

E-CAR
PROJECT CHALLENGE UTAC

I4 - PAP - Projet n°36 - 02/04/2021

Mentors : Mme Naila BOUCHEMAL, Mr Jae Yun JUN KIN

Auteurs : Franck ZHANG, Mohamed Elyes ZAHAR, Pascal CHEN, Pierre MOREAU, Lucas RIETSCH, Justine REYNAUD

Sommaire

Abstract

Glossaire

1. Introduction

2. Contexte du projet

2.1. Concours

2.2. Problématique

3. Fonctions et contribution des acteurs

3.1. Acteurs

3.1.1. Répartition du travail

3.1.2. Outils de communication

3.2. Planning

4. Description du projet

4.1. Etat de l'art

4.2. Description de la solution

4.3. Caractère innovant

5. Valorisation Recherche et Innovation

5.1. Veille technologique

5.1.1. Lidar

5.1.2. Caméra

5.1.3. Capteurs

5.2. Description de l'approche

5.3. Réalisation du prototype / expérimentations

5.3.1. Environnement de développement

5.3.1.1. The constructism

5.3.1.2. Installation machine virtuel & d'ubuntu

5.3.1.3. Installation Ros Kinetic

5.3.2. Scénario de fonctionnement / montage

5.3.2.1. Mise en place du projet

5.3.2.2. Lancement du projet

5.4. Evaluation du prototype

5.5. Problèmes rencontrés

5.5.1. Machine virtuelle sur nos PC

5.5.2. Problème installation ROS

5.5.3. Robot HUSARION

5.5.4. Problème PC de l'école

5.5.5. Problème sur le fonctionnement du PFE

6. Conclusion et perspectives

7. Références bibliographiques

Abstract

Platooning, which is the convoy of autonomous cars, has already shown its effectiveness through various studies as to its ability to smooth traffic and improve road safety. Indeed, if there are already many researches about "Platooning" techniques in environments such as highways or expressways, but few are present in urban environments, which poses the added value of this challenge. The urban infrastructures that could be used by the autonomous car system can be characterized by intelligent traffic lights, buildings, traffic signs or even camera systems coupled with artificial intelligence. We will focus on the research and development of platooning techniques with obstacle detection and trajectory planning, and we will propose a complete strategy describing an autonomous car system in urban environment, using V2X control systems and communication. We will propose a complete strategy describing an autonomous car system in an urban environment, using control systems and V2X communication. This study will contribute to the implementation of systems to make urban traffic safer and smoother in order to reduce the related greenhouse gas emissions.

Le platooning, qui est le convoi de voitures autonomes, a déjà montré son efficacité à travers différentes études quant à sa capacité de fluidifier le trafic et d'améliorer la sécurité routière. En effet, il existe déjà de nombreuses recherches concernant les techniques de "Platoon" dans des milieux tels que les autoroutes ou les voies rapides mais peu sont présentes en milieu urbain, ce qui pose la valeur ajoutée de ce challenge. Les infrastructures urbaines que pourraient utiliser le système de voiture autonome peuvent être caractérisées par des feux de croisement intelligents, des bâtiments, des panneaux de signalisation ou encore des systèmes de caméras couplés à des intelligences artificielles. Nous nous intéresserons à la recherche et le développement de technique de formation de platoon avec une détection d'obstacles et une planification de trajectoire. Nous proposerons une stratégie complète décrivant un système de voiture autonome en milieu urbain, grâce à des systèmes de contrôle et une communication V2X. Cette étude pourra contribuer à la mise en place de systèmes permettant de sécuriser davantage le trafic urbain et de le fluidifier afin de réduire les émissions de gaz à effet de serre qui lui sont liées.

Glossaire

V2I	Vehicle To Infrastructure	Des messages sont échangés entre le véhicule et l'infrastructure (en général l'équipement utilisateur UE) qui prennent en charge ce type de communication.
V2N	Vehicle To Network	Les messages sont échangés via EPS (Evolved Packet System) entre l'équipement utilisateur (UE) et le serveur d'applications qui prennent en charge la communication V2N13.
V2X	Vehicle To Everything	Des messages sont échangés entre le véhicule et n'importe quelles autres plateformes (infrastructure, réseaux, piétons, ..)
SAE	Society of Automotive Engineers	Organisation internationale comptant plus de 84 000 membres qui échangent des informations et des idées pour tout ce qui touche à l'ingénierie des véhicules.
NHTSA	(National Highway Traffic Safety Administration)	Agence fédérale américaine chargée de la sécurité routière.

1. Introduction

Dans un véhicule autonome et connecté, la fusion multicapteurs est une nécessité pour mener à bien les différentes tâches de détection d'obstacles et de la planification de la trajectoire.

Aujourd'hui de plus en plus de véhicules utilisent la technologie de communication V2X qui permet au véhicule de communiquer avec tout l'environnement qui l'entoure (les autres véhicules, les piétons, le réseau électrique, les infrastructures et aux dispositifs).

C'est dans ce contexte-ci que nous avons choisi le projet de véhicule autonome en collaboration avec le groupe PFE qui s'intègre dans le cadre du challenge UTAC. Le projet PFE se chargera de l'implémentation de la communication V2X. Dans un premier temps, ils devront maintenir l'inter-distance entre les véhicules, ensuite permettre une communication V2I et V2N afin d'adapter la vitesse des véhicules en fonction de la circulation.

Pour notre partie, nous allons nous intéresser à la détection d'obstacles, à l'évitement ainsi qu'à la planification de la trajectoire. En effet, avec l'utilisation de différents algorithmes que nous allons vous présenter, le véhicule sera capable d'identifier l'obstacle. Puis lorsque la distance qui le sépare de celui-ci devient insuffisante, le véhicule sera capable d'opter pour le meilleur chemin possible.

Pour la conception de notre projet, il était prévu initialement qu'on ait à disposition un robot husarion. Cependant avec la crise sanitaire, le projet a été réorienté. Nous avons donc dû réaliser notre solution sous forme de simulation.

2. Contexte du projet

2.1. Concours

L'UTAC CERAM, qui est l'abréviation de Union Technique de l'Automobile, du motocycle et du Cycle Centre d'Essais et de Recherche Appliqué à la Mobilité, organise leur tout premier challenge dédié aux étudiants. Ce concours est organisé avec le soutien de la Société des Ingénieurs de l'Automobile et a pour but de favoriser le développement des métiers tournés autour du domaine des véhicules autonomes. Pour ce challenge, nous candidatons dans la catégorie épreuve libre, qui n'a pas d'objectifs définis, et qui accepte donc toutes formes de projets que ce soit sous forme de robot ou simulation.

2.2. Problématique

Nous réalisons ce projet dans le but de participer au challenge de l'UTAC CERAM dans la catégorie libre. Pour pouvoir remporter cette catégorie, nous devons apporter une innovation dans le domaine du véhicule autonome. Dans le cadre de notre projet, l'innovation apportée est la fusion multicapteurs couplée à la communication V2X afin de permettre au véhicule d'éviter les obstacles, de planifier sa trajectoire et enfin de circuler dans un milieu urbain avec d'autres véhicules tout en maintenant une distance de sécurité.

3. Fonctions et contribution des acteurs

3.1. Mentors

Notre projet était encadré tout le long de l'année par deux personnes. Madame BOUCHEMAL et Monsieur JUN KIM ont été les mentors de ce PPE. Madame BOUCHEMAL agissait en temps qu'experte en communication et Monsieur JUN KIM quant à lui en tant qu'expert sur toute la partie robotique, capteur et fusions de données.

3.2. Présentation de l'équipe

3.2.1. Organigramme



3.2.2. Répartition du travail

Pour davantage d'efficacité, le groupe PPE s'est divisé en trois sous groupes. Un premier sous-groupe composé de Pierre et Pascal qui ont fait des recherches pour l'installation de tout l'environnement de ROS sur Ubuntu. Ils se sont occupés aussi de la mise en place de l'environnement de simulation sur Ubuntu et sur The Construct Sim avec l'utilisation des logiciels Gazebo et RVIZ. Franck et Justine, qui forment le deuxième binôme du groupe, se sont occupés de l'évitement d'obstacles en utilisant les données issues du LIDAR pour les traitées avec les différents algorithmes comme les méthodes des tentacules. Enfin le dernier groupe, formé par Elyes et Lucas, s'est occupé de mettre en commun la partie du projet du

groupe PFE avec la nôtre et se sont occupés de la fusion des données issues des multicapteurs.

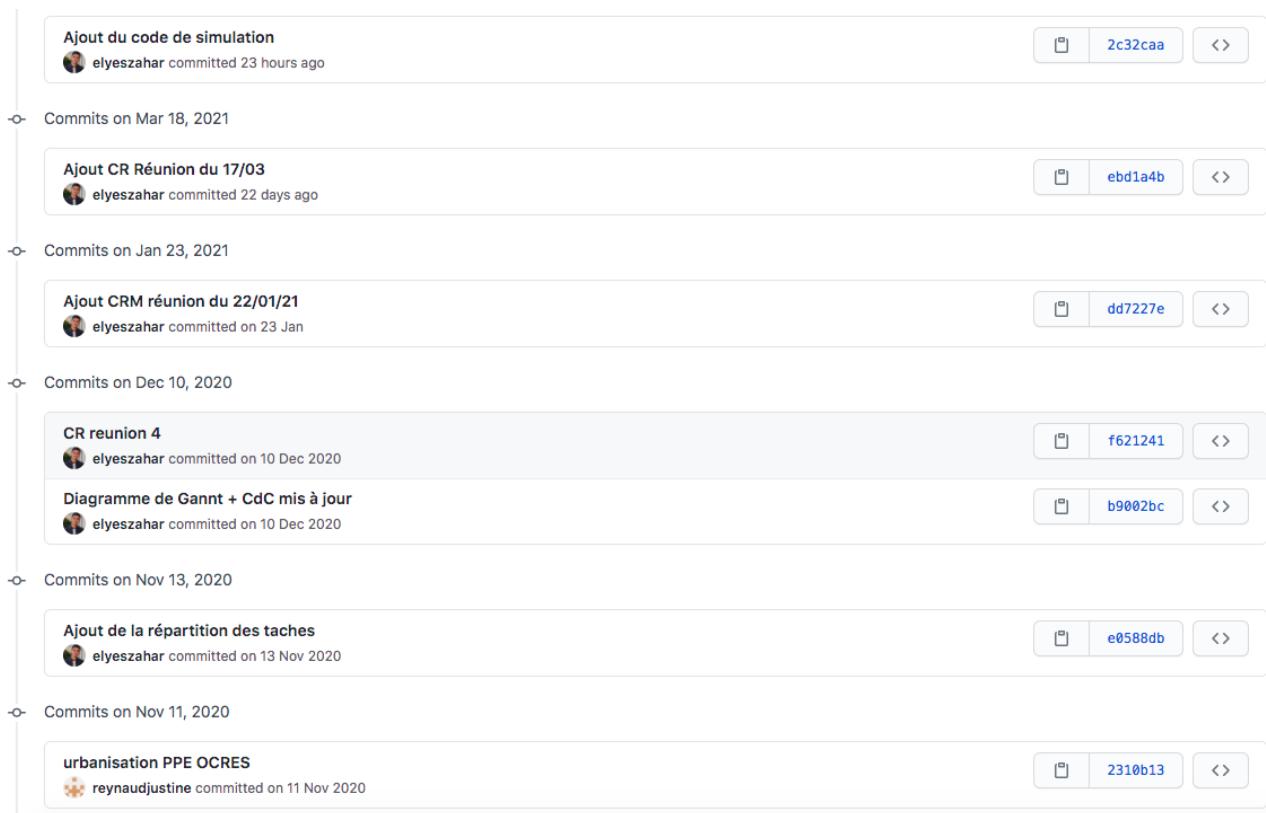
Afin de suivre l'avancement entre les groupes, nous faisions des comptes-rendus chaque semaine tous ensemble. Et pour chaque sous-groupe, nous avons désigné une personne qui communiquait avec les deux autres désignés pour se prévenir des différentes tâches réalisées au cours de la semaine.

3.2.3. Outils de communication

Lors de la réalisation du projet nous avons utilisé différents outils de communication afin de nous organiser et être le plus efficace possible. Un des outils majeurs qui nous permet de communiquer à n'importe quel moment est Facebook Messenger ainsi que Teams, cela nous permet de communiquer simplement et à tout moment pour se répartir les tâches et éclaircir les zones d'ombres.

De plus, pour la communication de documents et code pratiques nous avons utilisé tout au long du projet github. En effet grâce à cette plateforme à chaque fois qu'un document (planning, recherches, tâches à faire, codes...) était édité nous le mettions sur notre github.

https://github.com/elyeszahar/PPE_2020_VoitureAutonome



- o- Commits on Mar 18, 2021
 - Ajout du code de simulation
  [2c32caa](#) 
- o- Commits on Jan 23, 2021
 - Ajout CR Réunion du 17/03
  [ebd1a4b](#) 
- o- Commits on Dec 10, 2020
 - Ajout CRM réunion du 22/01/21
  [dd7227e](#) 
 - CR reunion 4
  [f621241](#) 
 - Diagramme de Gannt + CdC mis à jour
  [b9002bc](#) 
- o- Commits on Nov 13, 2020
 - Ajout de la répartition des tâches
  [e0588db](#) 
- o- Commits on Nov 11, 2020
 - urbanisation PPE OCRES
  [2310b13](#) 

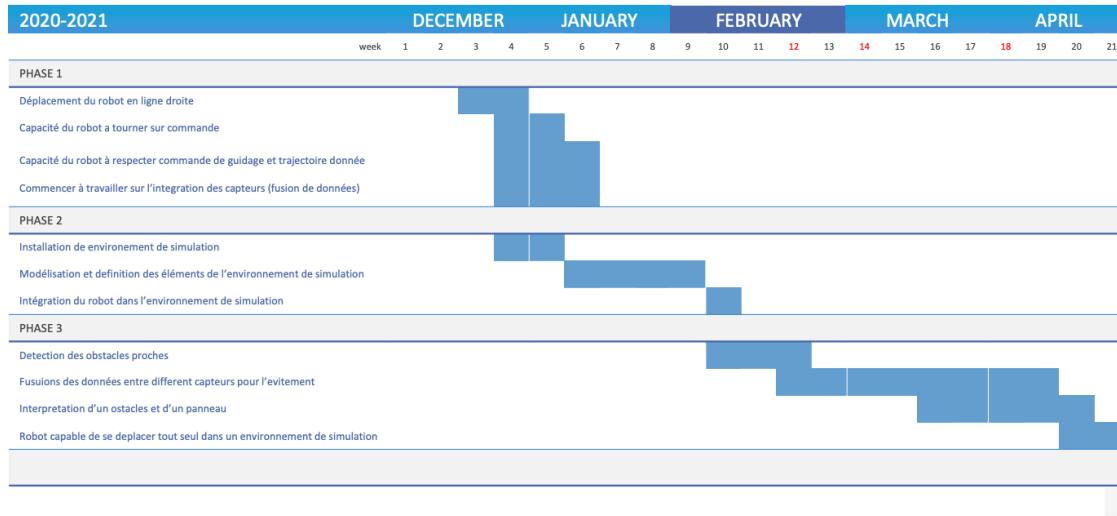
Enfin, les CRM nous ont permis de garder une trace du travail effectué et des sujets de chaque réunion. Cela a contribué à l'organisation de l'équipe et notamment à nous fixer des objectifs à court, moyen et long terme.

3.3. Planning



Resources Gantt Chart

Timetable



2

La partie technique de notre projet s'est déroulée en trois phases :

- La première, de mi-décembre à mi-janvier, où l'on s'est occupé de l'installation de l'espace de travail ainsi que de faire avancer le robot en ligne droite dans l'environnement de simulation. Nous nous sommes occupés de faire tourner le robot sur commande, nous avons essayé de faire en sorte que le robot respecte sa trajectoire donnée.
Enfin nous avons commencé à faire la fusion des données issues des multicapteurs.
- La deuxième phase qui s'est déroulée de fin décembre à début février était dédiée à la modélisation de l'environnement (création de la map) sur Gazebo.
Puis nous nous sommes occupés d'insérer le robot dans cette map.
- Enfin pour la troisième phase, de début février jusqu'à mi-avril, nous nous sommes occupés de créer un algorithme qui détecte et évite les obstacles à partir des données (que nous avons analysées et traitées) issues des différents capteurs.

4. Description du projet

4.1. Etat de l'art

De nombreuses solutions répondent aux enjeux relatifs à la sécurité routière. C'est un domaine qui demande beaucoup d'intérêt dans la mesure où il s'agit de la vie d'Hommes qui est en jeu. Aujourd'hui, de nombreuses entreprises se hâtent à développer un véhicule autonome et connecté:



Tout d'abord nous pouvons parler de Waymo, qui est la voiture autonome de Google, la plus aboutie actuellement sur le marché international. Le système de pilotage automatique via une Intelligence artificielle utilise un LIDAR, des radars, une caméra 360°, un récepteur GPS ainsi que des capteurs sur les roues motrices. C'est une voiture qui a une autonomie de niveau 3. La limite du projet est que le véhicule est dans l'incapacité d'interpréter les signes d'un agent de la route, de la police.



La France aussi a vu le développement de voitures autonomes sur son territoire tel que la voiture Drive 4U de Valeo qui est une voiture sans conducteur capable de rouler dans n'importe quelle ville. Le véhicule est équipé exclusivement de capteurs Valeo, comme le Radar MB79 et le Scala. De nombreux tests ont été fait dans la ville de Paris afin de mettre en situation ce véhicule.

Néanmoins ce véhicule n'intègre pas la communication V2X dans le cadre de la signalisation d'obstacles ou de changements dans l'environnement du véhicule.

A travers ces solutions déjà existantes, nous pourrions améliorer certains protocoles comme les aspects de la détection d'obstacles et de planification de trajectoire en améliorant par exemple les algorithmes des chemins viables qui peuvent être implémenté et testé avec les librairies en libre service de ROS (Robot Operating System). Par ailleurs, nous pouvons remarquer que l'utilisation de capteurs LIDAR et de caméras comme la solution de Google est la meilleure piste à retenir pour le développement de notre solution. A la différence de Valeo, nous utiliserons des capteurs de fabricants différents afin de pouvoir viser le produit qui propose un rapport qualité-prix qui nous intéresse. La plus value de notre projet serait aussi de combiner plusieurs recherches déjà existantes. Certains utilisent essentiellement le Lidar, d'autres des radars, ou caméras. Nous voulons que notre solution soit constituée de tous ces différents capteurs.

4.2. Description de la solution

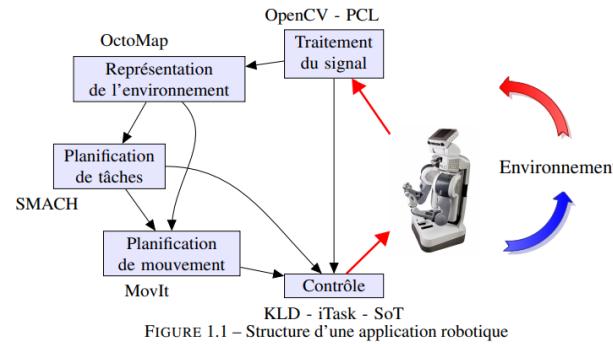


FIGURE 1.1 – Structure d'une application robotique

4.2.1. Environnement de simulation

La solution sera sous forme d'une simulation d'un déplacement autonome, incluant la planification de la trajectoire ainsi que la détection d'obstacles. Pour cela nous allons utiliser l'environnement Robot Operating System (R.O.S), c'est un environnement open source de développement logiciel pour la robotique. Son développement est aujourd'hui mené par l'Open Robotics. ROS est un méta-système d'exploitation standardisé qui apporte différentes fonctionnalités pour la robotique.

Le fonctionnement de l'environnement ROS est assez simple, c'est une application qui suit la structure représentée dans la figure 1. Elle consiste à générer une commande afin que le robot puisse agir sur un environnement à partir de sa perception de celui-ci.

Nous allons pour cela utiliser :

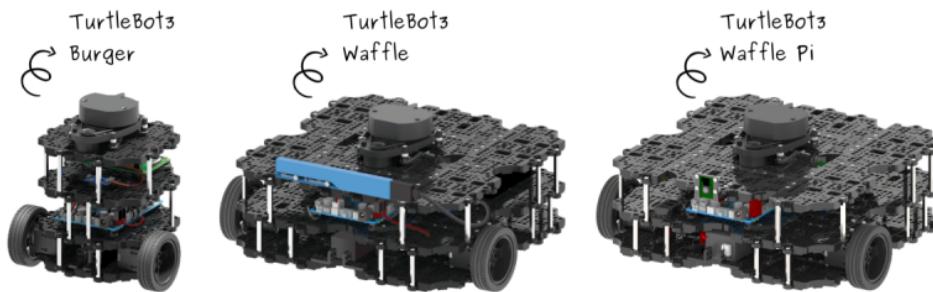
- Des nœuds, qui sont des processus qui effectuent un calcul ou une tâche. Ils ont la capacité de s'enregistrer auprès du nœud maître ROS et de communiquer avec les autres nœuds du système. L'idée de conception de ROS est que chaque nœud est indépendant et interagit avec les autres nœuds en utilisant la capacité de communication de ROS. L'une des forces de ROS est qu'une tâche particulière, comme le contrôle d'un robot mobile à roues, peut être séparée en une série de tâches plus simples. Ces tâches peuvent inclure la perception de l'environnement à l'aide d'une caméra ou d'un scanner laser, la réalisation d'une carte, la planification d'un itinéraire, la surveillance du niveau de la batterie du robot et la commande des

moteurs qui entraînent les roues du robot. Chacune de ces actions peut consister en un nœud ROS ou une série de nœuds pour accomplir les tâches spécifiques.

- Des Topics, qui sont dans ROS, les informations échangées entre les nœuds (ex : position du robot, données issues des capteurs ...) , les nœuds peuvent récupérer ou modifier ces informations que contiennent les topics.

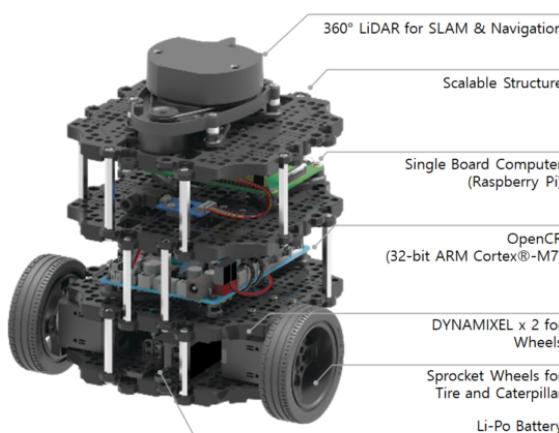
4.2.2. Turtlebot3

Pour la simulation, nous avons décidé de choisir le Turtlebot3 plutôt que le robot Husarion qui était prévu au départ, car il est plus facile à appréhender dû aux nombreuses librairies et ressources déjà disponibles.



Il existe plusieurs versions du TurtleBot3, nous avons utilisé la version Burger qui est constitué :

TurtleBot3 Burger



1 x LiDAR 360° : La technologie LiDAR est une technologie de télédétection qui mesure la distance entre le capteur et une cible. La lumière est émise par le LiDAR et se dirige vers sa cible. Elle est réfléchie sur sa surface et revient à sa source. Comme la vitesse de la lumière est une valeur constante, le LiDAR est capable de calculer la distance le séparant de la cible.

1 x Raspberry Pi 3 model b: Est un nano-ordinateur monocarte à processeur ARM de la taille d'une carte de crédit conçu par des professeurs du département informatique de

l'université de Cambridge dans le cadre de la fondation Raspberry Pi 3.

2 x Roues dynamiques : Les roues du robot lui permettent de se déplacer linéairement et angulairement. Chaque roue possède un servomoteur Dynamixel série X lui permettant de

se déplacer sur toute surface.

1 x châssis : Ce sont des plaques de gaufre qui jouent un rôle important sur le maintien du robot bien que leurs tailles soient aussi petites que la paume d'une main.

4.2.3. Gazebo

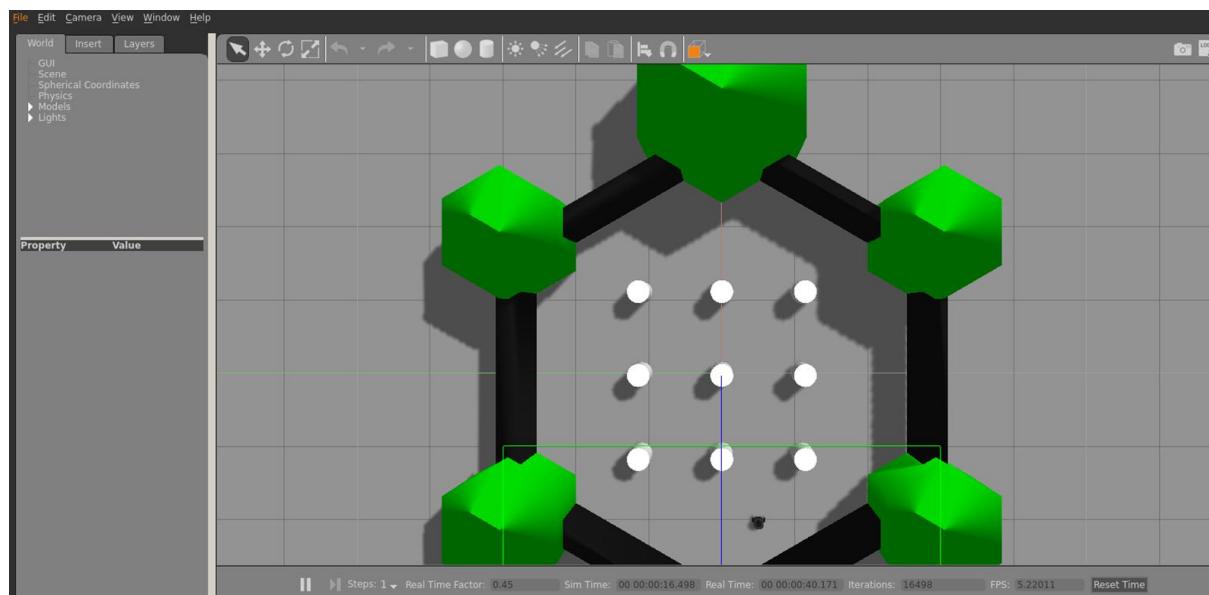
Nous avons eu besoin d'utiliser l'extension Gazebo pour mener à bien notre projet.

GAZEBO est un simulateur 3D permettant de simuler des robots dans différents environnements. Ce simulateur est dit "open source", cela signifie qu'il est libre de droit et peut être modifié. C'est le même principe avec LINUX.

Il permet de représenter des robots articulés dans des environnements complexes intérieurs ou extérieurs et est souvent utilisé dans la recherche robotique et dans l'intelligence artificielle. Nous pouvons l'utiliser sur ROS et même utiliser les API (interfaces de programmation), qui permettent de contrôler le robot lors des simulations. Le logiciel permet de faire des simulations réalistes. Le robot peut pousser des objets ou encore rouler sur le sol. Gazebo utilise plusieurs moteurs physiques tels qu'ODE (Open Dynamics Engine) ou encore Bullet.

Le logiciel peut ajouter et simuler des capteurs au robot comme les capteurs de force, de contact, laser et des caméras.

Pour pouvoir créer un environnement de simulation, nous devons créer un fichier ayant pour extension « .world », le langage utilisé de ce fichier est le XML et doit contenir toutes les informations nécessaires pour générer l'environnement. (gravité, ombre, ciel, ...).

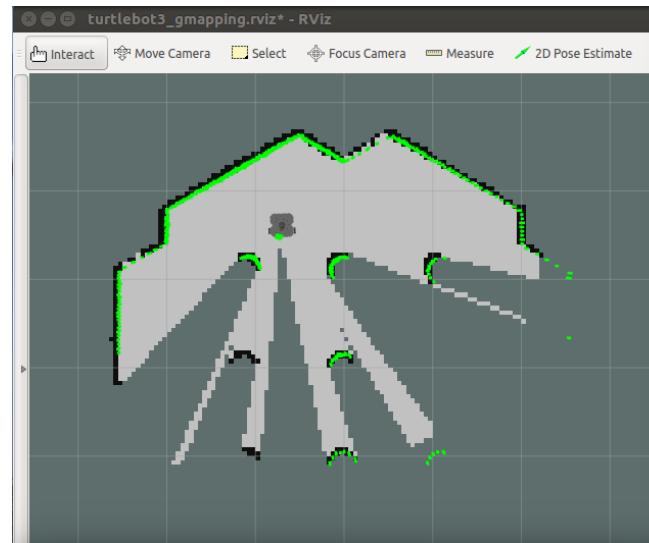


Simulateur GAZEBO

4.2.4. Rviz

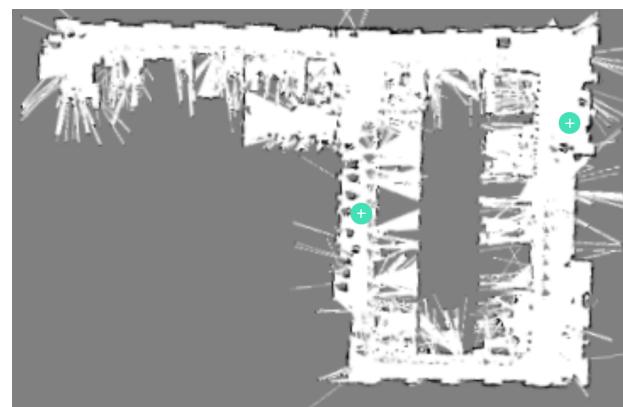
Il nous a fallu aussi utiliser Rviz, qui est une interface graphique permettant d'afficher les modèles des robots, des cartes de navigation reconstruites par des algorithmes de SLAM, d'interagir avec le robot, d'afficher des images, des points 3D fournis par des caméras 3D.

Dans le cadre de notre projet nous l'avons utilisé afin de pouvoir visualiser notre robot mais aussi, ce que perçoit le capteur lors du déplacement du robot dans son environnement.



Simulateur Rviz

4.2.5. Construction d'une map



Représentation de la carte avec GMAPPING

Afin de réaliser l'environnement où se déplacera le robot nous avons la méthode du GMAPPING, cette technique permet de créer la carte, elle a été intégrée en 2010 dans le logiciel ROS. Cette technologie utilise l'odométrie, grâce à cette méthode on peut estimer la position du robot qui se déplace sur la carte.



Représentation de la carte avec une coupure du LIDAR

La photo que nous avons ci-dessus montre ce qu'il se passe lors de la cartographie. Elle montre l'impact que peut avoir la coupure du LIDAR et de l'odométrie. Comme le montre l'image, nous avons une représentation ressemblante par rapport à la première image. Sauf que le fait d'avoir coupé le LIDAR n'affectera pas la création de la carte. Par contre le fait d'avoir coupé l'odométrie a engendré énormément d'erreurs, certaines parties de la carte deviennent floues ou même se sont dupliquées.

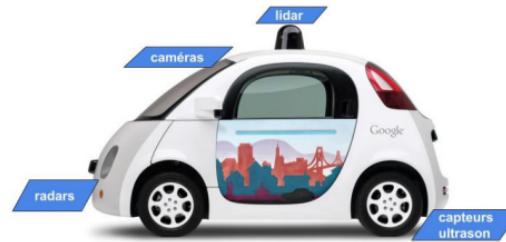


Représentation de la carte avec un défaut d'odométrie

Cet exemple nous montre l'impact que peut avoir un défaut d'odométrie sur la cartographie. Comme on peut le constater, la création de la carte devient beaucoup plus compliquée à réaliser du fait que le robot n'arrive pas à se situer dans son environnement. Ce qui donne une carte compliquée à comprendre dû à la superposition des informations.

5. Valorisation Recherche et Innovation

Dans cette partie, nous allons détailler toutes nos recherches qui ont mené à l'implémentation de notre solution. Nous nous sommes aidés d'articles de recherches, de travaux en cours et de tutoriels.



5.1. Veille technologique

Afin de procéder à la veille technologique, nous nous sommes appuyés sur les articles suivants :

- Laure Nuninger, Krištof Oštir, Ziga Kokalj, Ales Marsetic. Lidor. Acquisition, traitement et analyse d'images LiDAR pour la modélisation des paléo-reliefs de la plaine littorale du Languedoc oriental.. 2008, pp.95. fffhalshs-00483129f
- Véhicule autonome : Les systèmes de perception embarqués , : STPI/P6/2018 2019

5.1.1. LIDAR

Ayant servi pour cartographier et étudier la Lune, détecter des vestiges de la civilisation Maya ou mettre à jour une forteresse viking au Danemark, le LIDAR est la technologie qui permet d'effectuer des relevés topographiques inégalés.

De nos jours, les LIDARs sont utilisés par la majeure partie des industriels intéressés par les voitures autonomes. Le LIDAR est donc un faisceau laser qui balaye l'environnement du véhicule en permanence. On mesure le délai entre l'envoi du faisceau et le moment où il revient jusqu'à l'émetteur-récepteur. Grâce à la vitesse de déplacement du laser et le temps récupéré par l'émetteur-récepteur, le LIDAR peut calculer instantanément la distance le séparant de l'objet. Il est donc semblable au radar (qui utilise les ondes radio) et au sonar (qui utilise les ondes acoustiques) mais lui, utilise des ondes électromagnétiques.

Ce capteur se situe souvent dans la calandre du véhicule afin d'avoir une vision sur la route. Il peut mesurer la distance qu'avec un objet à la fois. Dans le domaine du véhicule autonome, il est utilisé pour pouvoir voir l'environnement où se situe la voiture. Il permet de détecter les obstacles, les informations que le LIDAR peut récupérer, peut-être ensuite

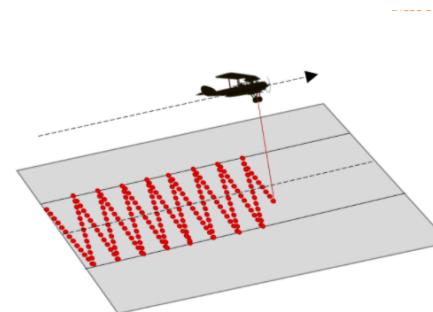
envoyé sur un ordinateur de bord et ainsi influencer les trajets de la voiture.

Une des limites du LIDAR concernant les voitures autonomes est qu'il n'est pas capable de faire autre chose que de calculer des distances. Comme le cite Elon Musk, CEO de Tesla, "utiliser des optiques incapables de lire les panneaux de signalisation n'a aucun sens".

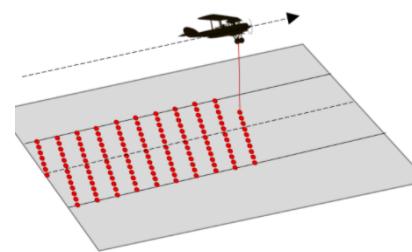
Le LIDAR reste néanmoins aujourd'hui la technologie la plus performante pour des projets automobiles, d'agriculture ou d'archéologie.

Il existe différents modèles de balayage des systèmes LiDAR en fonction de la méthode de balayage utilisée :

- Le miroir oscillant : Les lasers utilisant des miroirs oscillants dirigent les impulsions laser dans deux directions de la bande, créant ainsi un motif dentelé sur le sol. La largeur de la bande est donnée par l'angle d'inclinaison maximum du miroir. La densité de points qui en résulte n'est pas uniforme car les impulsions laser ont tendance à s'accumuler sur les bords de la bande en raison de la décélération du miroir.

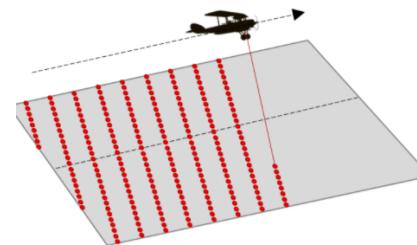


- Le miroir polygonal tournant : Ce système crée un modèle de balayage composé de lignes parallèles obliques par rapport à la direction de la progression de la plate-forme. Ici, le balayage se fait toujours dans la même direction. La largeur de la bande dépend de la largeur de la face du miroir polygonal.

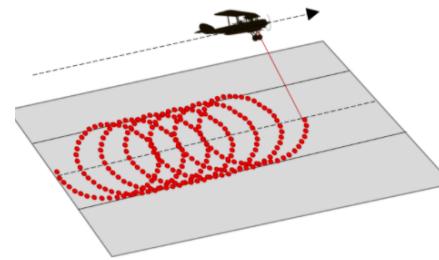


- Le miroir tournant : Avec son axe de rotation colinéaire avec la direction du faisceau laser émis par la source, le miroir rotatif permet de balayer presque 360°. Comme dans l'exemple précédent, le schéma de balayage est composé de lignes parallèles

obliques par rapport à la direction du mouvement du système. Cette fois, la taille de la bande dépend de la portée maximale du système LiDAR .

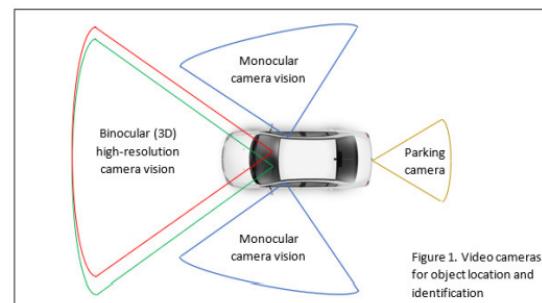


- Les prismes ou réseaux dispersifs : Les prismes dispersifs ainsi que les réseaux dispersifs tournent autour du même axe pour générer des motifs de balayage semblables à des roses. Par exemple, les prismes de Risley - également appelés prismes en coin - peuvent former des cercles ou des motifs de balayage en forme de roses.



Les limites d'utilisations d'un Lidar est : qu'il est sensible aux conditions météorologiques, à l'environnement (réflexion spéculaire ou angle de faisceau de rayonnement limité) et à l'environnement extérieur. Deuxièmement, le radar laser actuel est coûteux en termes de coût de traitement des données obtenues. Ils ne peuvent pas percevoir les informations de couleur dans des tâches telles que la reconnaissance des panneaux de circulation.

5.1.2. CAMÉRA



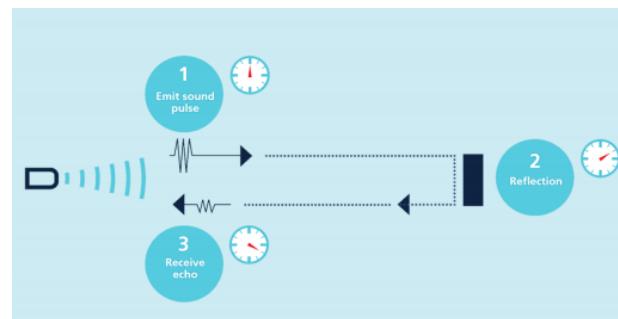
Dispositif optique connu depuis l'Antiquité, la caméra est un système permettant la formation d'une image sur une toile ou un écran. Les premières caméras ont été développées en 1889 suite à l'invention de la pellicule photographique. De nombreux types de caméras existent, telles que les caméras analogiques, argentiques, d'astronomie, hautes vitesses ou encore intelligentes.

Dans le domaine des voitures autonomes, les caméras sont utilisées pour offrir au système central une vision à 360 degrés. Elles serviront à détecter les lignes blanches au sol pour ajuster la trajectoire de la voiture.

D'une manière plus générale, les caméras servent dans notre cas à respecter le code de la route. Feux tricolores, panneaux, clignotants des autres usagers, elles analysent l'environnement extérieur pour adapter, en fonction de ce dernier, son comportement.

Les limites d'utilisation des caméras sont : qu'elles sont affectées par l'intensité de la lumière. En effet, le fonctionnement des caméras est facilement affecté par la lumière externe, telles que les fortes pluies, les nuits sombres et les conditions de luminosité brillante, et il est impossible de déterminer avec précision les obstacles.

5.1.3. Les radars



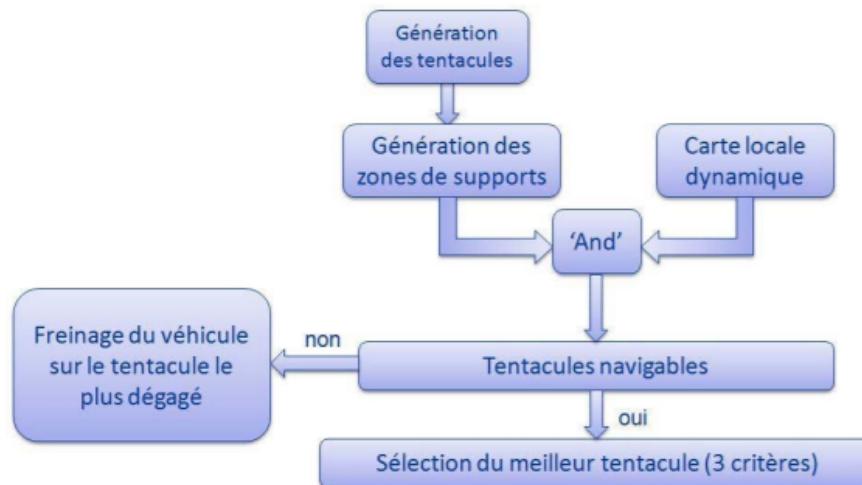
C'est un capteur et émetteur d'ondes radios qui permet de mesurer la distance entre le radar et un objet. Le capteur et l'émetteur sont côté à côté. L'émetteur envoie des ondes. Si elles sont renvoyées par un obstacle, le capteur les détecte. En fonction du temps entre l'émission et la réception et connaissant la vitesse du son, le système calcule la distance séparant le radar de l'obstacle. Il peut alors alerter l'ordinateur central, (qui vérifiera cette information avec d'autres capteurs, au cas où l'information serait erronée) s'il détecte un obstacle sur le chemin de la voiture. L'ordinateur peut alors décider de freiner ou d'arrêter le véhicule. Les radars peuvent aussi servir pour détecter le trafic environnant.

Les limites d'utilisations des capteurs : comparé au capteur lidar, sa précision angulaire est légèrement inférieure et le véhicule cible est perdu dans la courbe. Si plusieurs objets de la sonde sont proches les uns des autres, il peut être difficile de l'identifier. Par exemple, le radar peut traiter deux petites voitures à proximité comme une grosse voiture, envoyant un mauvais signal de proximité. Cependant, comparé au lidar, le radar peut déterminer

précisément la vitesse de déplacement relative et la vitesse des objets en mouvement en utilisant l'effet Doppler.

5.2. Description de l'approche (Méthode des tentacules/ Fusions multicapteurs)

Il existe plusieurs méthodes mathématiques de détection d'obstacles, l'utilisation de la **méthode des tentacules** est la plus performante. Elle consiste à utiliser un ensemble d'antennes virtuelles appelées « tentacules » dans une grille d'occupation égo-centrée liée au véhicule, et qui représente l'environnement proche du véhicule avec la position des obstacles. Une fois les trajectoires virtuelles créées, une évaluation de ces dernières est effectuée afin de choisir la meilleure selon un critère défini . Le meilleur tentacule ou trajectoire de référence, sera exécuté par le véhicule à chaque pas de calcul. Cette méthode présente l'avantage d'être très réactive, permettant d'éviter des collisions et de circuler dans un environnement incertain sans besoin d'une grande quantité d'informations a priori. En outre, elle n'accumule pas les données et utilise une carte locale égo-centrée, évitant ainsi le problème du SLAM (Simultaneous Localization and Mapping). Par ailleurs, la méthode des tentacules est l'une des plus simples qui permet au véhicule de rouler en sécurité dans un environnement inconnu. Elle est aussi la plus rapide car il n'y a pas accumulation des données. En outre, elle permet une circulation sûre tout en étant capable de traverser des passages étroits. Nous la décrirons plus en détail dans le paragraphe suivant.



A chaque pas de calcul (toutes les 100ms), des tentacules en forme d'arcs de cercle sont générés. Une zone de support de chaque tentacule est générée permettant de s'assurer que le véhicule puisse circuler sur ce dernier en toute sécurité (sans risque de collision). Une fois les zones de supports générées, chaque tentacule est évalué à l'aide d'une carte locale dynamique afin d'éliminer tout tentacule non navigable. L'ensemble des tentacules dont la longueur, au premier obstacle, est supérieure à la distance de collision est appelé «

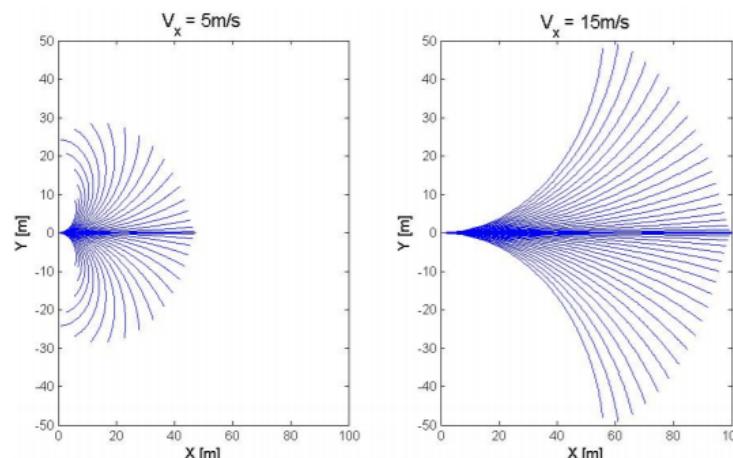
tentacules navigables ». Après cette évaluation, si plusieurs tentacules sont navigables, le meilleur tentacle est choisi suivant trois critères :

- le dégagement du tentacle (distance au premier obstacle) ;
- le changement de braquage ;
- et la trajectoire globale .

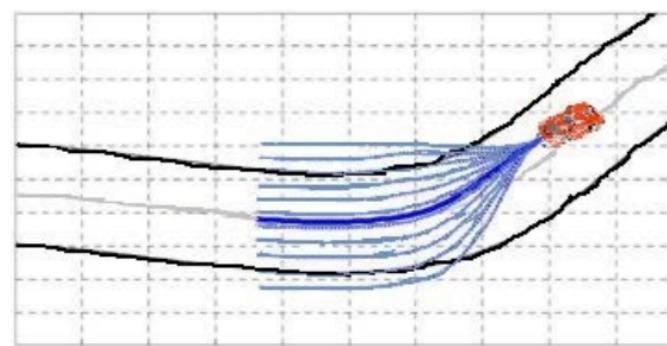
La navigation par cette méthode peut se décomposer en trois étapes : la génération des tentacules, la détermination des tentacules navigables et le choix du meilleur tentacle.

Tous les tentacules sont représentés dans le système de coordonnées locales du véhicule. Ils commencent au centre de gravité du véhicule et ont la forme d'arcs de cercle. Dans la littérature plus récente, les tentacules sont générés suivant plusieurs formes : nous pouvons citer les tentacules parallèles à la route et les clohoïdes.

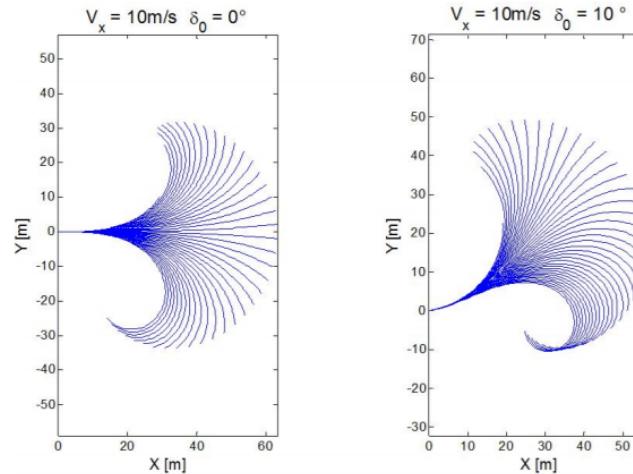
Tentacules en forme d'arc de cercle :



Tentacules parallèles à la route :



Tentacules en forme de clohoïdes :



Une fois les tentacules générés, une zone de support de chaque tentacle est déterminé. La zone de support est une zone autour du tentacule prenant en compte la longueur du véhicule et une distance de sécurité permettant de vérifier si le véhicule peut circuler sur le tentacule sans risque de collision. Elle est caractérisée par la distance de la zone de support. Après génération des zones de support, chaque tentacule est évalué en effectuant une correspondance entre la zone de support de ce dernier et la carte locale (obtenue par la fusion des informations de la perception et la localisation). Cette évolution permet de vérifier si le tentacule est navigable ou non. Si le tentacule peut être parcouru jusqu'à la distance de collision, il est qualifié de « tentacule navigable ».

Pour décider du "meilleur" tentacule, tous les tentacules navigables sont évalués. Dans l'algorithme de base, trois valeurs sont calculées et combinées de manière linéaire pour choisir le tentacule optimal [Hundelshausen et al., 2008] :

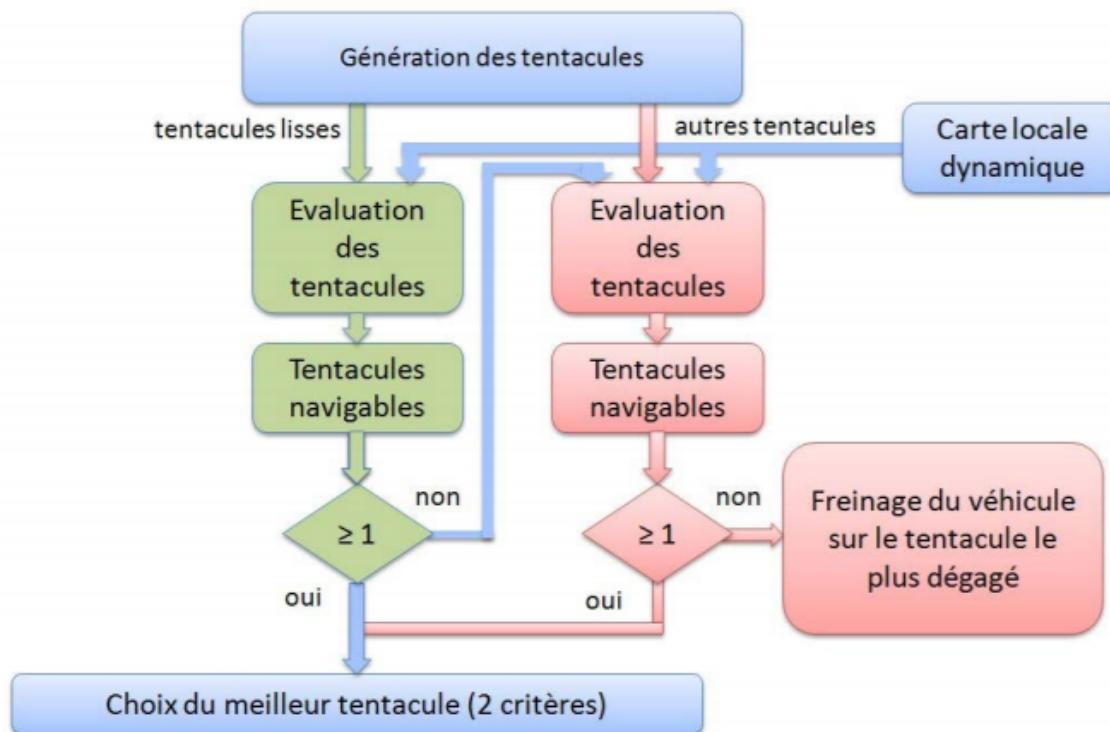
- le premier critère v_{libre} permet d'évaluer si le tentacule est libre d'obstacle ;
- Le deuxième critère v_{lisse} permet d'évaluer si le braquage est lisse (ou brusque) pour suivre le tentacule. Ce paramètre se calcule à partir du braquage courant ;
- Le troisième critère $v_{trajectoire}$ permet d'évaluer si le tentacule se rapproche de la trajectoire globale.

Ces trois critères permettent ainsi d'éviter les obstacles, d'avoir une conduite confortable tout en suivant la trajectoire globale. Une fois ces critères estimés, ils sont combinés avec pondération de la manière suivante :

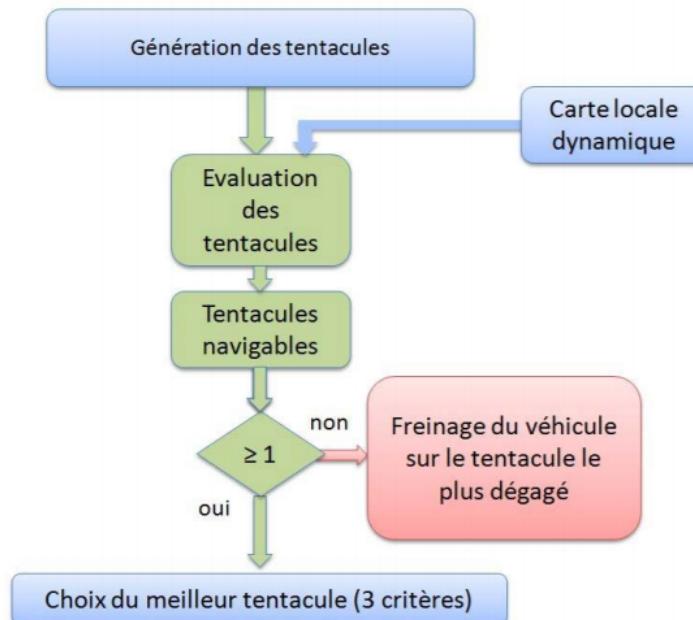
$$v_{tentacule} = a_1 * v_{libre} + a_2 * v_{lisse} + a_3 * v_{trajectoire}$$

où a_1 , a_2 et a_3 sont des constantes. La combinaison de ces critères fait apparaître le problème de pondération qui devient très difficile et délicat car dépendant du contexte de conduite. Les coefficients de pondération sont nécessairement modifiés selon le contexte.

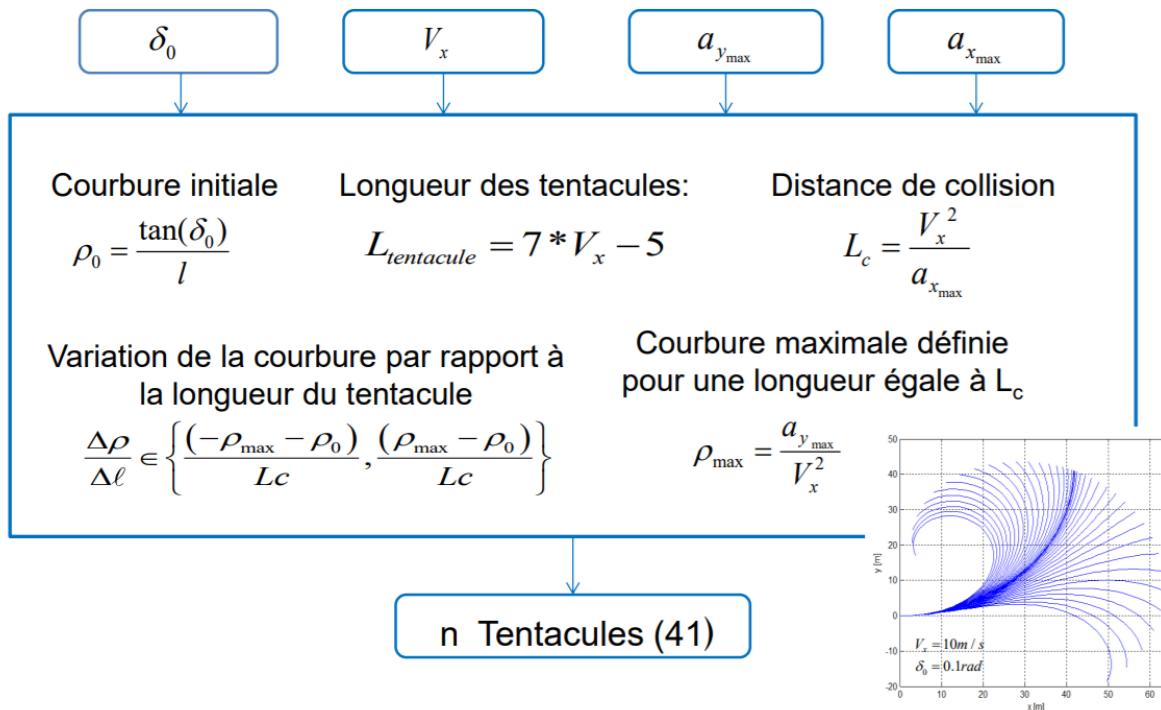
- Algorithme de la planification trajectoire de la méthode tentacule en forme d'arc de cercle:



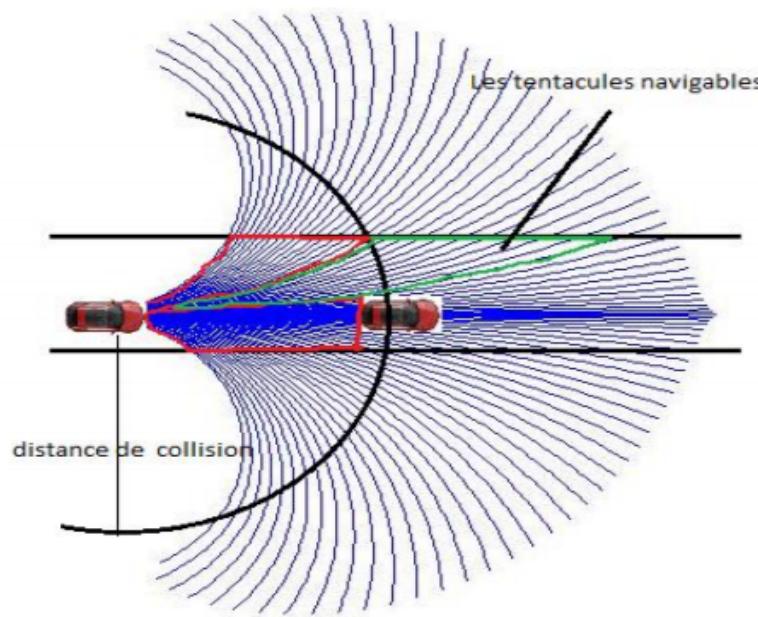
- Algorithme de la planification trajectoire de la méthode tentacule sous forme de clothoïde :



- Formules pour trouver un tentacule navigable



où δ_0 est le braquage initial, V_x est la vitesse du véhicule, $a_{y_{\max}}$ est l'accélération latérale maximale et $a_{x_{\max}}$ est la décélération longitudinale maximale



Distance de collision $L_c = \frac{V_x^2}{a_{x_{\max}}}$

5.3. Réalisation du prototype / expérimentations

5.3.1. Environnement de développement

5.3.1.1. The Construct Sim

The Construct Sim est une plateforme pour apprendre la robotique. Cette plateforme propose des cours avec de la théorie sur la robotique et sur la programmation, que ce soit pour un niveau débutant ou avancé. Leurs formations sont dédiées pour des particuliers, des entreprises ou pour l'éducation. Sur cette plateforme, nous possédons notre espace personnel dans lequel nous pouvons créer des projets ou visualiser des projets d'autres personnes (semblable à github). Lorsque nous accédons à notre espace personnel, nous avons la possibilité d'utiliser l'environnement de ROS pour faire des projets.

Dans le cadre de notre PPE, nous avons contacté le groupe PFE qui nous a présenté leur partie détaillée. Lorsque nous avons appris que notre projet se ferait sous forme de simulation, ils nous ont conseillé d'utiliser le site The Construc Sim qui permet de faire des simulations de l'environnement de ROS en ligne. Cependant ils ont rencontré plusieurs difficultés avec cette interface et ne pouvaient pas faire fonctionner entièrement leur projet.

Nous avons réussi à faire fonctionner des simulations de notre partie sur ce logiciel, mais comme il nous est nécessaire de récupérer leur partie pour clôturer le projet, nous n'avons pas pu aller plus loin sur cette plateforme.

5.3.1.2. Installation machine virtuel & d'Ubuntu

Pour la réalisation de notre PPE nous avons installé sur nos ordinateurs une machine virtuelle afin de pouvoir avoir l'environnement d'ubuntu. Nous avons utilisé Oracle VM VirtualBox comme logiciel de virtualisation.

Puis une fois installé nous avons mis en place le système d'exploitation Ubuntu qui nous permettait d'installer ROS.

5.3.1.3. Installation Ros Kinetic

Tutoriel d'installation Kinetic :

- Pour commencer, veillez à configurer votre ordinateur pour accepter les logiciels de packages.ros.org.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu \$ (lsb_release -sc) main"> /etc/apt/sources.list.d/ros-latest.list'
```

- Ensuite il faut configurer vos clés

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Si toutefois vous rencontrez des problèmes de connexion au serveur de clés, essayez de remplacer hkp://pgp.mit.edu: 80 ou hkp://keyserver.ubuntu.com: 80 dans la commande au-dessus.

A la place d'écrire la commande apt-key, vous pouvez aussi utiliser curl qui sera plus utile si vous êtes derrière un serveur proxy:

```
curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE8868B172B4F42ED6FBAB17C654' | sudo apt-key ajouter -
```

- Vérifier si l'index de paquet Debian est à jour :

```
sudo apt-get mise à jour
```

Puis nous installons le paquet recommandé : **Desktop-Full Install: (Recommended)** :

```
sudo apt-get install ros-kinetic-desktop-full
```

Pour ne pas initialiser l'environnement de ROS à chaque lancement d'un terminal, il faut entrer cette commande :

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Et au final pour utiliser les outils de ROS, nous installons les dépendances pour faciliter les installations des futurs ROS paquets :

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential
```

Ainsi que **rosdep**, qui permet d'installer les systèmes de dépendances plus facilement :

```
sudo apt install python-rosdep
```

Puis il faut initialiser rosdep :

```
sudo rosdep init
```

```
rosdep update
```

Voilà, nous avons installé les essentiels pour utiliser ROS.

5.3.2. Scénario de fonctionnement / montage (workspace + Packages + nodes)

5.3.2.1. Mise en place du projet

Commencez par créer votre espace de travail, ici nous l'avons appelé **catkin_ws**:

```
mkdir ~/catkin_ws
```

```
cd ~/catkin_ws
```

Ensuite dans votre dossier **catkin_ws**, créer un répertoire source que l'on nomme **src** :

```
mkdir src
```

On initialise ensuite le workspace

```
catkin_make
```

Après cette commande, deux dossiers se sont créés : **devel** et **build**.

Puis il faut cloner notre projet dans un répertoire vide.

```
git clone https://github.com/elyeszahar/PPE\_2020\_VoitureAutonome/
```

Ensuite, nous allons récupérer les dossiers de **PPE_2020_VoitureAutonome/catkin_ws/src** et **PPE_2020_VoitureAutonome/catkin_ws/devel** et mettre ces dossiers dans le répertoire que nous avons créé : **catkin_ws/src** et **catkin_ws/devel**.

Et relancez la commande :

```
catkin_make
```

5.3.2.2. Lancement du projet

Dans le système d'exploitation Ubuntu, démarrez un terminal pour pouvoir lancer le projet. A chaque ouverture d'un terminal, il faut actualiser les environnements de variable insérer la commande

```
source ~/catkin_ws/devel/setup.sh
```

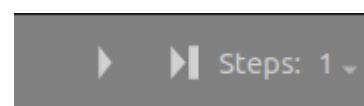
Nous utilisons Turtlebot 3 model burger. A chaque ouverture du terminal, il faut importer le bon modèle robot :

```
export TURTLEBOT3_MODEL=burger
```

Et l'étape finale est de lancer la simulation :

```
roslaunch ecebot ecebot_turtle.launch
```

L'environnement de simulation Gazebo sera affiché mais la simulation est en pause, il faut appuyer sur espace ou cliquer sur le bouton play pour lancer la simulation



5.4. Evaluation du prototype

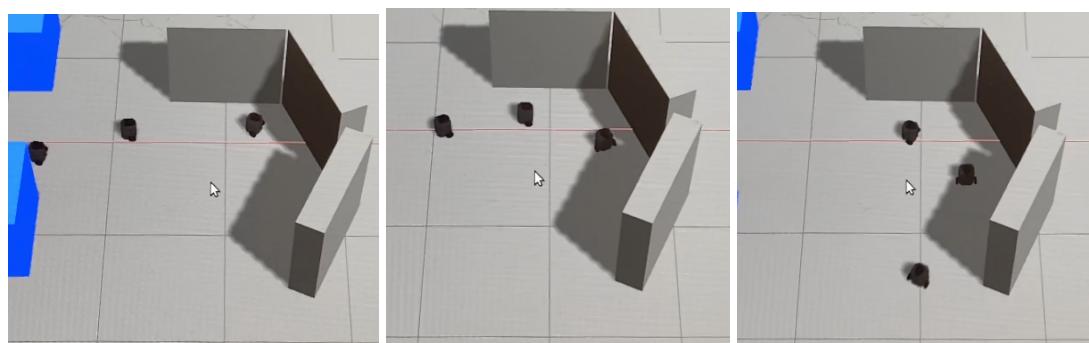


Figure 1

Figure 2

Figure 3

Ci-dessus la représentation de notre solution. Le but de nos robots est de suivre une trajectoire et en cas d'obstacles de la modifier. Dans un premier temps, le robot de tête va détecter un obstacle sur sa trajectoire et s'arrêter (**Figure 1**).

Par la suite, ce robot va se tourner vers la droite car il constate qu'il a un obstacle et qu'il ne peut pas y aller tout droit, ni à gauche.

Il commence sa rotation afin de retrouver une trajectoire et continuer sa route. Après avoir retrouvé sa trajectoire, le robot de tête envoie des messages (**Figure 4**) aux robots suiveurs afin qu'ils puissent à leurs tours modifier leurs trajectoires (**Figure 2**). Une fois que la communication a été faite entre les différents robots et que leurs trajectoires ont été modifiées, ils reprennent leur chemin de manière normale.

```
Vitesse : 0.3m/s
[ INFO] [1617878167.847732834, 58.281000000]: Envoie message CAM au vehicule : 2
[ INFO] [1617878167.971325057, 58.381000000]: Envoie message CAM au vehicule : 2
Vitesse : 0.3m/s
[ INFO] [1617878168.112828652, 58.481000000]: Envoie message CAM au vehicule : 2
[ INFO] [1617878168.239257670, 58.582000000]: Envoie message CAM au vehicule : 2
Obstacle detecte !!!
Vitesse : 0m/s
Rotation...
[ INFO] [1617878168.367695386, 58.681000000]: Envoie message CAM au vehicule : 2
[ INFO] [1617878168.496852614, 58.785000000]: Envoie message CAM au vehicule : 2
Obstacle detecte !!!
Vitesse : 0m/s
Rotation...
```

Figure 4 : Messages envoyés lors de la détection d'obstacles

Dans les messages que nous pouvons récupérer sur un terminal. Nous avons dans un premier temps la vitesse du robot qui est de 0.3m/s. Lors d'une détection d'obstacle un message est envoyé aux robots suiveurs afin de les arrêter. Un message apparaît sur le terminal afin de prévenir les robots suiveurs que le robot de tête est en train de tourner et donc de modifier sa trajectoire. Les messages sont envoyés à chaque fois que le robot de tête détecte quelque chose avec son LIDAR.

5.5. Problèmes rencontrés

5.5.1. Machine virtuelle sur nos PC

Dans un premier temps, nous avons rencontré pas mal de problèmes avec nos machines virtuelles, car pour certains lors de l'installation des logiciels nous permettant de réaliser le projet, on manquait de place pour les logiciels (ROS, GAZEBO, RVIZ). Donc nous avons dû régler ce problème et réinstaller de nouvelles machines.

Après avoir fait cela, nous avons eu un autre souci avec la machine car lorsqu'on lançait les simulations 3D sur GAZEBO, la machine virtuelle commençait à ralentir de plus en plus et avait un temps de latence énorme. Nous avons donc dû à nouveau la réinstaller et lui octroyer plus de mémoire vive. Ce qui lui a permis de lancer les simulations sans problème.

5.5.2. Problème installation ROS

Nous avons eu des problèmes avec le logiciel ROS lors de la réalisation de notre projet. Trois versions s'offraient à nous. ROS Kinetic, ROS Noetic et ROS Melodic. Nous avons au départ du projet opté pour le ROS Noetic car c'était la version la plus récente du logiciel. Lors de nos

différents tests pour afficher le robot, nous avions eu énormément d'erreurs car la plupart des tutos qu'on pouvait trouver sur internet étaient liés avec la version ROS Kinetic.

Or lorsque l'on réglait les soucis de compatibilité entre Noetic et Kinetic, il se pouvait que le robot s'affiche parfaitement sur une des machines virtuelles et que sur les autres il y ait encore d'autres soucis.

Le groupe PFE a lui aussi choisi cette version. Donc nous avons aussi opté pour ROS Kinetic.

5.5.3. Robot HUSARION



Robot HUSARION

Comme énoncé ci-dessus, nous avons eu des soucis avec la compatibilité entre ROS Kinetic et ROS Noetic. Notre robot arrivait à fonctionner sous ROS Noetic. Mais lorsqu'on lui ajoutait des choses à réaliser venant de ROS Kinetic cela posaient énormément de problèmes.

Nous n'arrivions pas à récupérer les données venant du capteur LIDAR situées sur le robot, afin de les traiter. Ainsi lorsque le robot arrivait près d'un obstacle on ne pouvait pas récupérer les données issues du LiDAR et du coup il ne l'évitait pas.

Nous avons donc choisi de changer de robot. Nous avons pris le Turtlebot 3 burger afin de réaliser notre projet et ainsi avoir le même robot que le PFE.

5.5.4. Problème PC de l'école

Etant donné que nous avons des problèmes avec nos machines virtuelles, on nous a mis à disposition deux PC au 5e étage de l'ECE disposant du système d'exploitation linux. Cela nous a permis d'avancer sur notre projet et d'éviter les éventuels problèmes de latence qu'on a pu avoir lors des simulations 3D. Or, nous avions décidé de changer de ROS et de passer sur une version plus ancienne (ROS Kinetic).

Le problème est que, comme la version de ROS est plus ancienne, elle n'était pas compatible avec le Ubuntu qui était installé sur les PC. Initialement le UBUNTU installé était la version 20.04 LTS et nous avons pour cela dû passer à la version 16.04 LTS pour pouvoir faire fonctionner le nouveau logiciel.

De plus, il se pouvait que sur une des deux machines certaines commandes fonctionnaient mais pas sur l'autre.

5.5.5. Problème sur le fonctionnement du PFE

Notre projet est en partenariat avec un groupe PFE. Notre tâche consiste à détecter des obstacles, à les éviter, à planifier une trajectoire et à fusionner les données issues des capteurs. Quant à eux, leur projet était sur le platooning et la communication entre les véhicules. Pour cela nous avons récupéré leur travail et essayer de le faire fonctionner sur les machines de l'école.

Le problème : les chemins d'accès qui étaient introduits dans les différents dossiers, fichiers et codes n'étaient pas bons. Nous avons donc dû tous les changer, par la suite il y a eu un problème de compatibilité entre nos logiciels, nous étions sur ROS Noetic et eux sur le Kinetic.

6. Conclusion et perspectives

Bien que nous ayons rencontré des problèmes réguliers au cours de ce projet, nous avons tout de même réussi à remplir les objectifs de notre valorisation. Après avoir étudié les technologies utilisées sur le marché, leurs évolutions, leurs utilisations dans différents secteurs d'activités, nous avons pu étudier dans quels cadres elles sont aujourd'hui mises en œuvre.

C'est donc un état de l'art complet que nous avons réalisé, qui permettra aux prochains groupes, qui le souhaitent, de reprendre notre travail. Nous nous sommes rendu compte que l'analyse de ce qui existait déjà était un vrai plus pour réaliser notre travail.

Nous sommes tout de même déçus de ne pas avoir eu le temps de terminer la partie technique de notre cahier des charges initial. Notre but était de faire déplacer en autonomie un robot grâce à des capteurs LIDAR, radars et/ou caméras. Bien que nous ayons réussi une première version de déplacement partiel du robot sur une map statique et prédéfinie, nous aurions souhaité aller plus loin et pousser nos recherches en matière de développement.

Nous avons quelques perspectives pour notre projet.

Nous avons réussi à permettre au robot de détecter des obstacles et de changer sa direction. Ce qui serait bien d'améliorer c'est de permettre au robot, lorsqu'il détecte un obstacle, de changer de trajectoire de manière naturelle comme si nous étions en train d'éviter un poteau en voiture. C'est-à-dire d'anticiper la détection pour pouvoir ainsi ralentir petit-à-petit, et ne pas marquer nécessairement un arrêt.

Nous aimerais aussi permettre au robot de suivre sa trajectoire en fonction de ce qu'il voit en utilisant la caméra et donc de suivre les lignes qu'il y a sur la route.

Au vue des conditions de travail que nous avons eu cette année, causées par le COVID, nous avons été contraint de réaliser ce projet, prévu initialement avec un robot physique, en simulation sur machine virtuelle. Le concevoir en vrai pourrait permettre d'avoir un projet concret et ainsi mieux comprendre comment fonctionne chaque composant du robot.

7. Références bibliographiques

- Documentation de ROS, <http://wiki.ros.org/fr>
- Gilles Tagne Fokam. *Commande et planification de trajectoire pour la navigation de véhicules autonomes*. Autre. Université de Technologie de Compiègne, 2014. Français. NNT : 2014COMP2004 tel-01160233
- Lidar, Les Horizons, 9 février 2020,
<https://leshorizons.net/lidar-les-yeux-technologiques/>
- NHTSA, Wikipedia,
https://fr.wikipedia.org/wiki/National_Highway_Traffic_Safety_Administration
- Planification de Trajectoires pour la Navigation des Véhicules Autonomes, Alia Chebly, 2015, http://jjcr2015.imag.fr/img/JJCR2015_Chebly.pdf
- Projet AUTONET2030,
<http://hal-mines-paristech.archives-ouvertes.fr/hal-01063484/document>
- Projet MAVEN, <http://maven-its.eu/>
- SAE International, Wikipedia, https://fr.wikipedia.org/wiki/SAE_International
- Tutoriel, Enansakib, Github,
<https://github.com/enansakib/obstacle-avoidance-turtlebot>
- Tutoriel Turtlebot3, Robotis e-manual,
<https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>
- Valeo, Drive4U, la première voiture à arpenter les rues de Paris, 2018,
<https://www.valeo.com/fr/valeo-drive4u-la-premiere-voiture-autonome-a-arpenter-es-rues-de-paris/>
- We're building the World's Most Experienced Driver, Waymo, <https://waymo.com/>