

Rapport de stage

de M. GROSSARD Mathieu

En vue de l'obtention de diplôme de Master " Électronique,
énergie électrique, automatique"
Parcours "Systèmes Embarqués et Communicant"

Promotion 2022 – 2023

Fusion et tracking multi-capteurs



Tuteur pédagogique

M. LAUFFENBURGER

Tuteur professionnel

M. CAZENAVE

Fiche archivage pour la bibliothèque universitaire

1. L'étudiant

NOM - Prénom : GROSSARD Mathieu.....

Année scolaire :2022-2023

Diplôme(s) préparé(s) :

☐ Ingénieur : ☐ GI
☐ ASE
☐ IR
☐ Méca
☐ TF

Statut :

☒ Étudiant
☐ Contrat de Professionnalisation
☐ Apprentissage

☐ Mastère Spécialisé en Ingénierie Textile



☒ Master M2 ☐ EEA – ASI
☒ EEA – SEC

☐ Méca – IveM
☐ Méca - MMF

2. L'entreprise

Dénomination de l'entreprise : Altran Technology & Engineering Center.....

Adresse complète : 4 avenue Didier Daurat 31700 Blagnac

Pays : France

Secteur d'activité (Mécanique, médical, ...) : Conseil en Ingénierie.....

Type d'entreprise (PME, SSII, ...) : SAS

Nombre de salariés : 55 000 Nombre de cadres :55 000

3. Le rapport de stage / de la thèse professionnelle

Titre du rapport : Le stage a pour sujet le développement de différents sous-systèmes de notre système d'aide à la conduite pour tramway.....

Mots-clés pour décrire le sujet de votre mémoire (5 maximum) :

Capteur, Fusion, Kalman, ROS, Carla

Résumé en français de votre mémoire (250 mots maximum) :

Le stage s'est déroulé au sein du projet STRASS, un projet de recherche et innovation de Capgemini Engineering. Il se concentre sur le développement de divers sous-systèmes pour l'aide à la conduite des tramways. L'objectif principal est de détecter les risques de collisions dans les zones à risques, en garantissant une sécurité accrue. Nous nous concentrons ici sur la fusion de données des capteurs afin de prédire la position et la vitesse des véhicules.

Nous présentons ici les différentes composantes de notre solution. Nous utilisons la simulation développée sous Carla et le logiciel ROS pour explorer différentes stratégies de traitements des données capteurs. Nous utilisons dans un second temps les résultats de ces traitements comme entrée pour la fusion des données capteurs à l'aide d'un filtre de Kalman. Nous examinons les techniques de mise en forme et d'homogénéisation des données, essentielles pour les rendre compatibles, avant de les soumettre au filtre de Kalman. De plus, nous présentons les diverses applications possibles de ce filtre, ouvrant ainsi de nouvelles perspectives.

Cette étude s'intègre dans le domaine de l'aide à la conduite des tramways.

La fusion des données constitue l'élément clé de la fiabilité et de la précision de notre solution. Les résultats prometteurs obtenus soulignent le potentiel de notre système, qui contribuera significativement à la sécurité et à la prise de décision dans les zones à risques.

Résumé en anglais de votre mémoire (250 mots maximum) :

The internship takes place within the STRASS project, a research project at Capgemini Engineering. It focuses on developing various subsystems for tramway driving assistance. The main objective is to detect collision risks in high-risk areas, ensuring enhanced safety. Specifically, this internship focuses on the data fusion aspect of sensor data to predict position and velocity of various vehicles.

This report details the various part of our solution. Using Carla simulation and ROS programming, we thoroughly explore the data processing strategies for the sensors. These carefully designed processing techniques prepare the data for fusion using a Kalman filter.

Data fusion plays a crucial role in our approach. We examine data formatting and homogenization techniques, essential for making the data compatible before subjecting it to the Kalman filter. Additionally, we present various potential applications of this filter, thereby opening new perspectives.

This study adds to the field of tramway driving assistance. Data fusion is the key element for the reliability and precision of our solution. The promising results obtained underscore the considerable potential of our system, which will significantly contribute to safety and decision-making in high-risk areas.

4. Diffusion

- ☒ Le contenu n'est pas confidentiel, diffusion immédiate du rapport.
- ☐ Le contenu est confidentiel, diffusion du rapport après une période d'attente de :
☐ 1 an ☐ 2 ans ☐ 3 ans ☐ 5 ans ☐ 10 ans

Approbations et diffusion par l'entreprise

5. Identifications

Nom et prénom du tuteur professionnel

CAZENAVE Paul

Nom et prénom du stagiaire

GROSSARD Mathieu

Nom de l'organisation

Altran Technology & Engineering Center

6. Approbation de diffusion

J'atteste par la présente avoir pris connaissance du rapport de stage de fin d'études et donne mon accord pour son envoi au jury de soutenance.

7. Reconnaissance de la clause de non-plagiat

Je reconnais avoir pris connaissance du formulaire d'information sur le plagiat, contenu dans le présent rapport.

8. Accord de diffusion

Par suite de l'accord entre l'Université de Haute Alsace et la Bibliothèque de l'Université, après audition du stagiaire, le rapport sera confié à la bibliothèque qui en assurera le stockage et la valorisation.

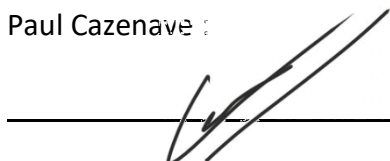
- ☒ Le contenu n'est pas confidentiel, j'autorise la diffusion immédiate du rapport*.
- ☐ Le contenu est confidentiel, j'autorise la diffusion du rapport après une période d'attente de : ☐ 1 an. ☐ 2 ans ☐ 3 ans ☐ 5 ans ☐ 10 ans*

Fait à Blagnac

Le 07/08/2023

Signature (**représentant de l'entreprise**) :

Paul Cazenave :



* Cocher la mention utile

Formulaire d'information sur le plagiat

Dans le règlement des examens validé par la CFVU du 2 octobre 2014, le plagiat est assimilé à une fraude.

Le plagiat consiste à reproduire un texte, une partie d'un texte, des données ou des images, toute production (texte ou image), ou à paraphraser un texte sans indiquer la provenance ou l'auteur. Le plagiat enfreint les règles de la déontologie universitaire et il constitue une fraude. Le plagiat constitue également une atteinte au droit d'auteur et à la propriété intellectuelle, susceptible d'être assimilé à un délit de contrefaçon.

En cas de plagiat dans un devoir, dossier, mémoire ou thèse, l'étudiant sera présenté à la section disciplinaire de l'université qui pourra prononcer des sanctions allant de l'avertissement à l'exclusion.

Dans le cas où le plagiat est aussi caractérisé comme étant une contrefaçon, d'éventuelles poursuites judiciaires pourront s'ajouter à la procédure disciplinaire.

Je soussigné(e) GROSSARD Mathieu

Etudiant(e) à l'Université de Haute Alsace en : 2022-2023

Niveau d'études : BAC+5

Formation ou parcours : Master Electronique, énergie électrique, automatique, Parcours Systèmes Embarqués

Reconnaît avoir pris connaissance du formulaire d'information sur le plagiat.

Fait à Toulouse

Le 05/05/2023

Signature : GROSSARD

Sommaire :

| | |
|--|----|
| Fiche archivage pour la bibliothèque universitaire | 2 |
| Approbations et diffusion par l'entreprise | 3 |
| Formulaire d'information sur le plagiat..... | 4 |
| Sommaire : | 5 |
| Présentation de l'entreprise..... | 8 |
| Introduction | 9 |
| Développement..... | 11 |
| 1. Environnement logiciel..... | 11 |
| 2. Traitement sur les données | 12 |
| 1. Lidar | 12 |
| 1. Etude bibliographique..... | 12 |
| 2. Traitement sur le Lidar | 12 |
| 2. Caméra | 13 |
| 1. Etude bibliographique..... | 14 |
| 2. Traitement sur la caméra | 15 |
| 3. Radar | 20 |
| 1. Etude bibliographique..... | 20 |
| 2. Traitement sur le Radar | 20 |
| 3. Référentiel | 21 |
| 1. TF tree | 21 |
| 2. Repère route | 22 |
| 4. Filtre de Kalman..... | 23 |
| 1. Etude bibliographique | 23 |
| 2. Kalman | 26 |
| 1. Association des sorties capteurs au filtre de Kalman | 27 |
| 2. Modalité d'implémentation du filtre de Kalman | 28 |
| 1. Implémentation du Filtre de Kalman : mise à jour unitaire des traitements capteurs | 29 |
| 2. Implémentation du Filtre de Kalman : mise à jour avec batch des traitements capteurs..... | 30 |
| 3. Comparaison des performances des deux méthodes..... | 32 |
| 3. Key performance Indicator (KPI) | 33 |
| 5. Application | 34 |

| | |
|-----------------------------------|----|
| 1. Système à trois capteurs | 34 |
| 2. Système à six capteurs | 35 |
| 3. Analyse..... | 37 |
| Conclusion et perspectives..... | 38 |
| Bibliographie | 39 |
| Résumé en français | 1 |
| Résumé en anglais..... | 1 |

Liste de figure :

| | |
|--|----|
| Figure 1: Réparation du Chiffre d'affaires par région, par métier et par secteur de Capgemini | 8 |
| Figure 2: Représentation du but du WP1 avec présentation de l'outil pour la simulation | 9 |
| Figure 3 : Représentation de la simulation avec affichage de la carte simulé, une voiture et un tramway, notre zone de digitalisation en jaune. | 10 |
| Figure 4 : Stack développée | 10 |
| Figure 5 : Affichage sur RVIZ des traitements sur les données brutes de Lidar (a), effectué dans l'ordre avec un filtrage (b) puis un clustering (c) et la projection au sol (c) | 13 |
| Figure 6 : Résultat de la détection par YOLO sur une image, avec la bounding box entourant les objets détectés en bleu..... | 14 |
| Figure 7 : Traitement sur la caméra (a) dans l'ordre en commençant par une égalisation d'histogramme (b) puis on utilise YOLO (c) pour ensuite projeter la bounding box au sol en 3D (d) | 15 |
| Figure 8 : Représentation 3D de la FOV_h et FOV_v et de H_{img} et W_{img} | 16 |
| Figure 9 : Représentation latérale du calcul de A_x pour le coin A à l'aide de l'angle alpha | 17 |
| Figure 10 : Représentation d'une image en sortie de YOLO avec en rouge la bounding box et ces coins respectifs. Ainsi que w_a et h_a qui vont nous permettre d'obtenir A_y et A_x respectivement. Estimation de l'angle alpha depuis la FOV_v et H_{img} avec h_a la distance entre le bas et le coin en bas gauche de la BB qui est représenté en rouge | 17 |
| Figure 11 : Représentation panoramique de A_y d'un véhicule depuis A_x et l'angle beta..... | 18 |
| Figure 12: Estimation de l'angle beta depuis la FOV_h et W_{img} avec h_a la distance entre le bas et le coin en bas gauche de la BB qui est représenté en rouge | 18 |
| Figure 13 : Estimation de la projection au sol de P_D et P_C depuis P_A et P_B ainsi que la connaissance de la longueur de la voiture et le rapport W_{bbox} et H_{bbox} | 19 |
| Figure 14 : Traitement sur le Radar (a) en passant par un filtrage spatial (b) pour finir par un clustering et une projection au sol (c) | 21 |
| Figure 15 : TF Tree d'un système à trois capteurs | 22 |
| Figure 16 : Transformation de Repère commun vers Repère route..... | 22 |
| Figure 17 : Affichage de la position estimé des véhicules par le filtre de Kalman | 27 |
| Figure 18 : Association du filtre de Kalman avec les données mesurées | 28 |
| Figure 19 : Séquentialité de la mise à jour | 29 |

| | |
|--|----|
| Figure 20: Réseau de Petri pour le filtre de Kalman en fonctionnement unitaire | 30 |
| Figure 21 : Threading mise à jour | 31 |
| Figure 22 : Réseau de Petri du filtre de Kalman avec un batch | 32 |
| Figure 23 : Système à trois capteurs (Caméra, Lidar, Radar) sur notre carte dans notre simulateur | 34 |
| Figure 24 : Mesure des KPI pour un système à trois capteurs. | 35 |
| Figure 25 : Système à six capteurs avec les trois capteurs (Lidar, Radar et Caméra) respectivement dans les rectangles rouges | 35 |
| Figure 26 : Complémentarité et redondance du système à six capteurs vus par le positionnement des deux Lidars et les données communes et supplémentaires | 36 |
| Figure 27 : Mesure des KPI pour un système à six capteurs..... | 36 |

Liste de Tableau :

| | |
|---------------------------------------|----|
| Tableau 1 : Index de nomination | 15 |
|---------------------------------------|----|

Présentation de l'entreprise

Ce stage a été effectué au sein de Capgemini Engineering, anciennement Altran Technology & Engineering Center avant d'être racheté par le groupe Capgemini en 2020. Capgemini est une entreprise spécialisée dans sept secteurs d'expertise, à savoir : les biens de consommation, le commerce, la distribution et le transport, les télécommunications, des médias et des technologies.

Capgemini est présent dans 50 pays, et accompagne plus de 600 acteurs industriels au travers de ses nombreux projets clients. En 2020, Capgemini totalise un chiffre d'affaires de 16 milliards d'euros. La répartition régionale, sectorielle et de métier est donné en Figure 1.

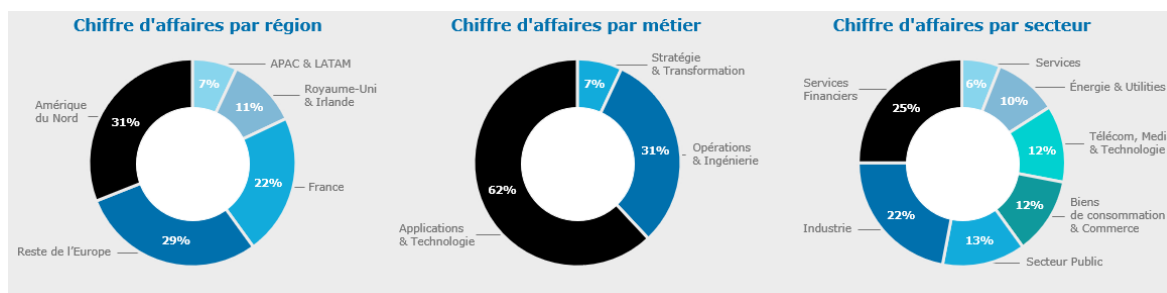


Figure 1: Répartition du Chiffre d'affaires par région, par métier et par secteur de Capgemini

Capgemini est une entreprise qui se divise en quatre marques principales. La première est Capgemini, qui est un leader mondial dans le domaine du conseil, des services informatiques et de la transformation numérique. La deuxième marque est Capgemini Invent, qui est la branche dédiée à l'innovation digitale, au conseil et à la transformation du groupe Capgemini. La troisième marque est Capgemini Engineering, qui aide les organisations innovantes à concevoir les produits et services de demain. La quatrième marque est Sogeti, pour créer des solutions personnalisées à partir de technologies existantes et émergentes.

Le site de Blagnac fait partie de Capgemini Engineering fournissant des services de conseil en ingénierie, de gestion de projet et de développement de logiciels. Le site emploie environ 500 personnes et dispose de laboratoires et d'installations de pointe pour soutenir ses activités.

Pendant mon stage, j'ai intégré l'équipe en charge du projet STRASS. Le projet STRASS est composé de cinq work packages (WPs) Le WP1, dirigé par Paul Cazenave (tuteur) a été l'objet de ce stage. Le WP2 a pour objectif la conception d'une navette de voie pour la manutention à l'aide d'un bras robotisé. Le WP3, est consacré au design de stratégie de supervision pour les flottes de tramway afin de garantir entre autres la sécurité de conduite, optimisation énergétique, desservissement optimisé. Le WP4, a pour but d'anticiper la perte d'adhérence due à la chute de feuille sur les voies ferrées, en utilisant un flux vidéo embarqué. Enfin, le WP5 avait pour objectif d'étudier les risques psychologiques induits par l'interaction avec des systèmes autonomes, avec une approche safety système.

Introduction

Dans un premier temps, nous allons présenter le contexte de l'étude. Mon stage se déroule dans le Work package 1 de Strass qui propose de développer de nouveaux systèmes de bord de voie pour une zone d'intersection rail-route. Le but du système est d'estimer les risques de collisions entre les tramways et les usagers dans des zones à risque. Ces systèmes permettront une connaissance en temps réel de l'environnement. Ce système peut être composé de différents capteurs tels qu'une caméra, un Lidar, un Radar....

Le but est d'effectuer une acquisition des données et identifier les conduites à risque dans une zone de digitalisation, qui correspond à une portion de la route, pour exploiter ces informations et agir sur le tramway. Le stage a pour objectif d'effectuer les traitements sur les capteurs et de réaliser une fusion à partir des données provenant des capteurs. La vitesse et la position des véhicules obtenus par la fusion serviront d'entrée à l'estimation des risques de collisions. Des protocoles de communication avec le tramway seront ensuite mis en place. La Figure 2 propose un résumé de la démarche.

Pour l'étude, il a été décidé d'utiliser un logiciel nommé Carla [1]. Il s'agit d'une couche logicielle de Unreal Engine, qui est un moteur graphique puissant qui permet le développement de jeux, de simulations et d'expériences en réalité virtuelle. Carla nous permet de simuler notamment du trafic et des capteurs. Afin de nous baser sur un scénario réaliste, une carte 3D d'un rond-point réel a préalablement été réalisée sous Unreal Engine, un moteur graphique. Nous utilisons le logiciel ROS [2] pour assurer la communication entre Carla et nos scripts d'analyses de données.



Figure 2: Représentation du but du WP1 avec présentation de l'outil pour la simulation

Après avoir étudié la question avant le début de ce stage, une étude a été effectuée sur l'accidentologie des tramways, cette étude a révélé que les intersections aux sein des tramways étaient les zones comportant un nombre d'accident importants et de gravité élevée. Cette étude justifie le développement du système OMNISCIENT. De manière à définir un cas d'étude représentatif d'une configuration, une carte modélisant le rond-point Jean-Mage situé à Blagnac a été développée sur le simulateur Carla [1]. Une vue de la carte est présentée sur la figure 3, avec l'illustration du concept de zone de digitalisation (coloriée en Jaune).

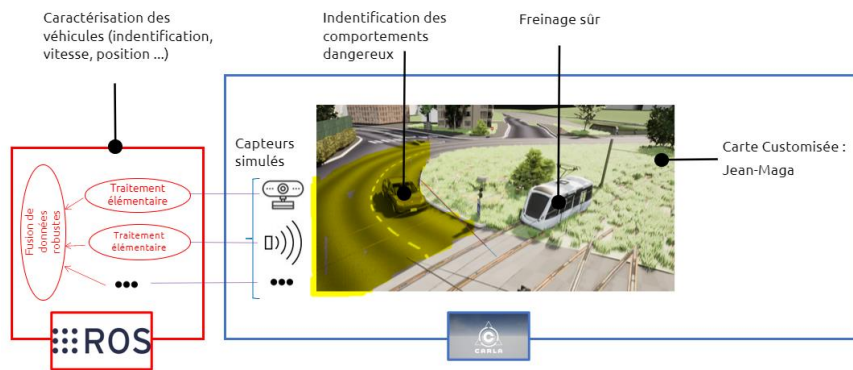


Figure 3 : Représentation de la simulation avec affichage de la carte simulé, une voiture et un tramway, notre zone de digitalisation en jaune.

Au début du stage, des travaux ont été préalablement réalisés, dont une première configuration de capteurs qui contient un Lidar et une Caméra. Ce stage se concentre sur la partie "acquisition des données", cf. Figure 2, et propose de développer une architecture logiciel assurant le suivi des véhicules sur la zone de digitalisation. Il était proposé d'atteindre cet objectif en mettant en œuvre une fusion multi-capteurs robuste, capable de résister à la perte ou au dysfonctionnement de certains capteurs intégrés au système.

La Figure 4 illustre l'architecture logiciel développée au cours de ce travail, qui constitue la base de l'élaboration de ce rapport. Dans un premier temps, nous présenterons l'Environnement logiciel avant de développer les Traitement sur les données appliqués sur les différents capteurs, le Lidar, la Caméra et le Radar, en décrivant les traitements spécifiques appliqués pour harmoniser leur format de données.

Par la suite, nous abordons la mise en place d'un Référentiel commun avec le fonctionnement du TF tree ainsi que la transformation nécessaire sur les données pour passer à un Repère route. Ensuite, nous nous concentrerons sur l'application du Filtre de Kalman et présentons deux méthodes développées pour la mise à jour. Finalement, nous présenterons l'Application avec deux configurations à un certain nombre de capteurs sur lesquelles nous allons comparer les mesures résultantes de la prédiction du filtre de Kalman.

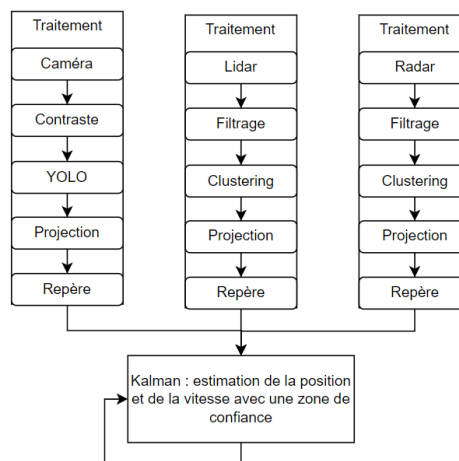


Figure 4 : Stack développée

Développement

1. Environnement logiciel

Une étude bibliographique a été nécessaire de manière à baser nos travaux sur les éléments de la littérature scientifique et documentation logiciel. Dans un premier temps nous présentons les logiciels les plus utilisés durant mon stage en commençant par Carla [1]. Carla Simulator est une plateforme de simulation open-source pour la recherche sur la conduite autonome. Elle permet une simulation détaillée et réaliste de divers scénarios de conduite en environnement urbain. Carla offre une API Python flexible qui permet d'interagir avec la simulation, de contrôler les véhicules autonomes, de définir les conditions météorologiques, et de manipuler les objets de la scène. Par ailleurs, Carla intègre un module nommé "ScenarioRunner" qui permet de définir et d'exécuter des scénarios de trafic spécifiques, facilitant ainsi la création de situations de conduite complexes pour tester et évaluer les systèmes de conduite autonome.

Nous avons ensuite travaillé avec le Robot Operating System (ROS) [2], une plateforme de développement open-source conçue pour la création de logiciels et de systèmes robotiques. Un élément central de ROS est le concept de "nœuds", qui sont essentiellement des processus indépendants exécutés simultanément et communiquant entre eux. Chaque nœud est chargé d'une tâche spécifique, par exemple, un nœud peut être responsable de la lecture des données d'un capteur, tandis qu'un autre peut effectuer le traitement des données. La communication entre ces nœuds est réalisée via des "topics", qui sont des canaux de communication nommés par lesquels les nœuds peuvent échanger des messages. Un nœud publie des messages sur un topic, et tous les autres nœuds qui sont abonnés à ce topic reçoivent ces messages.

En plus de ROS, nous avons utilisé ROS bridge [3], un composant logiciel qui facilite la communication entre ROS et Carla. Ce pont nous a permis d'intégrer les informations de la simulation de Carla dans le flux de données de ROS, nous permettant ainsi de manipuler ces informations à l'aide de l'architecture de nœuds et de topics de ROS. Les documentations correspondantes nous ont aidés à concevoir nos propres types de messages ROS et à comprendre comment récupérer et manipuler les informations de notre simulation.

Avant le démarrage de ce stage le logiciel Autoware [4] était utilisé pour effectuer des opérations basiques sur les données capturées par les capteurs afin de les préparer pour une utilisation ultérieure (nommés traitements dans la suite). Autoware est une plateforme logicielle open source dédiée à la conduite autonome, offrant des fonctionnalités de perception, de planification et de contrôle pour les véhicules autonomes. Il a été choisi au niveau du projet de se séparer d'Autoware en reprenant les idées au sein d'un code propre au projet.

2. Traitement sur les données

Notre objectif est de parvenir à une fusion des données de différents capteurs pour en tirer la prédiction de la position et la vitesse des données. Dans cette optique, nous avons commencé par effectuer des traitements spécifiques pour obtenir des données au format standardisé de ROS. Ce format nous permet d'afficher les données dans RVIZ, un outil de visualisation des données et traitements effectués dans ROS. Pour le Lidar et la caméra, nous avons opter pour une visualisation d'obtenir un polygone sur lequel nous extrayions le centre de masse. En revanche, pour le Radar, ayant l'information de la vitesse nous avons décidé d'obtenir directement le format sous le centre de masse.

1. Lidar

Le Lidar est un capteur qui utilise des impulsions laser pour mesurer la distance et la profondeur des objets environnants. Il émet des faisceaux laser qui sont réfléchit par les surfaces visées, la mesure du temps entre l'émission et la réception permet d'estimer la distance de l'objet rencontré. Les Lidars tridimensionnels, que nous utilisons dans notre système, captent des informations pour créer des nuages de points tridimensionnels. Ces derniers représentent la structure et la géométrie de l'environnement, rendant les Lidars particulièrement utiles pour la cartographie, la navigation autonome et la détection d'obstacles.

1. Etude bibliographique

Dans cette partie nous verrons l'étude bibliographique qui a été effectué pour réaliser les traitements sur le Lidar. Le filtrage spatial et le clustering euclidien des données provenant du Lidar sont effectuées à l'aide de la bibliothèque open source Point Cloud Library (PCL) [5] développée principalement en C++ pour le traitement des nuages de points 3D. Il a été choisi de produire en sortie des traitements élémentaires Lidar un format de donnée unique sous la forme d'un polygone dans le repère 2D correspondant à la projection au sol du nuage de point du Lidar correspondant au véhicule

2. Traitement sur le Lidar

La Figure 5 propose un exemple des différentes étapes de traitements appliqués aux données du Lidar, que nous allons détailler dans la suite, le clustering donne une couleur à la forme identifiée appartenant à un même groupe.

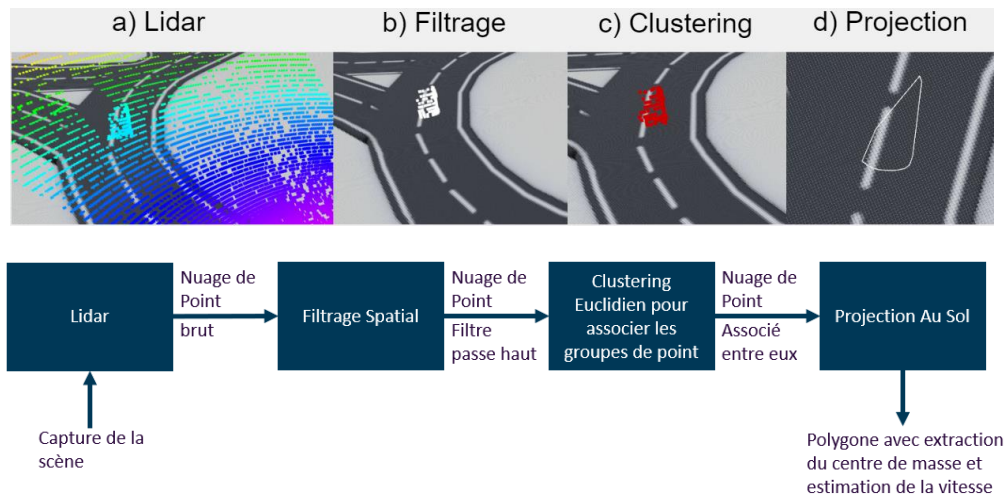


Figure 5 : Affichage sur RVIZ des traitements sur les données brutes de Lidar (a), effectué dans l'ordre avec un filtrage (b) puis un clustering (c) et la projection au sol (d)

Le Lidar observé ici est à 5m du sol. Pour la mise sous un même format des données, nous avons commencé par effectuer un filtrage spatial (cf. Figure 5b) sur les données du Lidar (cf. Figure 5a) pour ne conserver que les informations pertinentes dans la zone qui nous intéresse. Nous avons utilisé la bibliothèque PCL [5], qui offre des fonctions de filtrage et de clustering fonctionnels. Les paramètres pour le filtrage spatial x et z ont été choisis en fonction des caractéristiques du Lidar et de sa position par rapport à la zone d'étude.

Nous avons ensuite réalisé un clustering euclidien, avec la fonction `EuclideanClusterExtraction` de la bibliothèque PCL [5], pour identifier les groupes de points proches, correspondant à un véhicule (cf. Figure 5c). Nous avons utilisé les paramètres de taille minimale et maximale de cluster, ainsi que la tolérance en termes de mesure euclidienne pour adapter l'algorithme à notre situation. Nous commençons par extraire les clusters détectés pour ensuite utiliser la librairie PCL pour projeter les points du cluster sur un même axe horizontal (« projection au sol »). Juste après la projection au sol, nous effectuons une recherche du contour, à l'aide de PCL, du cluster qui a été projeté au sol pour n'avoir finalement que les points extérieurs. Cette représentation en polygone est ensuite utilisée pour la fusion, son centre de masse, et la visualisation dans RVIZ, cf. Figure 5.

2. Caméra

Une caméra, dans le contexte de notre système, produit des images sous forme de matrices de points où chaque point, représenté par un pixel, est défini par ses composantes rouge, vert et bleu (R, G, B).

1. Etude bibliographique

Dans cette partie nous verrons l'étude bibliographique qui a été effectué pour réaliser les traitements sur la caméra. Pour la caméra, nous avons choisi d'utiliser You Only Look Once (YOLO) qui est un algorithme de détection d'objets en temps réel basé sur l'apprentissage profond qui nous retourne des bounding box (BB) encadrant l'objet détecté et sa position dans l'image. La Figure 6 présente un résultat de la détection obtenue avec YOLO. En bleu sont représentés les BB encadrant les objets détectés (ici, les voitures). Chaque BB est associée à une classe et à une estimation de confiance dans la prédiction. Nous sommes passés de YOLOv3 dans la version d'Autoware à YOLOv8 dans la nôtre.



Figure 6 : Résultat de la détection par YOLO sur une image, avec la bounding box entourant les objets détectés en bleu

Un des enjeux de ce stage est de fusionner les données provenant des capteurs. Il a été choisi de produire en sortie des traitements élémentaires caméra un format de donnée unique sous la forme d'un polygone dans le repère 2D correspondant à la projection au sol de la BB caméra correspondant au véhicule. La projection au sol a été effectuée par calculs trigonométriques, avec une hypothèse forte sur la connaissance de la longueur du véhicule.

Nous avons consulté différents articles, tels que celui de M. Rezaei et. al [6], proposant une méthode pour détecter des objets en 3D sur une route complexe, en utilisant des BB en 3D. Bien que cette méthode soit intéressante, elle n'est pas la plus simple ni la plus rapide à implémenter, et nous avons préféré trouver une solution plus simple pour notre cas. Nous ne développerons donc pas davantage cet article pour notre travail, bien qu'il soit utile pour des références techniques et pour d'autres travaux.

L'article de A. Mousavian et. al [7] présente une méthode pour obtenir une BB en 3D autour des objets détectés en utilisant du Deep Learning et de la géométrie. Cet article est intéressant, mais n'est pas applicable à notre projet, car nous cherchons à limiter l'utilisation de techniques de machine learning, y compris de Deep learning, pour obtenir la BB en 3D. Les auteurs W. Ali et. Al [8] présentent une méthode permettant d'estimer avec de bons résultats une bounding box 3D à partir d'une unique caméra et de la connaissance a priori de la taille de l'objet. En plus de cela, les résultats annoncés de cette méthode amènent à des résultats très concluant. En considérant une taille générique pour chaque classe prédite par YOLO (camion, voiture, bus, etc.) et en estimant la direction prise par le véhicule, nous pouvons ainsi approximer la projection en profondeur d'un véhicule. Nous avons choisi de retenir cette méthode pour la suite de cette étude.

2. Traitement sur la caméra

| Notation | Description | Valeur | Notation | Description | Valeur |
|-----------|---|------------|--------------|--|--------|
| FOV_h | Angle de vue horizontal de la caméra, défini dans un fichier. Définit les limites angulaires d'acquisition de la caméra | 45° | a_i | Position dans le repère (i,j) en i du point A | pixel |
| FOV_v | Angle de vue vertical de la caméra, définit les limites angulaires d'acquisition de la caméra | ° | w_a | Distance sur l'axe i entre b_i et le centre de l'image | pixel |
| H_{cam} | Position de la caméra par rapport au sol, niveau de la route | 3.65m | a_y | Position dans le repère (x,y) en y du point A | m |
| H_{img} | Information donnée dans le fichier de création de la caméra avec la hauteur de l'image en pixel (Résolutions) | 256 pixels | H_{bbox} | Hauteur de la BB obtenue par YOLO | pixel |
| W_{img} | Information donnée dans le fichier de création de la caméra avec la largeur de l'image en pixel (Résolution) | 512 pixels | W_{bbox} | Largeur de la BB obtenue par YOLO | pixel |
| a_x | Position dans le repère (x,y) en x du point A | m | L_{obj} | Longueur connue de l'objet | m |
| h_a | Distance sur l'axe j entre b_j et H_{img} | pixel | T_{est} | Taille estimée de l'objet selon direction | m |
| a_j | Position dans le repère (i,j) en j du point A | pixel | A, B, C et D | Respectivement le coin gauche en bas, le coin droit en bas, le coin droit en haut et le coin gauche en haut de la BB | |

Tableau 1 : Index de nomination

La caméra est positionnée à 3.5 mètres en hauteur par rapport au niveau du trottoir sur un poteau de trafic et encore 15 cm pour le niveau de la route soit 3.65 mètres pour être au niveau du sol. Elle possède une Field of View Horizontal (FOV_h) de 45°. La Figure 7 propose un exemple des différentes étapes de traitements appliquées aux images de la caméra, que nous allons détailler dans la suite.

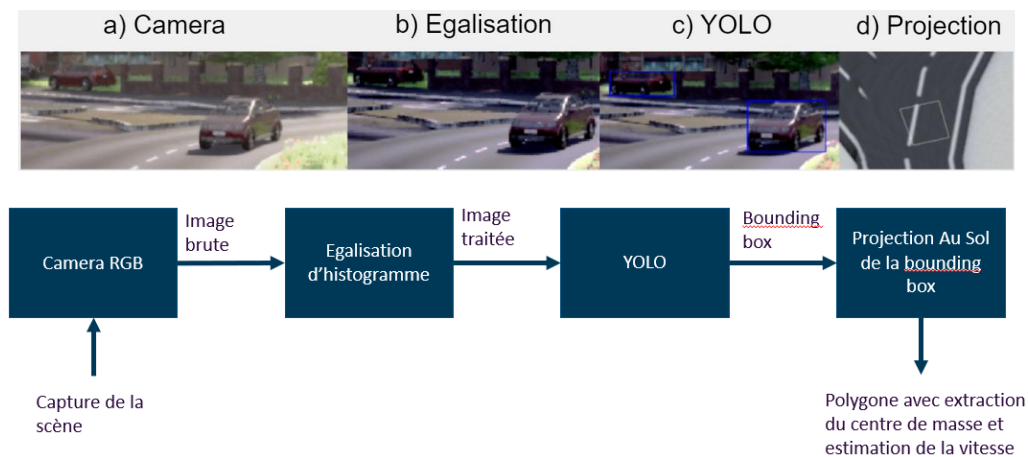


Figure 7 : Traitement sur la caméra (a) dans l'ordre en commençant par une égalisation d'histogramme (b) puis on utilise YOLO (c) pour ensuite projeter la bounding box au sol en 3D (d)

Nous commençons par appliquer une égalisation d'histogramme à l'image (cf. Figure 7b) afin de modifier son contraste et sa luminosité.

La méthode d'égalisation d'histogramme est une technique de traitement d'image permettant de répartir uniformément les niveaux de gris dans une image, améliorant ainsi le contraste global de l'image. Bien que l'égalisation ne soit pas la meilleure solution dans tous les cas, comme les cas extrêmes, par exemple un brouillard épais ou une nuit très sombre, elle offre une grande robustesse aux changements météorologiques et de luminosité.

Nous fournissons ensuite l'image obtenue à un réseau de neurones YOLOv8 déjà entraîné sur les classes fournies par le dataset COCO [9]. YOLO est un réseau de neurone pour la détection d'objets dans une image. Il permet de détecter les objets avec lesquels il a été entraîné. Pour chaque image, YOLO fournit une liste de BB (coordonnées représentant la position d'un objet), associée à une classe et un indice de confiance (cf. Figure 7c).

Pour l'étude, nous ne conservons que les BB dans les classes suivantes : voiture, un camion ou un bus. Pour convertir les données au format polygone de RVIZ, il est nécessaire de projeter au sol les BB obtenues en 2D (cf. Figure 7c) pour les passer en 3D (cf. Figure 7d). Pour cela, nous allons estimer la distance entre les coins de la BB et de la caméra dans le repère (x, y). La suite détaille la procédure suivie en l'appliquant à un seul des coins qui est le coin en bas à gauche nommé A. Les coordonnées du point A sont données par (A_i, A_j) dans le repère (i, j) et (A_x, A_y) dans le repère (x, y).

Il est important de noter que dans les paramètres de Carla pour la configuration d'une caméra, FOV_h, H_{img} et W_{img} sont définis, mais la valeur du champ de vision vertical (Field of View – FOV_v) ne l'est pas, ce qui motive l'importance de ce calcul à partir des informations de l'image.

Le champ de vision vertical représente l'angle de vue couvert par la caméra dans la direction verticale. Le schéma Figure 8 représente l'angle correspondant au champ de visions vertical et horizontal.

$$FOV_v = 2 * \operatorname{atan}\left(\frac{H_{img}}{W_{img}} * \tan\left(\frac{FOV_h}{2}\right)\right) \quad (1)$$

En remplaçant par les valeurs numériques, nous trouvons dans notre cas :

$$FOV_v = 2 * \operatorname{atan}\left(\frac{256}{512} * \tan\left(\frac{45}{2}\right)\right) = 23.4^\circ \quad (2)$$

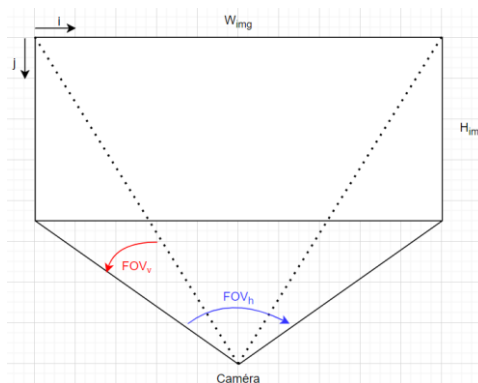


Figure 8 : Représentation 3D de la FOV_h et FOV_v et de H_{img} et W_{img}

La Figure 9 présente un schéma de la configuration. Nous définissons le repère orthogonal (x, y, z) où z correspond à la direction verticale et (x, y) sur le plan horizontal, tel que la caméra soit située au point (0, 0, 3.65) et que l'axe x pointe orthogonalement à la zone de digitalisation. Sachant que dans notre simulation, nous n'avons pas de distorsion, nous négligeons celle-ci dans les calculs. Soit A_x la position du coin gauche bas de la BB à la caméra selon l'axe x, qui représente une distance, ainsi que l'angle alpha tel qu'indiqué sur la Figure 9.

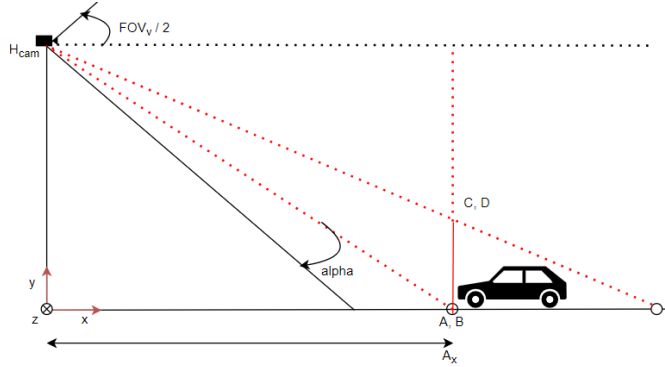


Figure 9 : Représentation latérale du calcul de A_x pour le coin A à l'aide de l'angle alpha

Ce qui fait qu'avec la position de la caméra, nous obtenons une vision en sortie de la caméra telle que la Figure 10.

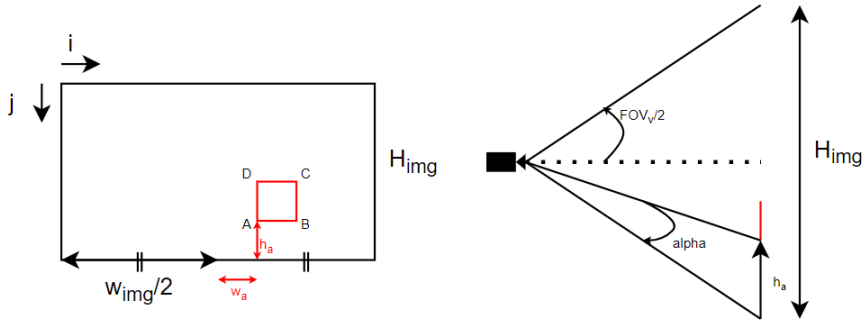


Figure 10 : Représentation d'une image en sortie de YOLO avec en rouge la bounding box et ces coins respectifs. Ainsi que w_a et h_a qui vont nous permettre d'obtenir A_y et A_x respectivement. Estimation de l'angle alpha depuis la FOV_v et H_{img} avec h_a la distance entre le bas et le coin en bas gauche de la BB qui est représenté en rouge

Ainsi, nous pouvons obtenir la valeur de h_a comme suit :

$$h_a = H_{img} - A_j \quad (3)$$

A partir de la distance h_a , nous pouvons déterminer l'angle alpha de la Figure 9 depuis la connaissance de la FOV_v et de H_{img} comme nous le voyons sur la Figure 10.

$$\frac{h_a}{\alpha} = \frac{H_{img}}{FOV_v} \Rightarrow \alpha = \frac{FOV_v * h_a}{H_{img}} \quad (4)$$

En appliquant un calcul de trigonométrie à partir de la Figure 9, on a :

$$A_x = \frac{H_{cam}}{\tan\left(\frac{FOV_v}{2} - \alpha\right)} \quad (5)$$

Avec alpha donné par la formule (4) et H_{cam} et FOV_v connus.

On cherche maintenant à calculer la distance A_y . Pour illustrer notre démarche, nous utilisons la Figure 11, sur laquelle nous cherchons à trouver la position cherchée A_y en trouvant dans un premier temps l'angle beta.

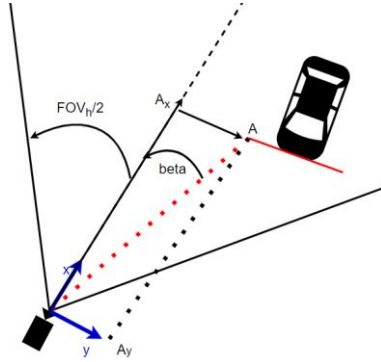


Figure 11 : Représentation panoramique de A_y d'un véhicule depuis A_x et l'angle beta

Nous supposons également que la moitié de la FOV_h , voir Figure 11, correspond à une valeur de y égale à 0m. Par conséquent, nous aurons à la fois des valeurs positives et négatives pour A_y et plus précisément pour w_a sur la Figure 10. Ainsi, nous cherchons à obtenir dans un premier temps w_a :

$$w_a = A_i - \frac{W_{img}}{2} \quad (6)$$

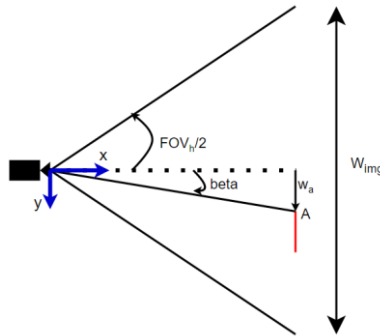


Figure 12: Estimation de l'angle beta depuis la FOV_h et W_{img} avec h_a la distance entre le bas et le coin en bas gauche de la BB qui est représenté en rouge

A partir de la Figure 12, on peut déduire :

$$\frac{\frac{FOV_h}{2}}{\beta} = \frac{\frac{W_{img}}{2}}{w_a} \Rightarrow \beta = \frac{\frac{FOV_h}{2} * w_a}{\frac{W_{img}}{2}} \quad (7)$$

Finalement, nous appliquons de la trigonométrie avec la tangente en connaissant son angle beta et la position A_x pour estimer la position A_y comme sur la Figure 11.

$$A_y = A_x * \tan(\beta) \quad (8)$$

Notons qu'une légère erreur dans la position en x du véhicule détecté peut arriver. Cela peut être attribué à des facteurs tels que la distorsion de la caméra lorsque nous passons à un système réel ou des approximations dans le calcul, de la position. Mais aussi, avec la détection par YOLO selon la précision de la BB et sa position de détection.

Afin de créer un polygone en 2D représentant la BB entourant le véhicule détecté, nous appliquons les formules à tous les coins de la BB pour obtenir leur projection que nous nommons P_A , P_B , P_C et P_D que nous voyons sur la Figure 13. Cependant, lors du calcul pour la projection de P_C et P_D , nous remarquons une distorsion pouvant être importante lors de la formation du polygone. Pour corriger l'erreur de position de P_C et P_D , sur la taille du véhicule, nous nous sommes appuyés sur une étude bibliographique [8] qui propose une solution appropriée en estimant la position de ces deux points. On nomme P_{Dest} et P_{Cest} la position des deux points estimés. Pour cela, nous nous restreignons à l'étude de la classe voiture et nous émettons l'hypothèse qu'une voiture mesure 4m de long (L_{obj}). Nous supposons aussi que lorsque le véhicule est perçu de face, le ratio $\frac{W_{bbox}}{H_{bbox}}$ est proche de 1. À partir de cette approximation et des tailles de BB estimées par YOLO, nous estimons l'orientation du véhicule de la manière de la Figure 13.

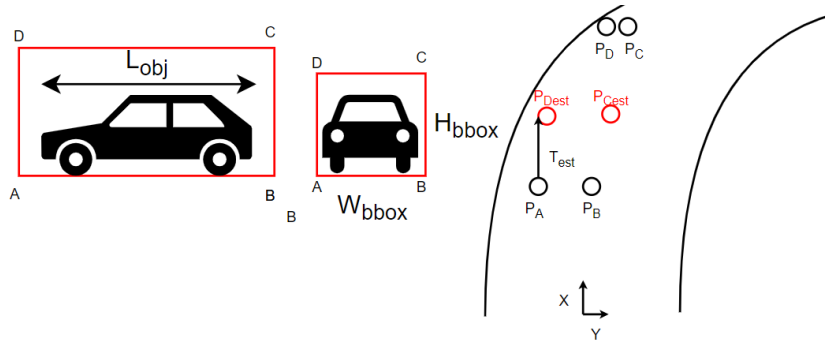


Figure 13 : Estimation de la projection au sol de P_D et P_C depuis P_A et P_B ainsi que la connaissance de la longueur de la voiture et le rapport W_{bbox} et H_{bbox}

Ainsi, en ayant une connaissance sur la longueur de l'objet (L_{obj}) avec les informations de la hauteur et la largeur de la BB, nous pouvons estimer une longueur T_{est} , qui est une approximation de la profondeur du véhicule selon l'axe x, ajoutée pour modéliser « l'orientation » du véhicule puisque la distance aura un impact sur la position en y :

$$T_{est} = L_{obj} * \frac{H_{bbox}}{W_{bbox}} \quad (9)$$

Ainsi, nous pouvons estimer par exemple la position de P_{Dest_x} depuis P_{A_x} et T_{est} .

$$P_{Dest_x} = P_{A_x} + T_{est} \quad (10)$$

Ensuite, il ne nous reste plus qu'à calculer les coordonnées en y, en appliquant la formule précédente (cf. formule 8). Grâce à ces formules, nous avons réussi à estimer des tailles de polygones, $P=(P_A, P_B, P_{Dest}, P_{Cest})$, qui se rapprochent des dimensions réelles de nos véhicules. Un exemple de ces polygones est donné en Figure 7d.

3. Radar

Un Radar Doppler est un capteur utilisé pour mesurer la vitesse relative des objets se déplaçant par rapport au capteur lui-même. Il fonctionne en émettant des ondes électromagnétiques, généralement des ondes radio ou micro-ondes, et en mesurant le décalage de fréquence des ondes réfléchies lorsqu'elles sont renvoyées par les objets en mouvement. Ce décalage de fréquence est directement proportionnel à la vitesse relative de l'objet.

1. Etude bibliographique

Dans cette partie nous verrons l'étude bibliographique qui a été effectuée pour réaliser les traitements sur le Radar. Le filtrage spatial et le clustering euclidien des données provenant du Radar sont effectuées à l'aide de la bibliothèque open source Point Cloud Library (PCL) [5] développée principalement en C++ pour le traitement des nuages de points 3D.

Il a été choisi de produire en sortie des traitements élémentaires du Radar un format de donnée unique sous la forme d'un centre de masse dans le repère 2D correspondant à la projection au sol du nuage de point du Radar correspondant au véhicule avec la moyenne résultante de la vitesse.

2. Traitement sur le Radar

Nous avons exprimé le souhait d'harmoniser le format des données provenant de nos différents capteurs afin de faciliter leur fusion. Pour le Radar, il a été choisi que le traitement des données ne fournisse pas un polygone qui apporte uniquement des informations sur la position du véhicule, mais retourne de l'information sur la position du véhicule mais aussi sur sa vitesse. La raison en est que tous les points du nuage de point contiennent une position en (x, y, z) associé avec une vitesse. Lors de l'étape de traitement élémentaire, nous avons appliqué des techniques similaires à celles utilisées pour le Lidar, avec quelques adaptations visant à extraire la vitesse.

Contrairement à une représentation sous forme de polygone, nous avons choisi de représenter directement les données Radar par un point unique, le centre de masse du véhicule, incluant l'information de vitesse. Cette décision nous permet de simplifier la représentation des données Radar tout en conservant les informations essentielles requises pour la fusion et le suivi des objets détectés. Cette approche a démontré son efficacité dans notre système et contribue à améliorer la précision de nos estimations de position et de vitesse des véhicules, sachant que pour fusionner, nous utilisons le centre de masse des polygones du Lidar et de la caméra avec le point du Radar.

La figure 14 propose un exemple des différentes étapes de traitements appliquées aux données du Radar, que nous allons détailler dans la suite.

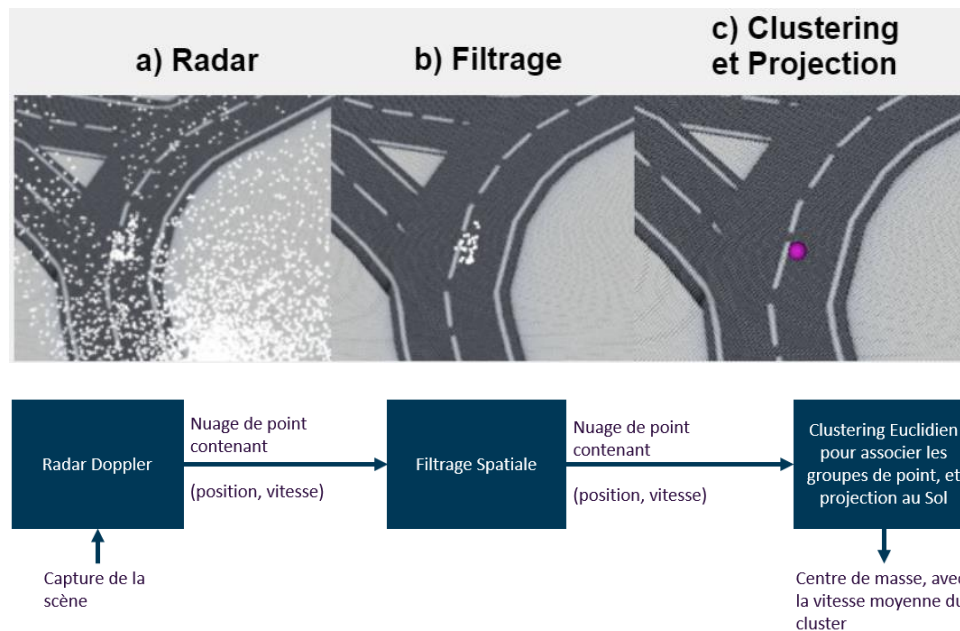


Figure 14 : Traitement sur le Radar (a) en passant par un filtrage spatial (b) pour finir par un clustering et une projection au sol (c)

Lors de nos traitements des données Radar, nous avons rencontré certaines difficultés liées au format des données fournies par le Radar dans le cadre de notre système ROS. En effet, les données étaient initialement présentées sous la forme d'un nuage de points comprenant des informations supplémentaires par rapport au Lidar puisque les données étaient sous le format (x, y, z) avec la vitesse, l'angle d'élévation, l'angle azimut et la distance. Afin de pouvoir traiter les données sous ROS, nous avons donc d'abord utilisé la bibliothèque PCL pour convertir cette représentation dans un repère orthogonal (x,y,z). Nous avons effectué un filtrage spatial sur les données Radar et créé un nouveau nuage de points qui conserve la position des points filtrés, tout en récupérant l'information de la vitesse pour chaque point (cf. Figure 14b). Puis, nous avons appliqué un processus de clustering euclidien similaire à celui utilisé pour le Lidar afin d'extraire le nombre de véhicules détectés tout en conservant les informations supplémentaires du Radar. En adaptant le clustering, nous avons récupéré les centres des clusters et les avons associés aux informations fournies par le Radar en effectuant une moyenne sur tous les points constituant le cluster, pour obtenir la vitesse moyenne du cluster. Nous avons également projeté les positions au sol en fixant la valeur de l'axe Z. En conséquence, nous avons publié un nouveau message contenant un point représentant la position détectée par le Radar (cf. Figure 14c), ainsi que les informations de vitesse. L'étape suivante est de mettre les données dans un référentiel commun.

3. Référentiel

1. TF tree

Notre système est composé de trois capteurs, chacun fournissant des données dans leur propre référentiel. Bien qu'il soit possible de fusionner les données provenant de référentiels distincts en appliquant les transformations appropriées.

Il est préférable d'avoir les données de capteurs dans un référentiel commun avant la fusion afin d'assurer une cohérence et une compatibilité entre les informations provenant de différents capteurs. Cela permet d'éviter les erreurs et les incohérences lors de la fusion des données, garantissant ainsi des estimations plus précises et fiables.

Nous utilisons les outils CARLA et ROS2 pour interagir avec nos capteurs. Grâce à ROS, nous pouvons utiliser une fonctionnalité appelée TF (Transform) pour appliquer facilement des transformations entre référentiels.

La structure TF Tree, Figure 15, dans ROS, est un arbre qui représente les relations de transformation entre les différents cadres de référence dans un système robotique. Nous avons défini trois référentiels distincts dans notre système. Tout d'abord, nous avons le référentiel du poteau, nommé référentiel commun dans la suite. Ensuite, nous avons le référentiel du Lidar et le référentiel de la caméra connecté comme la Figure 15.

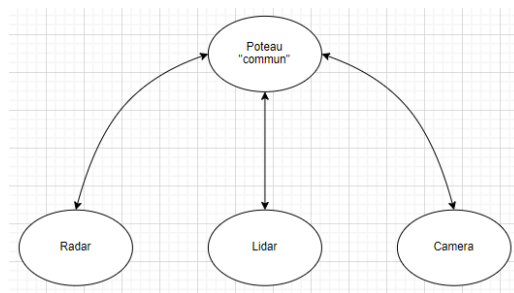


Figure 15 : TF Tree d'un système à trois capteurs

2. Repère route

Ce que nous souhaitons faire ensuite est de transformer les données du référentiel commun, qui représente un trajet en courbe, vers un repère route qui correspond à une trajectoire en ligne droite de la manière de la Figure 16. L'objectif de cette approche est d'étudier, à terme, uniquement la distance en x et la vitesse entre le véhicule détecté et l'origine du repère. Cette analyse simplifiée permettra d'évaluer les risques de collision et de déterminer les interventions appropriées sur le tramway. Il faut bien prendre en compte, qu'à la fin, nous ne changeons pas de référentiel, mais nous appliquons une transformation à toutes les données qui est ensuite utilisé lors de la fusion entre les données.

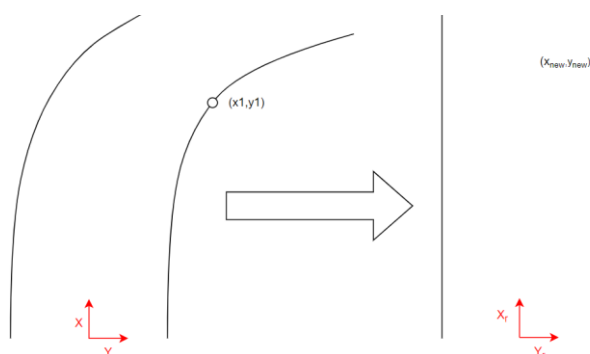


Figure 16 : Transformation de Repère commun vers Repère route

Nous avons pris la décision de créer une représentation de la route en utilisant une équation polynomiale afin de filtrer les véhicules qui se trouvent en dehors de notre zone de digitalisation et de mieux exploiter les informations de position et de déplacement sur l'axe des x. Dans une première version, nous avons estimé la courbure en utilisant les déplacements des véhicules. Cependant, il est important de noter que cette équation n'est pas encore totalement précise et qu'il reste des améliorations à apporter, notamment en obtenant des informations sur la route à partir de capteurs ou d'autres techniques internes.

Repère cartésien $\mathbb{R}_c = (x, y)$ et repère route noté $\mathbb{R}_r = (x_r, y_r)$

On note la fonction de transformation de repère $\mathcal{F}_{\mathbb{R}_c \rightarrow \mathbb{R}_r}$ avec selon l'axe y :

$$\mathcal{F}_{\mathbb{R}_c \rightarrow \mathbb{R}_r}(x, y) = (x - (y - y_{start}) * \sin(\alpha), (y - y_{start}) * \cos(\alpha))$$

Avec $\alpha = \text{atan}\left(\frac{y}{x}\right)$ et $y_{start} = a * x^2 + b * x + c$

Ce qui nous permet d'obtenir finalement une projection des données dans le référentiel commun et avec l'application d'une transformation des données pour les représenter sur une route en ligne droite. La prochaine étape, consiste à trouver une méthode permettant d'estimer la courbure de la route de manière automatisée, plutôt que de la créer manuellement. Cette estimation de courbure sera basée sur les coordonnées obtenues à partir des capteurs ou d'une connaissance exacte de la route. Maintenant, nous allons nous intéresser au filtre de Kalman.

4. Filtre de Kalman

1. Etude bibliographique

Une étude bibliographique a été nécessaire de manière à baser nos travaux sur les éléments de la littérature scientifique et documentation logiciel. Dans cette partie, sera présenté brièvement les documents utilisés pour le développement du filtre de Kalman. Il a été imposé dans le cadre de ce stage l'utilisation d'un filtre de Kalman pour réaliser la fusion des données. J.Z. Sasiadek et P. Hartana [10] présente une utilisation du filtre de Kalman pour une application de fusion de différents capteurs. L'article s'intéresse également dans un scénario avec une trajectoire en courbe. Pour cela, ils ont fait les choix d'utiliser un Extended Kalman Filter (EKF) qui devrait être la suite du développement du projet. La perte de capteur et donc de sa robustesse lors de manque d'information ne font pas parti des aspects traités par l'article.

L'article de Chingiz Hajiyeu, Halil Ersin Soke [11] présente une étude sur un filtre de Kalman Robuste pour l'estimation de la dynamique d'un drone. Cet article est connexe à notre étude car il propose d'adapter le filtre de Kalman en cas de perte de capteurs ou d'informations erronées. Les auteurs ont modifié les matrices Q et R pour adapter le filtre de Kalman dans leur étude, ce qui pourrait être une piste intéressante à explorer dans notre propre travail.

Dans leur étude, Muhammad Adeel Akramet. Al [12], proposent une méthode pour adapter la matrice H du filtre de Kalman afin de prendre en compte les pertes de capteurs, tout en conservant la convergence et la stabilité du filtre. Ils présentent également plusieurs variantes du filtre de Kalman, ainsi que les résultats obtenus avec ces méthodes. Cette étude est particulièrement intéressante pour notre projet, car elle nous permettrait d'adapter le filtre de Kalman en modifiant seulement la matrice H, plutôt que de devoir recalculer les matrices Q et R à chaque perte de capteur. Cette approche est préférable puisque notre configuration est minimale et donc plus simple à adapter.

Enfin, le livre de Dan Simon [13] est une ressource très complète sur le filtre de Kalman, qui fournit de nombreuses informations utiles pour notre projet.

Dans une première version, nous avons opté pour le suivi mono-capteur des données traitées avant d'effectuer la fusion des capteurs avec le filtre de Kalman. Cette méthode n'utilisait pas la sortie du filtre de Kalman. Cette décision a été motivée par plusieurs raisons :

- Cette méthode permet de réduire des erreurs potentielles introduites par la fusion.
- Nous possédons une meilleure prise en compte des caractéristiques spécifiques de chaque capteur.
- Cela nous permet de réduire la complexité des calculs, étant donc plus avantageux pour une application temps réel.
- Cela nous permet d'avoir une visualisation plus fluide des données sous RVIZ.

Pour effectuer le tracking avant la fusion, nous avons choisi d'utiliser l'algorithme nommé « The centroid tracking algorithm » développé par Adrian Rosebrock [14]. Il s'agit d'une méthode de suivi d'objets en temps réel qui consiste à assigner un identifiant unique à chaque objet détecté dans une séquence d'images.

L'avantage de l'algorithme de suivi de centroïdes est qu'il est rapide et facile à implémenter, ce qui en fait une méthode populaire pour le suivi d'objets en temps réel. Il ne dépend pas de la forme spécifique de l'objet, car il se base uniquement sur les centroïdes. Ainsi, il peut être utilisé avec des BB ou des polygones ou le centre de masse pour obtenir les centroïdes.

L'idée de départ est de calculer le centre de chaque objet détecté et de l'associer à l'objet correspondant dans l'image suivante en utilisant un critère de distance. L'algorithme commence par détecter tous les objets dans la première image de la séquence en utilisant un détecteur d'objets tel que YOLO ou Faster R-CNN. Ensuite, pour chaque objet détecté, l'algorithme calcule le centre de masse ou centroïde en utilisant la formule suivante :

$$x_{centroid} = \frac{(x_{min} + x_{max})}{2} \quad (11)$$

$$y_{centroid} = \frac{(y_{min} + y_{max})}{2} \quad (12)$$

Où (x_{min}, y_{min}) et (x_{max}, y_{max}) sont les coordonnées du coin supérieur gauche et inférieur droit de la BB de l'objet détecté.

Pour nous, nous remplaçons la recherche de YOLO par les polygones projetés au sol pour la caméra et le Lidar et le centre de masse du Radar. Une fois que les centroïdes ont été calculés pour tous les objets de la première image, l'algorithme procède à l'association de chaque centroïde à un identifiant unique en utilisant une mesure de distance euclidienne. Dans l'image suivante, l'algorithme calcule les centroïdes correspondant et recherche l'objet le plus proche dans l'image précédente. Si la distance entre les deux est inférieure à un seuil prédéfini, l'algorithme considère que l'objet est le même que l'objet précédent et met à jour le centroïde et l'identifiant de l'objet.

Cependant, cet algorithme peut être sensible aux occlusions et aux changements brusques de direction des objets, ce qui peut entraîner des erreurs de suivi. Il est important de choisir un seuil approprié pour la distance euclidienne afin de maintenir une correspondance précise entre les objets d'une image à l'autre. Pour fusionner ces données, nous attribuons initialement un identifiant à chaque donnée. Ensuite, lors de la fusion, nous récupérons le centre de masse des polygones, nous vérifions si les données des capteurs peuvent être associées en calculant la distance euclidienne minimale entre les centres de masse obtenue des capteurs et en vérifiant si elle est inférieure à une distance d'association prédéfinie.

Nous créons ensuite notre filtre de Kalman en utilisant les données associées par la distance euclidienne et en leur attribuant les identifiants correspondants. Cela nous permet de gagner du temps lors des itérations ultérieures du filtre de Kalman, car le suivi a déjà été effectué et nous pouvons mettre à jour le filtre avec les nouvelles données d'entrée associées aux identifiants.

Il est important de prendre en compte le cas où un identifiant n'a pas été mis à jour depuis un certain temps. Pour cela, on incrémente une valeur du filtre de Kalman à chaque prédiction. Si le Kalman peut être associé lors d'une mise à jour, la valeur passe à 0. Si on obtient une valeur limite telle que 25, choisi selon le nombre d'appels à la prédiction considérée comme suffisant pour perdre l'information sur notre zone de digitalisation, on considère que le capteur a perdu cette donnée et il peut être associé à une donnée plus récente.

Après une étude et des essais, nous avons décidé de ne pas effectuer de tracking mono-capteur avant la fusion des données. Puisque nous effectuons une fusion en utilisant une distance euclidienne entre les identifiants, pouvant provoquer une erreur d'association selon les cas et ne pouvant pas mettre à jour correctement. Le fait est qu'en utilisant cette méthode, nous ne profitons pas de la puissance du filtre de Kalman au maximum.

L'autre raison, en faisant du tracking avant la fusion, nous pouvions donner plus d'importance à un capteur par rapport à un autre, mais ce n'est pas ce qui a été souhaité pour réaliser notre fusion. Le problème qui a été posé est que nous ne souhaitons pas avoir de pondération d'un des types de capteur trop important qui occulterais les résultats d'autres capteurs. Pour ces raisons, bien que nous l'ayons initialement implémenté et utilisé, nous avons finalement choisi d'opter pour l'utilisation du filtre de Kalman pour le suivi. Nous exploitons ainsi son état prédit ainsi que la matrice de covariance pour associer les données issues des différents capteurs. C'est pourquoi nous avons opté pour la version du filtre de Kalman qui sera présentée par la suite.

2. Kalman

Nous avons étudié la méthode de fusion de données pour notre système. Nous avons choisi d'utiliser la première version du filtre de Kalman, qui est un filtre de Kalman discret et qui suit une loi d'évolution linéaire. Cependant, nous avons dû adapter le filtre de Kalman pour notre cas spécifique, car notre système ne dispose pas de commande en entrée et nous ignorons le bruit d'évolution. Le filtre de Kalman suppose que le processus discret réel x_k suit la loi d'évolution linéaire suivante :

$$x_k = A_k x_{k-1} \quad (13)$$

Dans laquelle A_k est la matrice d'évolution, ou matrice de transition, reliant l'état précédant $k-1$ à l'état actuel k et modélisant le mouvement de l'objet. À chaque instant k , le processus est observé par des mesures z_k s'exprimant comme suit :

$$z_k = H_k x_k + v_k \quad (14)$$

Avec H_k qui est la matrice d'observation et v_k le bruit de mesure, gaussien centré et de matrice de covariance R_k .

Ensuite, il reste encore à présenter les deux méthodes qui sont la prédiction qui s'active à un pas de temps fixe pour prédire l'état et la position probable du véhicule. La seconde, est la mise à jour qui s'active uniquement quand les données d'un capteur sont obtenues.

La prédiction fonctionne de telle manière :

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} \text{ (État prédit)} \quad (15)$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k \text{ (Estimation prédite de la covariance)} \quad (16)$$

Dans laquelle $P_{k-1|k-1}$ est la matrice de covariance a posteriori estimée à l'état précédent $k-1$ et la matrice de covariance du bruit d'évolution Q_k .

La mise à jour est décrite par les équations suivantes :

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} \text{ (Innovation)} \quad (17)$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \text{ (Covariance de l'innovation)} \quad (18)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \text{ (Gain de Kalman)} \quad (19)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \text{ (Etat mis à jour)} \quad (20)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \text{ (Covariance mise à jour)} \quad (21)$$

Dans laquelle z_k est l'observation ou mesure du processus à l'instant k et R_k est la matrice de covariance du bruit de mesure.

Le vecteur d'état contient les coordonnées dans le plan horizontal de la position (cf. Figure 17, les points rouges représentent les positions des véhicules détectés), ainsi que les vitesses du véhicule, car notre objectif est de prédire la position et la vitesse futures. Le filtre de Kalman nous permet d'estimer et de mettre à jour ces valeurs en utilisant les mesures provenant de nos capteurs, ce qui nous permet d'obtenir l'information de la position et de la vitesse.

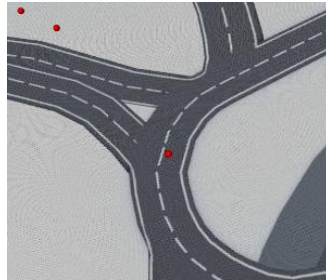


Figure 17 : Affichage de la position estimée des véhicules par le filtre de Kalman

Ainsi, nous avons conçu notre vecteur d'état x_k pour contenir les positions x et y , ainsi que leurs vitesses associées, ce qui donne $x_k = [x \ y \ \dot{x} \ \dot{y}]$. À la première instance du filtre de Kalman, ce vecteur est rempli avec les valeurs du centroïde associé. Nous avons modélisé le mouvement du véhicule en utilisant la matrice de prédiction A_k , conçue pour représenter un mouvement

rectiligne à vitesse constante (d'où l'importance du repère route) soit $A_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

1. Association des sorties capteurs au filtre de Kalman

Dans le contexte de la fusion de données, le filtre de Kalman peut être utilisé pour prédire l'état d'un objet, puisque pour n objet nous aurons n filtre de Kalman associés, détectés grâce au traitement élémentaire de chaque capteur qui fournissent pour la Caméra et le Lidar un polygone duquel on extrait son centre de masse et lui déterminons une vitesse avec une distance parcourue et le temps écoulé, quant au Radar lui nous fournit directement le centre de masse avec la vitesse associée. Le centre de masse et la vitesse est fourni dans le vecteur z_k , qui les intègre dans son modèle de mise à jour pour calculer (équ.17) et produit une estimation de l'état courant (équ.20).

La Figure 18, nous présente le fonctionnement de cette association que nous détaillons pour la compréhension. Nous avons donc utilisé la matrice de covariance P_k et l'état du filtre x_k de Kalman pour associer les données provenant des différents capteurs (centre de masse). L'état du filtre de Kalman nous donne l'information de position de notre véhicule prédit et la confiance de la prédiction à travers la matrice de covariance P_k . Cette approche présente plusieurs avantages. Tout d'abord, le filtre de Kalman permet de prendre en compte les estimations et les incertitudes de chaque capteur à l'aide de la matrice R_k . La matrice est de taille 4×4 initialisé à 0 et construite sur la diagonale pour accélérer l'étude car on considère que les erreurs de mesure de différentes variables sont supposées être non corrélées. Les bonnes valeurs dans cette matrice permettent d'améliorer la précision de la fusion des données.

De plus, en utilisant l'état prédit (équ.15) du filtre de Kalman, nous pouvons effectuer une association plus robuste des données entre les capteurs avec Q_k construit de la même manière que R_k mais avec des valeurs différentes et représentant les incertitudes sur le modèle, en prenant en compte les trajectoires et les mouvements des objets avec la modélisation du mouvement avec la matrice A_k .

Lors de la réception du premier jeu de données, aucun Kalman n'existe. Le premier filtre de Kalman est généré à la réception des premières données, on s'assure dans un premier temps que les données font références à un même véhicule (distance euclidienne entre les points faible). Puis l'état du Kalman est fixé à par rapport aux valeur des données. Lorsqu'un nouvel objet est détecté par un capteur, le filtre de Kalman utilise son état prédit ainsi que la matrice de covariance pour estimer la probabilité que cet objet corresponde à l'un des objets déjà suivis. Cette probabilité est calculée en comparant la distance entre les prédictions du filtre de Kalman et les mesures réelles du nouvel objet. Si la distance est inférieure à un seuil prédéfini, l'algorithme considère que le nouvel objet est le même que l'objet précédent et met à jour l'estimation de son état en utilisant les nouvelles mesures. Sinon, l'algorithme considère que le nouvel objet est un objet distinct et lui attribue un nouvel identifiant.

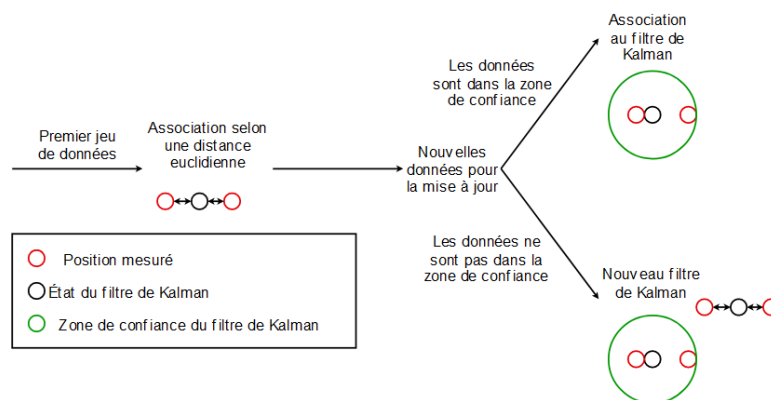


Figure 18 : Association du filtre de Kalman avec les données mesurées

2. Modalité d'implémentation du filtre de Kalman

Les réseaux de Petri, cf. Murata [15], sont un formalisme d'expression de systèmes à événements discret, nous avons fait le choix d'utiliser ce formalisme pour exprimer deux implémentations différentes du filtre de Kalman. Les réseaux de Petri sont des graphes composés de deux types de nœuds : les places (cercles) qui contiennent l'information de l'état du système. La configuration des jetons dans les places constitue sur l'état courant du système. Les transitions (carré) qui modélisent les possibilités d'évolution du système.

Une transition est dite sensibilisée si les places prédécesseurs de cette transition comportent au moins autant de jetons que son pondéré les arcs. Les arcs inhibiteurs empêchent la sensibilisation d'une transition si la place à laquelle il est lié comporte des jetons. Une transition sensibilisée peut être tirée, le tir d'une transition prélève dans les places prédécesseurs à la transition le nombre de jetons indiqué sur les arcs et génère dans les places successeurs le nombre de jetons indiqué sur les arcs.

1. Implémentation du Filtre de Kalman : mise à jour unitaire des traitements capteurs

Dans une première approche de l'implémentation, nous avons développé le filtre de Kalman en utilisant les fonctionnalités de base de ROS2. Dans ROS2, un des mécanismes d'appel de fonction se fait par "callback", c'est-à-dire que l'appel d'une fonction se fait par observation du topic, la fonction est appelée uniquement lors de l'arrivée de nouvelles données dans le topic. Cela signifie que les rappels sont activés les uns après les autres, même si des données provenant de plusieurs capteurs peuvent arriver simultanément.

Cette approche présente certaines hypothèses implicites sur son fonctionnement que nous avons déduit. Tout d'abord, elle suppose que nous effectuons plusieurs calculs de mise à jour pour des capteurs ayant des données temporelles presque ou exactement identiques. Cela peut entraîner un accès excessif à l'état du véhicule pour une mise à jour dans la même temporalité, ce qui risque de créer des petites perturbations et de rendre la mise à jour moins stable. D'autre part, cette approche séquentielle peut introduire des erreurs temporelles.

La Figure 19 illustre les résultats de cette analyse, en représentant spatialement les données du problème. Les données en noir représentent les mesures provenant des capteurs, la donnée en vert correspond à la prédiction du filtre de Kalman, et les données en rouge représentent les mises à jour du filtre de Kalman. Chaque chiffre indique l'ordre d'apparition ou l'étape dans le code. Dans cet exemple, nous commençons par la prédiction du filtre de Kalman, puis nous effectuons une première mise à jour en utilisant la première donnée 1, ce qui donne la position marquée en rouge avec le chiffre 1, suivie d'une mise à jour supplémentaire en utilisant la deuxième donnée 2, ce qui donne la position en rouge marquée avec le chiffre 2. Enfin, le même processus est répété pour la troisième donnée 3, donnant la position en rouge marquée avec le chiffre 3.

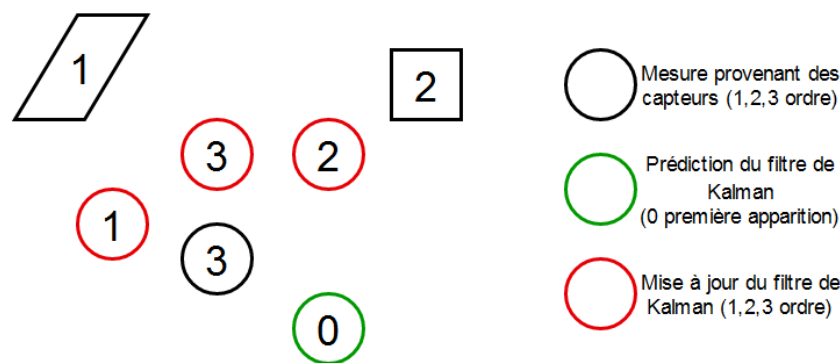


Figure 19 : Séquentialité de la mise à jour

Cette contrainte est imposée par le fonctionnement de ROS, qui sans paramétrage spécifique implique qu'une seule fonction de callback ne peut être traitée simultanément, ce mutex est modélisé par la place « A » et la place « Attente topic » qui modélise la file de données. L'accès à un callback dépendra du capteur qui arrive en premier, que ce soit le Lidar (« L »), la caméra (« C ») ou le Radar (« R »). Une fois dans le callback, nous commençons par filtrer les données pour déterminer si elles se situent à l'intérieur des limites de notre zone de digitalisation. Si les données ne sont pas acceptées, elles sont dirigées vers « >Lim », « lim » ou « >lim ».

En revanche, si les données se situent dans les limites, nous pouvons mettre à jour l'état du filtre de Kalman en passant par « MAJ », « Maj » ou « maj ». Si l'accès à la mémoire est possible, pour effectuer la mise à jour (« update », « Update » ou « UPDATE »), puis publier les résultats avec « PUB », « Pub » ou « PuB ».

Étant donné que toutes les 10 ms, une prédiction (« pred ») est effectuée sur l'état du véhicule, seule l'une des deux actions peut être réalisée, soit la mise à jour, soit la prédiction. La place « Accès » a été implémenté pour éviter l'écriture simultanée des éléments du filtre de Kalman, cela permet d'avoir un accès mutuellement exclusif soit de la prédiction soit de la mise à jour. Si nous choisissons la prédiction, nous passons par « P » pour prédire l'état du véhicule (« prédiction ») avant de publier l'état prédit (« pub »).

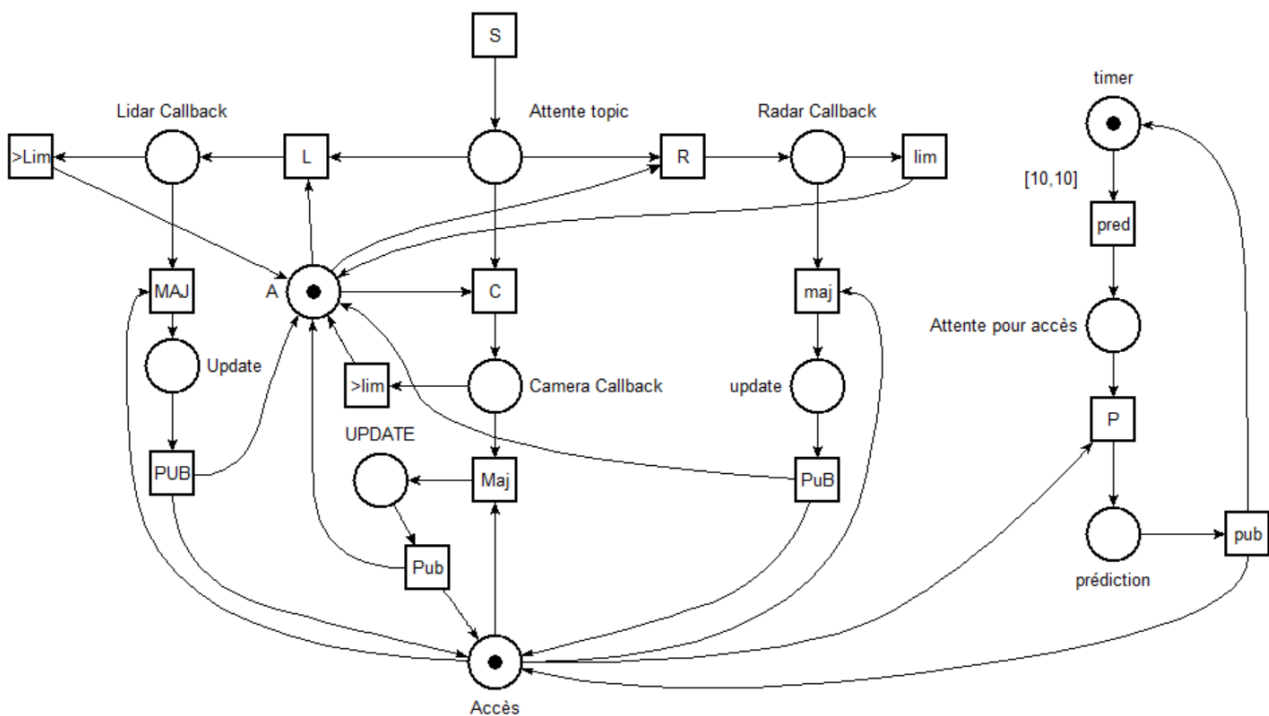


Figure 20: Réseau de Petri pour le filtre de Kalman en fonctionnement unitaire

Cette analyse nous permet de mieux comprendre la séquentialité des mises à jour dans notre système de fusion de données basé sur le filtre de Kalman. Cependant, il est essentiel de prendre en compte ces résultats dans le contexte global de notre application et de les évaluer par rapport aux performances attendues et aux objectifs spécifiques de notre projet.

2. Implémentation du Filtre de Kalman : mise à jour avec batch des traitements capteurs

La deuxième approche que nous avons explorée consiste à permettre l'utilisation de threads pour les callbacks des capteurs, afin de pouvoir traiter les données d'entrée de manière parallèle. Nous avons mis en place une fenêtre de temps, qui permet d'attendre que les données arrivent dans leurs conteneurs respectifs avant de procéder à la fusion.

Cela signifie que nous donnons un certain laps de temps aux capteurs pour que leurs données soient collectées avant de les combiner ensemble. Cela garantit que toutes les informations pertinentes sont présentes avant d'effectuer la fusion, assurant ainsi des résultats plus complets et précis. Cela nous permet de sélectionner les données les plus récentes dans la fenêtre de temps et de les fusionner en effectuant une seule mise à jour du filtre de Kalman, au lieu de trois mises à jour séparées. L'hypothèse sous-jacente déduite sur le fonctionnement est que cela offre un avantage en termes de calculs et d'accès aux mises à jour du filtre de Kalman, ce qui rend la donnée plus stable.

Cette méthode offre également un meilleur contrôle via la fenêtre de temps sur le décalage entre les données. Il est important de ne pas choisir une fenêtre de temps trop grande, afin d'éviter les erreurs temporelles potentielles lors de la fusion. De plus, il est essentiel de ne conserver que la valeur la plus récente des données et de ne pas accumuler plusieurs occurrences d'une même donnée d'un capteur dans la fenêtre de temps.

La Figure 21 illustre le processus de récupération des données dans une seule fenêtre de temps, ce qui explique pourquoi les trois données noires ont toutes une valeur de 1. Après un délai de 50 ms, nous effectuons une mise à jour de l'état de Kalman, représentée par la position en rouge. L'avantage de cette approche, comme nous pouvons le voir dans cet exemple par rapport à la Figure 19, est que nous obtenons l'état final en une seule fois au lieu de trois états séparés, ce qui confère une plus grande stabilité au système.

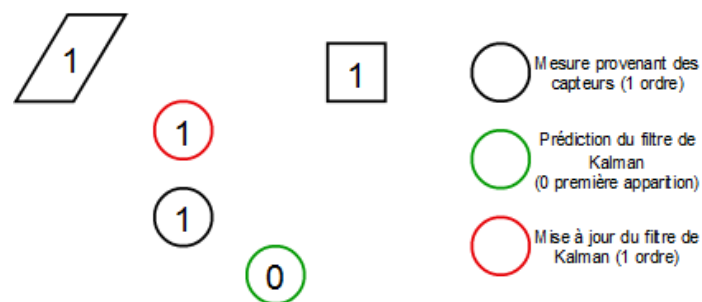


Figure 21 : Threading mise à jour

Ainsi, nous avons fait le choix de partir sur cette modélisation pour réaliser mon filtre de Kalman en fonctionnement threading. Dans un premier temps ci-dessous dans la Figure 22 une représentation du réseau de pétri pour présenter le fonctionnement du filtre de Kalman.

S (source) indique une donnée qui arrive sur ROS 2. C (resp. L, R) représente le capteur (Caméra, resp. Lidar, Radar) sur lequel arrive la donnée. Lorsque la donnée tombe en dehors des limites fixées, elle est directement évacuée vers la transition (resp. lim, lim1, lim2). Sinon :

- Si la place C_vect (resp. R_vect, L_vect) est libre, le vecteur Cv (resp. Rv et Lv) est créé.
- Si la place n'est pas vide, nous passons à une mise à jour (Maj) qui remplace la valeur du vecteur correspondant par la valeur la plus récente.

L'occupation des places C_vect, R_vect, L_vect par un jeton signifie ainsi qu'une donnée est prête à être consommée, par les transitions goC, goR et goL, respectivement. Afin d'associer les données d'une même fenêtre temporelle, ces transitions sont mises en attente par la lecture de la place Wait. Toutes les 50ms, un jeton transitionne depuis Timer_update vers Wait et les transitions goC, goR et goL sont effectuées.

A ce stade, la place All peut contenir 0, 1, 2 ou 3 jetons. La transition sync est effectuée vers la place Association_vecteur. Si all contient au moins un à trois jetons, la transition correspondante (1_top, 2_top, 3_top) est réalisée et le filtre de Kalman est mis à jour avec les données correspondantes. Enfin, la transition pubu (ou la transition rien, si all ne contenait aucun jeton) est réalisée vers la place timer_update et à la place mutex.

La partie gauche du réseau de Petri modélise la prédiction effectuée par le filtre de Kalman. La prédiction est effectuée toutes les 10ms (modélisée par le timer timer_predict) en se basant sur l'état du filtre de Kalman. La partie update et la partie predict du filtre de Kalman sont toutes les deux utilisées pour mettre à jour la même variable représentant l'état du véhicule.

L'utilisation de l'une ou l'autre est déterminée par la réception ou non de données capteurs et la différence entre la fréquence de prédiction et la fréquence d'association. Les calculs étant effectués séparément pour les deux branches, la place mutex est utilisée pour contrôler l'accès à la variable qui représente l'état du véhicule.

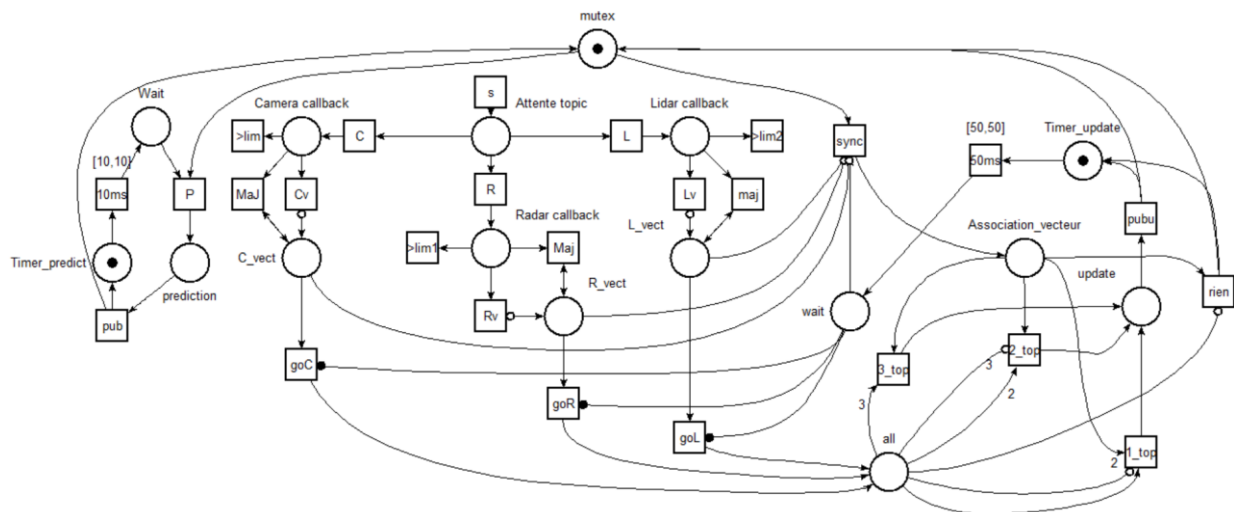


Figure 22 : Réseau de Petri du filtre de Kalman avec un batch

3. Comparaison des performances des deux méthodes

Nous verrons ici une étude qui a été effectuée sur la performance dans un scénario de nos deux méthodes. Dans un premier temps nous avons la méthode unitaire sur lesquelles nous obtenons les résultats suivant. Nous réalisons l'enregistrement d'un scénario de changement de ligne d'une durée de 7.199s avec nos capteurs. Le nombre de mise à jour a ensuite été comptabilisé dans le temps du scénario, soit : un seul véhicule en mouvement dont nous prédisons la position toutes les 10ms.

Le nombre de mises à jour de la position du filtre de Kalman s'élève à 277 avec 66 issues de la caméra, 131 du Lidar et 80 du Radar. La fusion entre les différents capteurs n'est jamais effectuée au préalable.

Nous effectuons la même analyse que précédemment appliquées à la méthode du batch, en mesurant le nombre de mises à jour lors d'un enregistrement de 7.199s comportant un unique véhicule. Le nombre de mises à jour de la position du filtre de Kalman s'élève à 126, tandis que nous comptabilisons 66 mises à jour de la caméra, 129 du Lidar et 80 du Radar. Le nombre de fusions s'élève cette fois à 73 dont 8 fois (caméra, lidar), 8 fois (caméra, radar), 24 fois (lidar, radar) et 33 fois avec les trois capteurs.

Nous pouvons constater que certaines données ne sont pas utilisées en faisant le décompte de l'utilisation des données capteurs. Une fois retirées les données consommées par les événements de fusion, il reste 17 données caméras, 64 données lidar et 15 données radar, soit 96 données individuelles. Or, si nous retirons les 73 événements de fusion, il reste seulement 53 événements de mises à jour dus aux capteurs individuels, soit 43 données non utilisées. La raison du fait que des données ne sont pas utilisées est que nous ne prenons que la dernière donnée dans la fenêtre de temps pour effectuer la mise à jour. D'où par exemple, si dans les 50ms nous avons deux fois le Lidar, une fois à 1ms et une autre fois à 41ms par exemple, nous allons effectuer la mise à jour avec les données à 41ms.

Suite à notre étude, il est évident que chaque méthode présente ses propres avantages et inconvénients. Le choix de l'une ou l'autre dépend véritablement du type de conception que l'on souhaite mettre en œuvre. Pour le cadre de notre projet, nous avons décidé d'opter pour la méthode par lots (batch).

3. Key performance Indicator (KPI)

Enfin, la dernière étape dans le processus du filtre de Kalman consiste à ajuster les matrices R et Q afin de rendre les prédictions plus précises. La matrice Q permet de définir la zone de prédiction du véhicule, ce qui nous permet d'effectuer l'association des données. La matrice R, quant à elle, est construite pour donner une confiance pondérée aux données des capteurs en fonction de leur bruit de mesure. En ajustant ces deux matrices, nous indiquons au filtre de Kalman s'il doit accorder plus de confiance à la prédiction ou aux mesures lors de la mise à jour.

Pour effectuer cet ajustement, nous collectons des mesures sur la position du véhicule récupéré depuis le simulateur Carla et la position prédite par notre filtre de Kalman de ce véhicule. Nous effectuons une différence de positions qui nous retourne nos indicateurs de performance clés (KPI). En analysant ces résultats, nous modifions les valeurs diagonales des matrices R et Q afin d'optimiser l'ajustement entre nos prédictions et l'état réel du véhicule.

Il convient de noter que cette étape d'ajustement est essentielle pour adapter le filtre de Kalman à notre application spécifique, en prenant en compte les caractéristiques des capteurs et les contraintes du système. Elle nous permet d'obtenir des estimations plus fiables de la position et de la vitesse du véhicule, ce qui est crucial pour de nombreuses applications liées à la perception et au contrôle autonomes.

5. Application

1. Système à trois capteurs

Une première configuration de capteur a été proposée pour implémenter nos solutions, cette configuration comporte une caméra, un Lidar et un Radar qui font face à la zone de digitalisation. Pour la suite, nous considérons une caméra, un Lidar et un Radar dont le positionnement est indiqué en Figure 23 (dans le rectangle rouge).



Figure 23 : Système à trois capteurs (Caméra, Lidar, Radar) sur notre carte dans notre simulateur

A partir de cette configuration nous avons effectué quelques KPI en modifiant les valeurs des matrices Q et R de notre filtre de Kalman utilisées pour nous rapprocher d'un résultat que l'on considère comme acceptable. La Figure 24 nous montre un exemple de résultat de ces mesures pour la prédiction d'un véhicule. On remarque rapidement que la prédiction de la position du véhicule sur l'axe x à une erreur comprise entre $[-2m, 2m]$. La raison pourrait venir de la position des capteurs qui capture l'avant du véhicule et nous faisons la mesure en prenant le centre du véhicule de Carla avec la position prédite de celui-ci qui provoque cette erreur et ce décalage. En revanche, nous remarquons qu'avec cet angle de vue l'erreur sur l'axe y est minime et assez rapidement dans notre zone souhaitée de $[-0.5m, 0.5m]$. Finalement, avec cette méthode nous remarquons que l'estimations de la vitesse du véhicule est proche de ce que l'on recherche puisque l'on est à presque 0 en différence.

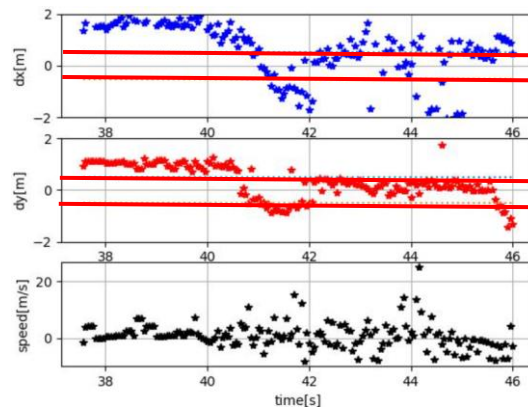


Figure 24 : Mesure des KPI pour un système à trois capteurs.

2. Système à six capteurs

L'approche de combiner les données de plusieurs capteurs nous permet de réduire les erreurs et les incertitudes, et ainsi d'obtenir des estimations plus précises de la position, de la vitesse et de l'environnement du véhicule. En ajoutant des capteurs supplémentaires dans notre système, nous avons renforcé la redondance et la complémentarité des informations, ce qui est crucial pour garantir la fiabilité et la robustesse de la fusion des données. De plus, en positionnant stratégiquement ces capteurs, nous avons amélioré la couverture et la précision de la perception en observant la zone de digitalisation dans les deux sens.

La Figure 25 comporte l'ajout des trois capteurs supplémentaires, dans les rectangles rouges, qui ont été positionnés sur un lampadaire de l'autre côté de la zone, fournissant ainsi des informations supplémentaires sur les objets et les obstacles environnants. Cette configuration nous permet d'avoir une vision plus complète de la scène et des données complémentaires aux capteurs existants.

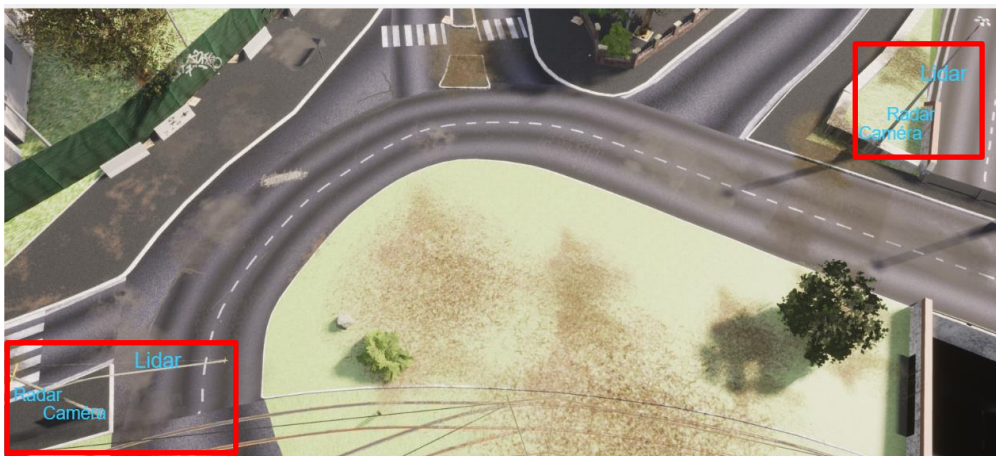


Figure 25 : Système à six capteurs avec les trois capteurs (Lidar, Radar et Caméra) respectivement dans les rectangles rouges

Cette approche améliore la sécurité et la prise de décision en fournissant des informations plus fiables et en réduisant les risques liés à la défaillance d'un capteur individuel. Sur la Figure 26 en couleur, nous avons les données du Lidar de notre système à 3 capteurs et les données en blanc correspondent aux données du Lidar rajouté.

Le nouveau Lidar couvre en partie la zone déjà couverte, mais d'un autre point de vue qui apporte une redondance en cas de défaillance, mais aussi une complémentarité d'informations. De plus, elle ouvre des perspectives intéressantes pour de futures améliorations, telles que l'intégration de nouveaux capteurs ou l'utilisation de techniques avancées de fusion des données.

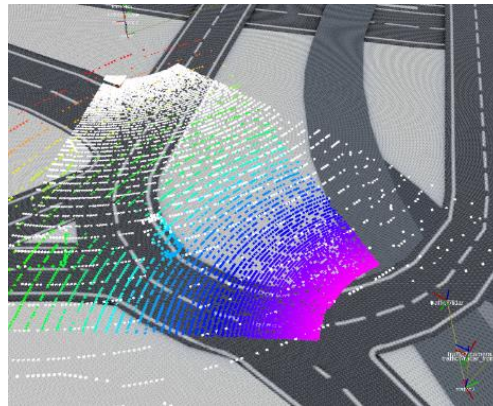


Figure 26 : Complémentarité et redondance du système à six capteurs vus par le positionnement des deux Lidars et les données communes et supplémentaires

Une étude approfondie des risques, des détections requises et des limites du système serait nécessaire pour positionner correctement les capteurs et optimiser leur champ de vision, tout en prenant en compte les différents scénarios et les caractéristiques spécifiques de l'environnement de conduite. Bien que notre disposition pratique des capteurs sur des poteaux aient donné des résultats encourageants, des recherches supplémentaires sont nécessaires pour optimiser le placement et l'orientation des capteurs, en tenant compte des exigences spécifiques de l'application et des contraintes environnementales. Ces travaux pourraient être réalisés dans le cadre de futurs projets afin d'exploiter pleinement le potentiel des capteurs en termes de détection, de redondance et de complémentarité d'informations.

La Figure 27, représente les mesures des KPI pour le système à six capteurs. Malheureusement, nous avons manqué de temps pour améliorer les résultats. Puisque, nous avons pu remarquer que pour avoir de bons résultats, nous devons modifier les matrices R et Q pour les adapter au système. Les résultats sont toutefois acceptables pour un projet à court terme mais à long terme il est nécessaire d'adapter la prédiction du mouvement et les matrices jouant sur la précision de Kalman.

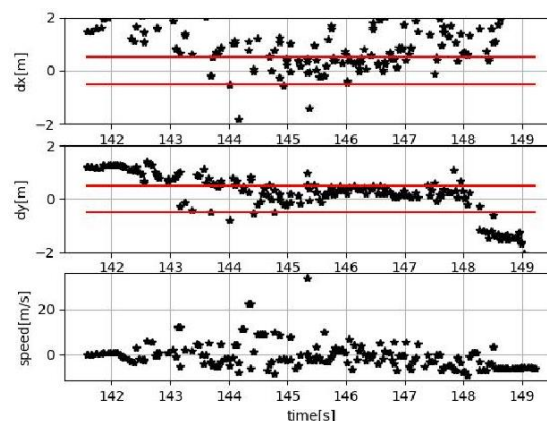


Figure 27 : Mesure des KPI pour un système à six capteurs

3. Analyse

L'ajout de capteurs offre une richesse d'informations grâce à la redondance et la complémentarité, tout en procurant des points de vue variés de la scène. Cela optimise les résultats du filtre de Kalman vis-à-vis des KPI formulés. Cependant, les résultats actuels ne corroborent pas totalement cette affirmation. En effet, les contraintes de temps ont limité notre capacité à adapter pleinement le filtre de Kalman à une configuration de six capteurs et à obtenir une prédiction aussi précise que souhaitée. Ce problème pourrait aussi être lié à notre incapacité à modéliser un comportement de conduite adéquat pour notre scénario, en raison de ces mêmes contraintes temporelles. Néanmoins, la robustesse du système est prouvée par le fait que les KPI peuvent être calculés même en présence d'une information provenant d'un seul capteur parmi les six, ou de tous les capteurs simultanément.

Conclusion et perspectives

L'objectif principal de mon stage était de contribuer au développement d'une méthode de fusion des données provenant de capteurs différents, ainsi que d'assurer la robustesse du système multi-capteurs et de permettre une prise de décision efficace basée sur les informations obtenues. Au début de mon stage, seuls deux capteurs étaient présents disposais de traitements élémentaires issus de bibliothèque sur "étagère" (autoware). J'ai proposé l'ajout d'un capteur Radar pour augmenter la redondance des informations et renforcer la robustesse de la fusion des données. J'ai étudié et testé plusieurs techniques de filtrage de Kalman et j'ai sélectionné celle qui permet de prédire avec la plus grande précision la position et la vitesse en utilisant les données de la caméra, du Lidar et du Radar.

Les résultats obtenus sont encourageants. Grâce à la fusion des données des trois capteurs, nous avons pu obtenir des estimations plus précises sur la position et la vitesse du véhicule. Les simulations et les tests réalisés sur la plateforme Carla confirment l'efficacité de notre approche. La complémentarité des informations provenant de chaque capteur permet d'améliorer la fiabilité et la résistance du système aux perturbations.

Cependant, certaines tâches n'ont pas pu être réalisées dans le cadre de mon stage. La validation expérimentale du système sur des scénarios réels aurait apporté une évaluation plus approfondie de sa performance dans des conditions réelles de conduite. Une maquette avec trois caméras était en cours de développement, mais n'a pas été finalisé pendant la durée de mon stage.

Sur le plan personnel, ce stage m'a offert une véritable immersion dans le domaine de la fusion de capteurs et du traitement des données pour la perception des véhicules autonomes. J'ai acquis de nouvelles connaissances et compétences en programmation ROS, en filtrage de Kalman et en intégration de capteurs. J'ai également développé ma capacité à travailler en équipe et à m'adapter à un environnement de recherche et développement dynamique.

En conclusion, mon stage m'a permis de contribuer activement au développement D'une méthode de perception basée sur la fusion de données de capteurs hétérogènes. Les résultats obtenus ont démontré la faisabilité de notre approche et ouvrent des perspectives intéressantes pour de futures améliorations. Malgré les tâches non réalisées, le travail accompli constitue une base solide pour poursuivre les développements futurs, notamment en explorant davantage la redondance et la complémentarité des capteurs. Je suis reconnaissant d'avoir eu l'opportunité de participer à ce projet et je suis convaincu que les travaux réalisés auront un impact positif.

Bibliographie

- [1] [Carla](#)
- [2] Robot operating system, [Ros](#)
- [3] [ros-bridge](#)
- [4] The Autoware foundation, [Autoware Documentation](#)
- [5] Point Cloud Library, PCL, <https://pointclouds.org/>
- [6] Rezaei, Mahdi, Mohsen Azarmi, and Farzam Mohammad Pour Mir. "Traffic-net: 3d traffic monitoring using a single camera." *arXiv preprint arXiv:2109.09165* (2021)
- [7] Mousavian, Arsalan, et al. "3d bounding box estimation using deep learning and geometry." *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017.
- [8] Ali, Waleed, et al. "YOLO3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud." *Proceedings of the European conference on computer vision (ECCV) workshops*. 2018.
- [9] [COCO](#)
- [10] Sasiadek, J. Z., and Pande Hartana. "Sensor data fusion using Kalman filter." *Proceedings of the Third International Conference on Information Fusion*. Vol. 2. IEEE, 2000.
- [11] Hajiyeve, Chingiz, and Halil Ersin Soken. "Robust adaptive Kalman filter for estimation of UAV dynamics in the presence of sensor/actuator faults." *Aerospace Science and Technology* 28.1 (2013): 376-383.
- [12] Akram, Muhammad Adeel, et al. "A state optimization model based on Kalman filtering and robust estimation theory for fusion of multi-source information in highly non-linear systems." *Sensors* 19.7 (2019): 1687.
- [13] Simon, Dan. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.
- [14] Adrian Rosebrock, Simple object tracking with OpenCV, [Link](#)
- [15] Murata, Tadao. "Petri nets: Properties, analysis and applications." *Proceedings of the IEEE* 77.4 (1989): 541-580

Résumé en français

Le stage s'est déroulé au sein du projet STRASS, un projet de recherche et innovation de Capgemini Engineering. Il se concentre sur le développement de divers sous-systèmes pour l'aide à la conduite des tramways. L'objectif principal est de détecter les risques de collisions dans les zones à risques, en garantissant une sécurité accrue. Nous nous concentrons ici sur la fusion de données des capteurs afin de prédire la position et la vitesse des véhicules.

Nous présentons ici les différentes composantes de notre solution. Nous utilisons la simulation développée sous Carla et le logiciel ROS pour explorer différentes stratégies de traitements des données capteurs. Nous utilisons dans un second temps les résultats de ces traitements comme entrée pour la fusion des données capteurs à l'aide d'un filtre de Kalman. La fusion des données joue un rôle crucial dans notre approche. Nous examinons les techniques de mise en forme et d'homogénéisation des données, essentielles pour les rendre compatibles, avant de les soumettre au filtre de Kalman. De plus, nous présentons les diverses applications possibles de ce filtre, ouvrant ainsi de nouvelles perspectives.

Cette étude s'intègre dans le domaine de l'aide à la conduite des tramways. La fusion des données constitue l'élément clé de la fiabilité et de la précision de notre solution. Les résultats prometteurs obtenus soulignent le potentiel de notre système, qui contribuera significativement à la sécurité et à la prise de décision dans les zones à risques.

Résumé en anglais

The internship takes place within the STRASS project, a research project at Capgemini Engineering. It focuses on developing various subsystems for tramway driving assistance. The main objective is to detect collision risks in high-risk areas, ensuring enhanced safety. Specifically, this internship focuses on the data fusion aspect of sensor data to predict position and velocity of various vehicles.

This report details the various part of our solution. Using Carla simulation and ROS programming, we thoroughly explore the data processing strategies for the sensors. These carefully designed processing techniques prepare the data for fusion using a Kalman filter.

Data fusion plays a crucial role in our approach. We examine data formatting and homogenization techniques, essential for making the data compatible before subjecting it to the Kalman filter. Additionally, we present various potential applications of this filter, thereby opening new perspectives.

This study adds to the field of tramway driving assistance. Data fusion is the key element for the reliability and precision of our solution. The promising results obtained underscore the considerable potential of our system, which will significantly contribute to safety and decision-making in high-risk areas.