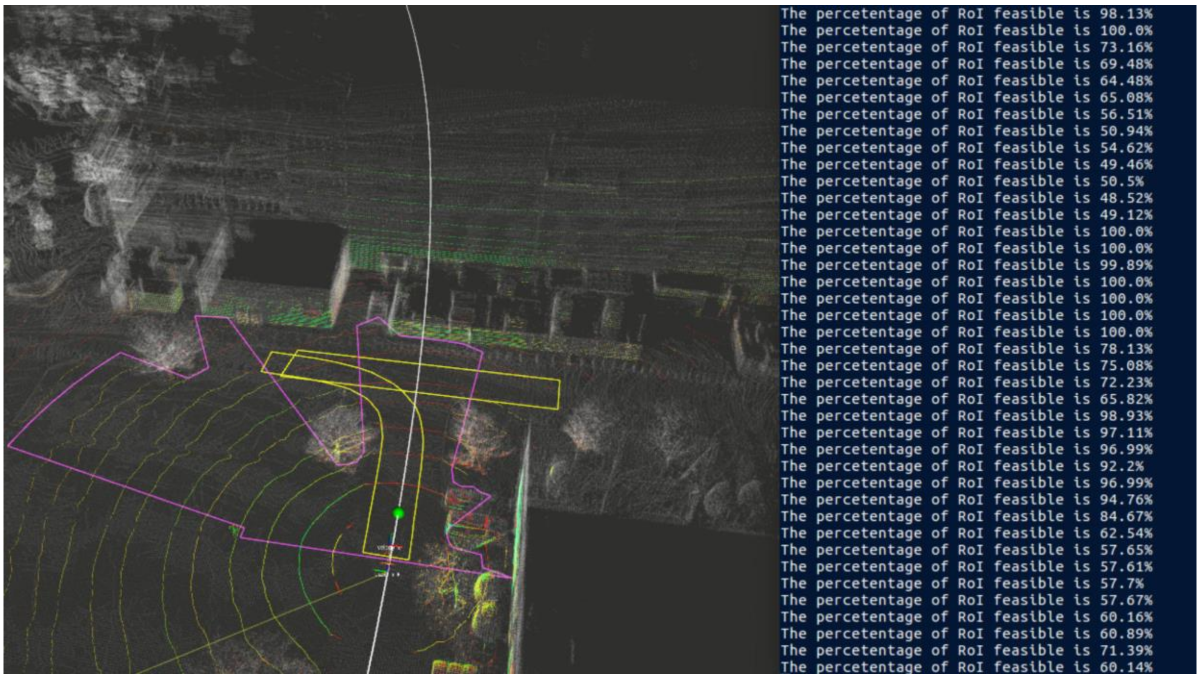


Figure 57 Polygon range. Figure 2

In Figure 57 Polygon range. Figure 2 we have the way we chose to represent the occlusion from objects. From the point cloud information we will receive numerous quantity of points were objects were found, but processing all these objects will take time. To avoid that we just projected all the points in a ground level layer, and took the worst case scenario, which was the closest point to the reference system, the Velodyne frame. Obviating all other points, either the closest point to the frame, if any, or the maximum range point, if none, were chosen and jointed together by a polygon. This would not represent a truly realistic occlusion range, but it will most likely give us an area similar to the real one. Using that area a feasible RoI can be calculated. After taking all the points in the circumference, and for further processing after this step, there is a necessity of ordering all the points based on their thetas relative to the frame. Same as we did with both the other RoI, this information of a Polygon will be published in one topic for a later node to process all this info and create a feasible RoI. To improve its performance it is possible to select some sectors depending on whether there is RoI inside of it or not, yet this modification have been not tested.

7.3 Results

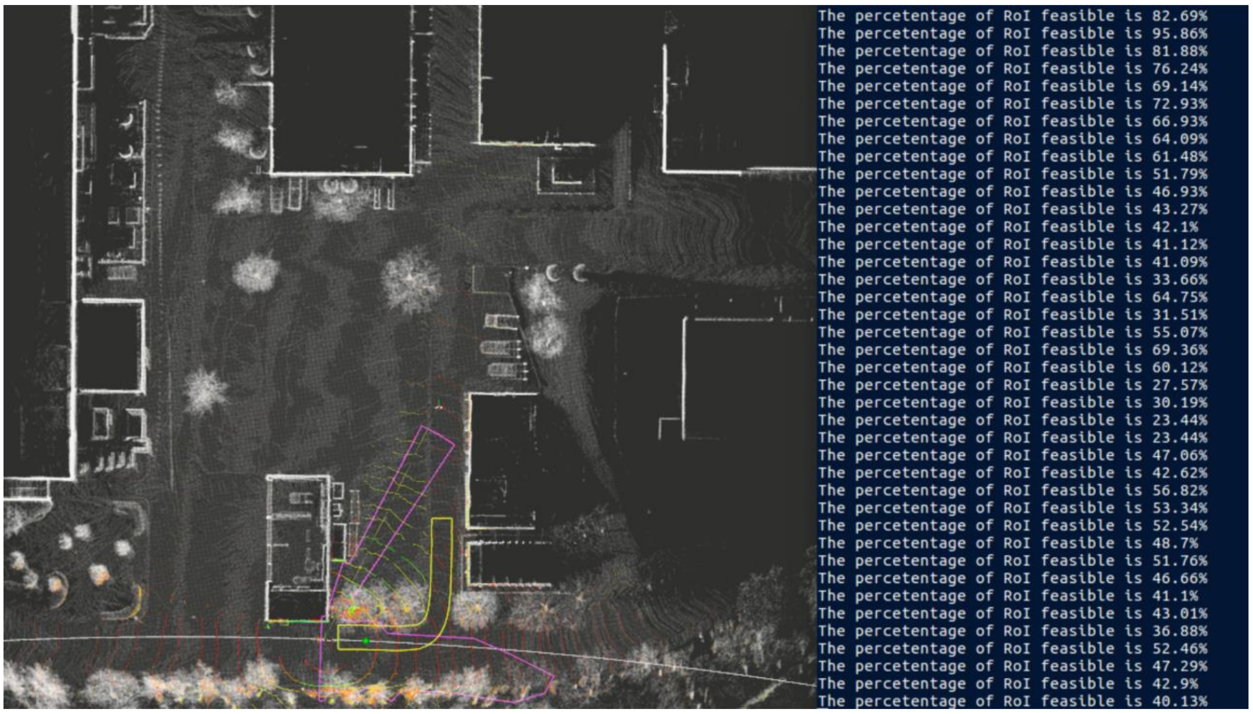
Putting all the pieces up until here together we can proceed with the final test over the ROSbag we were working on. We already created both functions to determine the RoI either for the path the car is going to follow and the different intersections inside our map. And we delimited the range of coverage of our main sensor (and only one available on these experiments). Processing the area of intersection between both areas, we could calculate, same as we did in Matlab, the feasible RoI and how much percentage of the actual RoI is covering that feasible RoI. When trying to plot that kind of polygon with different parts (as you can remember from previous sections, the feasible RoI can be split in different polygon with no connections among them) the program that represent the whole experiment crashes. Therefore, there is no possibility of printing the feasible RoI. Yet, we can still calculate the percentage of it, which we will later use for decision making purposes. In Figure 58 RoI Feasible. Third Person View Plot and Figure 59 RoI Feasible. Birds Eye Plot we took a screenshot of the video that contains the whole experiment, from two different angles.



```
The percetentage of RoI feasible is 98.13%
The percetentage of RoI feasible is 100.0%
The percetentage of RoI feasible is 73.16%
The percetentage of RoI feasible is 69.48%
The percetentage of RoI feasible is 64.48%
The percetentage of RoI feasible is 65.08%
The percetentage of RoI feasible is 56.51%
The percetentage of RoI feasible is 50.94%
The percetentage of RoI feasible is 54.62%
The percetentage of RoI feasible is 49.46%
The percetentage of RoI feasible is 50.5%
The percetentage of RoI feasible is 48.52%
The percetentage of RoI feasible is 49.12%
The percetentage of RoI feasible is 100.0%
The percetentage of RoI feasible is 100.0%
The percetentage of RoI feasible is 99.89%
The percetentage of RoI feasible is 100.0%
The percetentage of RoI feasible is 100.0%
The percetentage of RoI feasible is 100.0%
The percetentage of RoI feasible is 100.0%
The percetentage of RoI feasible is 78.13%
The percetentage of RoI feasible is 75.08%
The percetentage of RoI feasible is 72.23%
The percetentage of RoI feasible is 65.82%
The percetentage of RoI feasible is 98.93%
The percetentage of RoI feasible is 97.11%
The percetentage of RoI feasible is 96.99%
The percetentage of RoI feasible is 92.2%
The percetentage of RoI feasible is 96.99%
The percetentage of RoI feasible is 94.76%
The percetentage of RoI feasible is 84.67%
The percetentage of RoI feasible is 62.54%
The percetentage of RoI feasible is 57.65%
The percetentage of RoI feasible is 57.61%
The percetentage of RoI feasible is 57.7%
The percetentage of RoI feasible is 57.67%
The percetentage of RoI feasible is 60.16%
The percetentage of RoI feasible is 60.89%
The percetentage of RoI feasible is 71.39%
The percetentage of RoI feasible is 60.14%
```

Figure 58 RoI Feasible. Third Person View Plot

As we can see in both figures, we have the same color for the RoI than in the previous figures and we have a new area, in pink color, that represents the coverage area of the sensor. The intersection, in percentage scale, of both areas is being printed by other node every time either the RoI or the coverage changes its values. As it is implemented right now, both should be changing at the same time, but in case there is an anomaly or a modification in the code, if one of those two changes and the value is not updated could mislead to some problems in the security check of the action. The change of view allow us to observe and check the whole behavior of the program. As the feasible RoI might not be a variable we need to update that often, depending on the purpose we want to give it, we could change the update statement to a different one, doing so periodically or by request, if desired.



```
The percetentage of RoI feasible is 82.69%
The percetentage of RoI feasible is 95.86%
The percetentage of RoI feasible is 81.88%
The percetentage of RoI feasible is 76.24%
The percetentage of RoI feasible is 69.14%
The percetentage of RoI feasible is 72.93%
The percetentage of RoI feasible is 66.93%
The percetentage of RoI feasible is 64.09%
The percetentage of RoI feasible is 61.48%
The percetentage of RoI feasible is 51.79%
The percetentage of RoI feasible is 46.93%
The percetentage of RoI feasible is 43.27%
The percetentage of RoI feasible is 42.1%
The percetentage of RoI feasible is 41.12%
The percetentage of RoI feasible is 41.09%
The percetentage of RoI feasible is 33.66%
The percetentage of RoI feasible is 64.75%
The percetentage of RoI feasible is 31.51%
The percetentage of RoI feasible is 55.07%
The percetentage of RoI feasible is 69.36%
The percetentage of RoI feasible is 60.12%
The percetentage of RoI feasible is 27.57%
The percetentage of RoI feasible is 30.19%
The percetentage of RoI feasible is 23.44%
The percetentage of RoI feasible is 23.44%
The percetentage of RoI feasible is 47.06%
The percetentage of RoI feasible is 42.62%
The percetentage of RoI feasible is 56.82%
The percetentage of RoI feasible is 53.34%
The percetentage of RoI feasible is 52.54%
The percetentage of RoI feasible is 48.7%
The percetentage of RoI feasible is 51.76%
The percetentage of RoI feasible is 46.66%
The percetentage of RoI feasible is 41.1%
The percetentage of RoI feasible is 43.01%
The percetentage of RoI feasible is 36.88%
The percetentage of RoI feasible is 52.46%
The percetentage of RoI feasible is 47.29%
The percetentage of RoI feasible is 42.9%
The percetentage of RoI feasible is 40.13%
```

Figure 59 RoI Feasible. Birds Eye Plot

With all this information obtained, and all the nodes already working we need to increase the robustness of the program. When starting the program we need to have a series of Waypoints in a folder, we need to have access to the RoI boundaries files and we need to have a certain amount of data to work. As for simulations if there were fails obtaining one of those data, we just needed to rerun the node and the problem would disappear, but when wanting to implement the code in real life there is no room for those mistakes. For that reason, before starting there is a check of whether all the information were collected or not, and if not, the node would stop its process and not publish any data until it receives all the required data. Other problem may appear from synchronization problems between the polygons, as in some times, the program would try to check the intersection between two polygons that were not completely defined. In those cases just a warning appeared and the program kept running, but to avoid further due with these errors, there were check for both validation of polygon shape and intersection of areas. If one of those were incorrect or the area of intersection were founded as zero percent, there would be no new publication, as it is consider to be a problem of processing, not the presence of uneven cases. By letting the topic unpublished we will keep the last value it was published for other node to read, that could causes some troubles. To avoid reading an outdated information about the feasible RoI, there is a header with time stamps for both RoI and Feasible RoI to check if there is so much delay from real time.

As the feasible RoI might not be a variable we need to update that often. Yet, we could change the update statement to a periodically or by request in case the module failed on the work. If the safety check is not working a warning is sent to warn the user that the car is not able to check its surroundings.

With the feasible RoI we obtained a quantitative value for the safety validation check, and defined an action contract that improved the safety state of the vehicle. By reducing the speed of the vehicle on those situations we become able to react properly to unforeseen events that may occur.

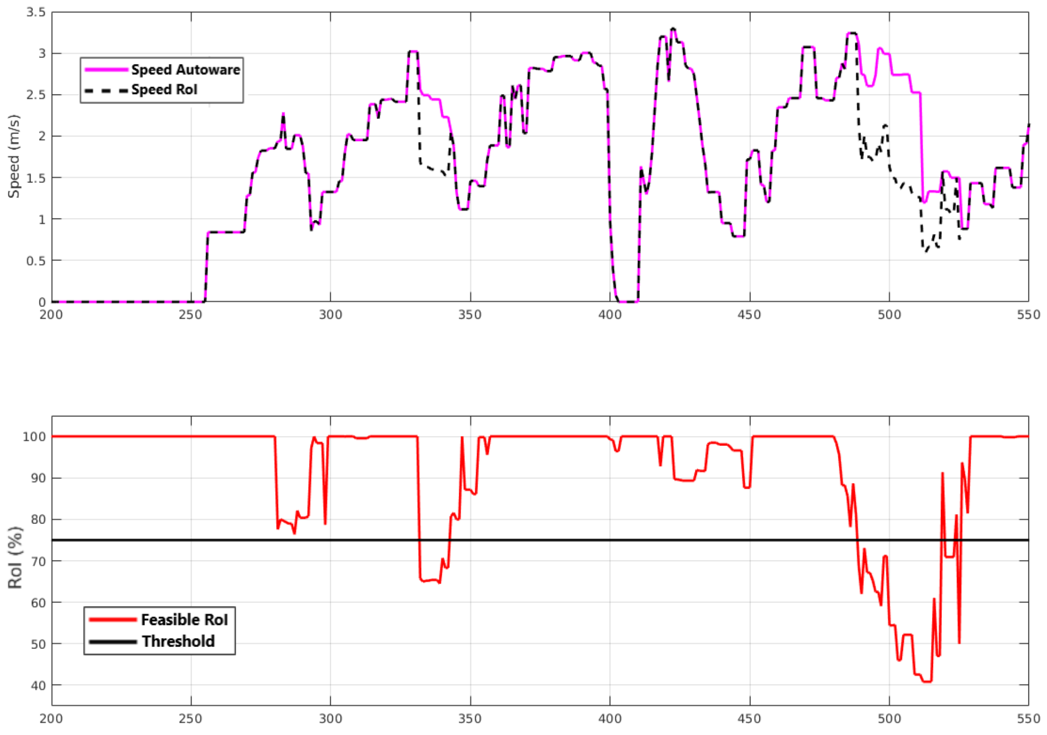


Figure 60 Speed comparison based on feasible RoI values

8 CONCLUSIONS AND FUTURE WORK

We developed a novel system that guarantee safety for AV in scenarios where dynamics such as weather conditions or limitations in the sensors can be overcame by measuring the percentage of surroundings can be checked by all the sensors. We proved this idea in several simulations and toolbox (Matlab and Autoware) with highly positive results. We also took the next step and implemented this idea in an autonomous vehicle and tested its results in a controlled scenario with also positive response. This novel idea can be a required step for a level 4 automation in AV. This idea also works as an upper level addition for the machine behavior, allowing it to be implemented in any kind of decision making plan with just a few arrangements.

As for future work relating this idea, there are several ways to increase its effectiveness. Firstly it would be mandatory to test this idea in a larger amount of samples, different scenarios to assure the effectiveness of this idea no matter the scenario. This would be a mandatory step in the development of this idea.

As for any additional content that can be included in this idea there a different approaches:

- A standardization of the region of interest (RoI): To standardize all the region of interest for the whole possibilities in the road will allow this idea to be implemented in all the current state-of-the-art AV behavior. As of the moment multiple lanes scenarios and intersections scenarios are defined, but other unique elements, such round wheels are not yet defined. There is a necessity of adding this definition for this scenario to the idea in order to work in countries that includes this elements in their road infrastructure.
- Map reading by the machine: Being able to adapt for all the scenarios includes a necessity for this idea to be able to read maps of the surroundings in order to create an accurate region of interest of the surroundings. This could be done by including a map of the whole country where the system might be tested or, rather, to download the maps from a cloud and being able to turn those into information the machine can check for occlusions.
- Other sensor´s occlusion: The project showed the occlusion in sensors such as LiDAR and radar, which works on most the AV vehicles, yet, those are not the only one that can be used for measuring the environment. If we take, for example, Tesla´s car only uses camera for a safe autonomous driving. Therefore, to include a program that can check the occlusion of the camera based on the variables read by itself, will increase the acknowledge of the machine about its surroundings and all the occlusions and limitations happening around it, and therefore, will increase the security of the system itself.

REFERENCES

[] Lemaignan, S., Warnier, M., Sisbot, A., Clodic, A. & Rachid, A. (2017). Artificial cognition for social human–robot interaction: An implementation
[] *Center for Educational Research and Innovation.* (2007) Understanding the Brain: The Birth of a Learning Science.
[] Boden, M. A. (1997).Creativity and artificial intelligence. Sussex: *School of Cognitive and Computing Sciences, University of Sussex.*
[] Nothman, J., Ringland, N., Radforda, W., Murphya, T. & Curran, J.R. (2012).Learning multilingual named entity recognition from Wikipedia.
[] Serj, M.F., Lavi, B. & Hoff, G. (2018). A Deep Convolutional Neural Network for Lung Cancer Diagnostic
[] Verma, M. (2014). Medical Diagnosis using Back Propagation Algorithm in ANN
[] Schoettle, B. & Sivak, M. (2014) A survey of public opinion about autonomous and self-driving vehicles in the U.S., the U.K. and Australia. *University of Michigan, Transportation Research Institute*.
[] Alcantarilla, P.F., Bartoli, A. & Davison, A.J. (2012). KAZE Features.
[] Andersson, O. & Marquez, S.R. (2016). A comparison of object detection algorithms using unmanipulated testing images Comparing SIFT, KAZE, AKAZE, and ORB.
[] Bay, H., Tuytelaars, T. & Gool, L.V. (2008). SURF: Speed Up Robust Features.
[] Ahmed Khan Tareen, S. & Saleem, Z. (2018). A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. International Conference on Computing, Mathematics and Engineering Technologies iCoMET 2018.
[] Krizhevsky, A. & Sutskever, I. (2012). ImageNet Classification with Deep Convolutional Neural Networks.
[] Kumar, C., Chawla, K. & Arora, S. (2018). The comparison between various object detection algorithms. Delhi: Maharaja Agrasen Institute of Technology.
[] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Jia, Y.... Rabinovich, A. (2014). Going Deeper with Convolutions. Michigan: Ann Arbor Magic Leap Inc. & University of North Carolina.
[] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. & Berg, A.C. (2016). SSD: Single shot multibox detection.
[] Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. University of Washington.
[] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. Google Research.
[] Girshick, R. (2015). Fast R-CNN. Microsoft Research.
[] Girshick, R., Ren, S., He, K. & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.
[] Dalal, N. & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. INRIA
[] Pavani, S., Delgado, D. & Frangia, A.F. (2010). Haar-like features with optimally weighted rectangles for rapid object detection.
[] Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Visual Geometry Group,
[] Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. Canada: Computer Science Department, University of British Columbia.
[] Chao, J., Al-Nuaimi, A., Schroth, G. & Steinbach, E. (2013). Performance Comparison of Various Features Detector-Descriptor Combinations for Content-based Image Retrieval with JPEG-encoded Query Images. Munich: Institute for Media Technology.
[] Dai, A.M. & Le, Q.V. (2015).Semi-supervised Sequence Learning. *Google Inc.*
[] Rojas, R. (1996). Neural Networks - A Systematic Introduction.
[] Argall, P. S. & R. J. Sica. (2007). Lidar (Laser Radar). *The Optics Encyclopedia*
[] Schwarz, B. (2010). Mapping the world in 3D. *Nature Photonics,* volume 4, pages 429–430
[] (2018). AR0820AT: CMOS Image Sensor, Datasheet.

[] *Frost & Sullivan*. (2016). LiDAR: Driving the Future of Autonomous Navigation. Analysis of LiDAR technology for advanced safety.
[] *Wind, Intel*. (2015). Autonomous driving: An eye on the road ahead.
[] NVIDIA. (2018). GEFORCE RTX 2080 ti. Retrieved from https://www.nvidia.com/en-us/geforce/graphics-cards/rtx-2080-ti/
[] NVIDIA. (1999). GEFORCE 256. Retrieved from https://www.nvidia.com/page/geforce256.html
[] Brechtel, S., Gindele, T. & Dillmann, R. (2014). Probabilistic Decision-Making under Uncertainty for Autonomous Driving using Continuous POMDP. Humanoids and Intelligence System Laboratories, Karlsruhe Institute of Technology.
[] Galceran, E., Cunningham, A.G., Eustice, R.M. & Olson, E. (2017). Multipolicy Decision-Making for Autonomous Driving via Changepoint-based Behavior Prediction: Theory and Experiment.
[] Bansal, M., Krizhevsky, A. & Ogale, A. (2018). Chauffeur Net: Learning to Drive by Imitating the Best and Synthesizing the Worst. Waymo Research.
[] Heckerman, D. (1995) A tutorial on Learning with Bayesian Networks. Microsoft Research, Advanced Technology Division.
[] Web source information:
https://ch.mathworks.com/products/matlab.html
[] Orzechowski, P.F., Meyer, A. & Lauer, M. (2019) Tackling occlusions & limited sensor range with set-based safety verification. Karlsruhe, Germany
[] Schratter, M., Bouton, M., Kochenderfer, M.J. & Watzenig, D. (2019) Pedestrian collision avoidance system for scenarios with occlusions. Stanford University & Graz University of Technology
[] Chung, W., Kim, S., Choi, M., Choi, J., Kim, H., Moon, C. & Song, J. (2009) Safe navigation of a mobile robot considering visibility of environment.
[] Reynolds, C.W. (1999) Steering Behaviors for Autonomous Characters. Sony Computer Entertainment America
[] Autoware papers
[] AASHTO (2004). Geometric Design of Highways and Streets.
[] "*Toyota invest $500 million in Uber for development of self-driving cars"* The Japan News Webpage: TMC & Uber, August 28th, 2018.
[] "*Autonomous taxi trials carrying passengers begin in Tokyo"* Kyodo, August 27th, 2018.
[] *"First self-driving car hub in Japan set for Tokyo's Shinagawa"* Nikkei Assian Review Webpage. September 14th,2018.
[] *"Uber self-driving car crash: Vehicle detected Arizona pedestrian 6 seconds before accident"* USA Today Article, Nathan BOMEY, May 24th, 2018.
[] *"Self-driving cars test won't need permission to use public, say NPA guidelines"* The Japan News Webpage: April 11th, 2016.
[] *"Driver Shortage in Japan Speeds Up Commercial Use of Self-Driving Tech"* Transport Topics Website: August 28th, 2018.
[] *"Open Innovation for Fully Automated Driving"* Japan Government Webpage: Winter 2018.
[] *"Who's Winning the Self-Driving Car Race?"* Elisabeth BEHRMANN & David WELCH. Bloomerg Hyperdrive
[] *"Which companies are making driverless cars?"* Christina MERCER & Tom MACAULAY. 2019. Techworld Webpage.
[] *"Japan: Autonomous Driving Patents Rated; American And Japanese Top Competitors*" Mondaq Webpage. []
[] Nikkei Asian Review. Website. Online resource available here:
"*https://asia.nikkei.com/Business/Business-Trends/Self-driving-cars-in-Japan-clear-insurance-hurdle"*
[] *"The economic development of Japan"* (Original name: *"Tojokoku Nippon no Ayumi: Edo kara Heisei madeno Keizai Hatten*") Kenichi OHNO. 2005, Tokyo.
[] *"Japan works to launch self-driving car system by 2020"* Sean KEANE. June 4th, 2018 [4] *"Japan looks to launch driverless car system in Tokyo by 2020"* Stanley WHITE. June 4th, 2018
[] IOT for all, Website. *"The 5 Autonomous Driving Levels Explained*" Isabel HERNER. October 23th, 2017. Online resource available here:
"https://www.iotforall.com/5-autonomous-driving-levels-explained/"

[] *“Self-driving cars hit European speed bump. The future is now, but the EU could find itself left behind.”* Joanna PLUCINSKA, Joshua POSANER. August 25th, 2016.
[] Allen & Overy. *“Autonomous and connected vehicles: navigating legal isues”*. 2017.
[] *“EU motors ahead with rules for self-driving cars Bid to catch up with China and US and become world leader in autonomous vehicles”* Peter CAMPBELL. May 14th, 2018
[] *“Psychological roadblocks to the adoption of self-driving vehicles”* Azim SHARIFF, J.F. BONNEFON, Iyad RAHWAN. September 2017.
[]*“Breaking the psychological barrier to autonomous vehicle adoption*” Frederic WEILLER, 15th August, 2018.
[] *“Trust in driverless cars: Investigating key factors influencing the adoption of driverless cars”* Kanwaldeep KAUR, Giselle RAMPERSAD. Journal of Engineering and Technology Management, Volume 48, 2018, Pages 87-96.
[] The Electronic Chronicles: “*Electric Motor Power and HP Ratings*” Ted DILLAR, July 21st, 2014.
[] *“The cost of keeping a car in Japan”* City-cost webpage. Jan 23rd, 2017. Based on (https://www.keikenkyo.or.jp/)
[] The World Bank Data. Avalaible here:
https://web.archive.org/web/20140209085318/http://data.worldbank.org/indicator/IS.VEH.NVEH.P3?order=wbapi_data_value0+wbapi_data_value+wbapi_data_value-last&sort=desc
[] “*Motor Vehicle Statics of Japan”* JAMA, 2016.
[] Japan census 2015. Data from Website PopulationPyramid.
[] “*Where are the world’s best metro system?”* CNN Travel, Edward FALZON, July 12th, 2017.
[] “*Statics Brief: Urban Public Transport in the 21st Century*” Advancing Public Transport, October 2017.
[] “*NTT Data, Gunma University to test self-driving ride services on publics road near Yoyosu*” Kazuaki NAGATA, September 13th, 2018.

ANNEX A

RQT Graph from Autoware simulation:

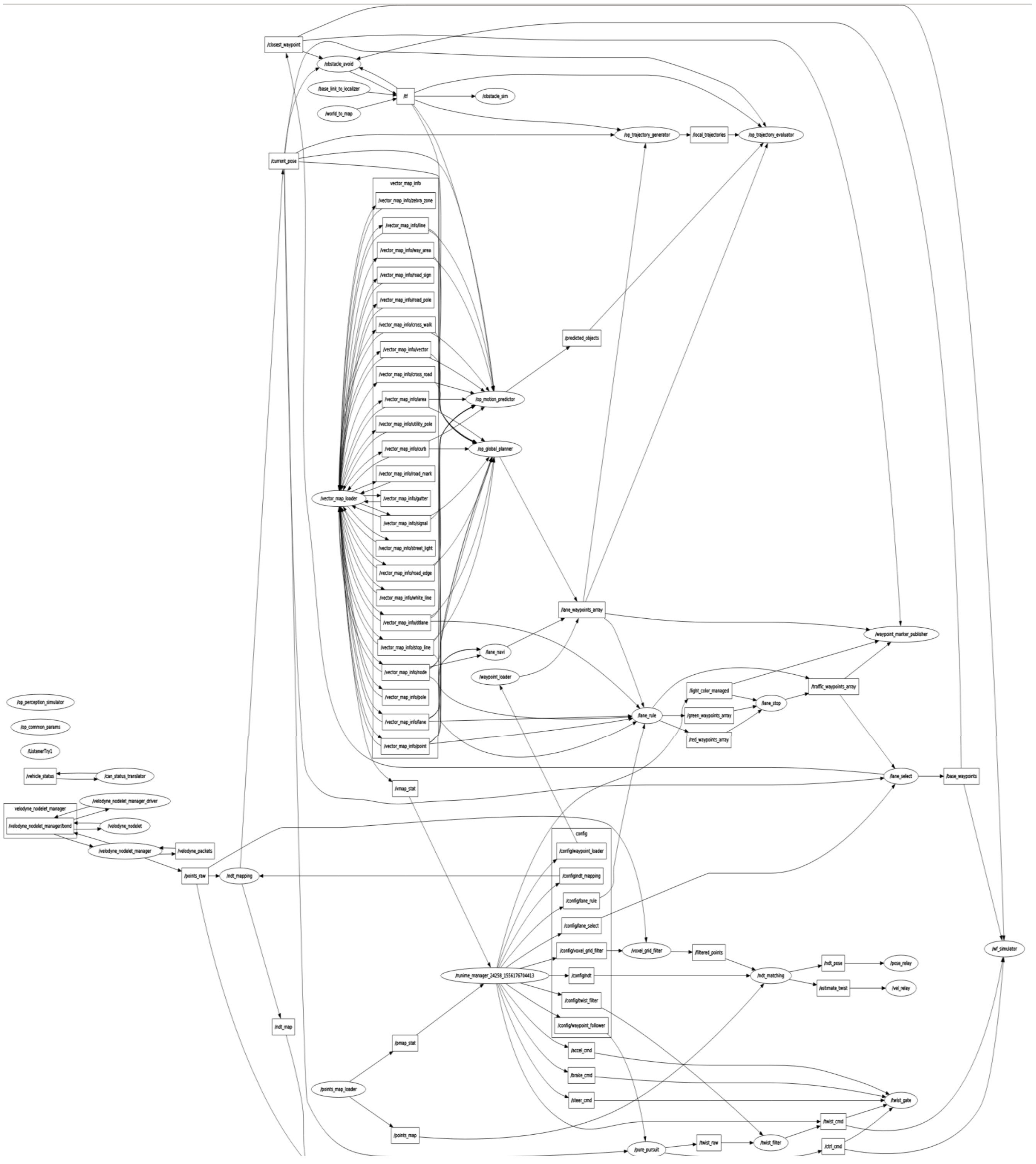


Figure 61 RQT from Autoware simulation