Luc Boulesteix

IUT d'Orsay
DUT Mesures Physiques

DUT MESURES PHYSIQUES

Instrumentation d'un véhicule électrique pour la perception de l'environnement.

Étude et mise en fonctionnement d'un système multi-capteur embarqué.

Luc BOULESTEIX

Tuteur entreprise :

Sergio Rodriguez

Tuteur IUT :

Bastien Vincke

Année universitaire 2021/2022

Soutenance du 23 juin 2022

S4, spécialité Techniques Instrumentales

Glossaire :

« Bloc logiciel » / « Bloc de perception » : logiciel et/ou algorithme implémenté sur le véhicule assurant une tâche précise.

« Plateforme » / « véhicule » / « Plateforme expérimentale » : Renault Twizy instrumentée.

« Espace 3D/2D » : espace orthonormé et direct, de dimension finie associé à un repère de même dimension.

« Repère » : système de vecteurs à partir duquel nous pouvons définir tout point d'un espace (ex : repère cartésien).

« Espace plan » : espace 2D sur lequel on projette des données LIDAR en fonction de leurs coordonnées.

« ADAS » : sigle en anglais qui dénote Advanced Driver Assistance System, système d'aide à la conduite.

« Véhicule Robotisé » : véhicule capable d'agir au travers d'actionneurs sur ces dégrées de liberté ex. longitudinal et latéral.

« LiDAR » sigle en anglais dénotant Light Detection And Ranging, une technologie de télémétrie effectuant une mesure de distance par temps de vol.

« Caméra RGBD » / « caméra 3D » : caméra capable de fournir une image de profondeur issue de l'appariement stéréoscopique des pixels.

Remerciements

Nous tenons tout d'abord à débuter ce rapport en apportant nos sincères remerciements à ceux qui nous ont accompagné au cours de ce stage, ainsi qu'à tous ceux qui ont rend possible cette expérience professionnelle.

Nous remercions Bastien Vincke, notre tuteur IUT, qui nous a accompagné tout au long de cette période de stage.

Nous remercions Sergio Rodriguez, notre tuteur en entreprise, pour son accompagnement au cours de cette période de stage ainsi que pour son aide.

Nous remercierions également l'équipe technique de l'IUT qui s'est occupé de bons nombres de travaux sur le véhicule à instrumenter.

Nous voudrions aussi remercier le laboratoire SATIE, qui nous as accueilli lors des vacances de Pâques, période durant laquelle l'IUT était fermé.

Enfin, nous remercions l'UTAC CERAM et la Société des Ingénieurs de l'Automobile, qui ont organisé le concours « Challenge UTAC » auquel l'IUT a participé.

Table des matières

I. Introduction

Les véhicules à conduite automatisée, communément appelés « véhicules autonomes », sont un vecteur technologique en pleine expansion. Cette expansion est née d'une réduction massive du coût des composants (capteurs, calculateurs) nécessaires à leur développement, ainsi que d'avancées dans le domaine de l'intelligence artificielle, et motivée par l'importance grandissante accordée à la mobilité dans un monde de plus en plus interconnecté.

Ces systèmes robotiques complexes sont modélisables par une boucle de fonctionnement comprenant trois blocs : la perception de l'environnement, le traitement de données résultantes et enfin l'exécution de commandes agissant sur l'état du véhicule. En tant que premier bloc dans cette chaîne, la perception de l'environnement joue un rôle clé dans le bon fonctionnement d'un tel véhicule.

C'est pour cette raison que l'IUT d'Orsay, en lien avec l'Université Paris Saclay, a fait l'acquisition d'un véhicule dont l'objectif sera de servir de plateforme expérimentale pour l'étude de systèmes de perception.

Le stage de fin d'études en DUT Mesures Physique est un stage professionnalisant d'environ 3 mois, organisé en fin de deuxième année. Dans le cadre de notre stage de fin d'études, nous avions la volonté de choisir un sujet de stage en lien avec notre projet professionnel futur, qui porte sur le domaine de l'informatique embarquée et son application sur véhicule. C'est dans cette optique que nous avons postulé à l'offre de stage proposée par l'IUT aux étudiants ayant participé au projet « Perception Mobile », projet pour lequel nous nous avons assumé le rôle de chef d'équipe.

Ce stage nous permettrait d'avoir un premier contact avec le domaine que nous visons pour notre avenir professionnel, et nous permettrait de manipuler des technologies et techniques nouvelles réellement utilisées en industrie.

1. Cadre du stage

Le stage s'est déroulé dans les locaux de l'IUT d'Orsay, et plus spécifiquement au sein du laboratoire pédagogique d'instrumentation (rattaché au département mesures physiques) dans lequel la plateforme expérimentale est hébergé. Ce stage a été encadré par Mr. Sergio Rodriguez (tuteur entreprise) et Mr. Bastien Vincke (tuteur IUT).

Le stage aura lieu du 11 avril au 24 juin 2022, avec un rythme horaire 9h-17h. Le suivi de l'avancement du projet serait assuré par une correspondance hebdomadaire avec Mr. Vincke (côté IUT) ainsi qu'une correspondance régulière avec Mr Rodriguez (côté entreprise), en complément de visites régulières de sa part.

L'IUT étant fermée sur la période des vacances de Pâques, le laboratoire SATIE (rattaché à l'ENS Paris Saclay) nous as temporairement accueillis pour pouvoir travailler pendant cette période.

2. Présentation de l'entreprise

L'Université Paris Saclay est une université fondée en 2019. Elle est constituée d'une grande variété d'institutions dont l'Université Paris Sud. L'IUT d'Orsay, précédemment une composante de Paris Sud, est maintenant inscrite dans le périmètre de l'Université Paris Saclay.

L'IUT d'Orsay existe sur le plateau du Moulon à Orsay depuis 1970. Cet établissement est constitué de trois départements : le département informatique, le département chimie et le département mesures physique. C'est au sein du département mesures physique qu'on retrouve le laboratoire pédagogique d'instrumentation de l'IUT d'Orsay, laboratoire dans lequel j'entame mon stage.

3. Enjeux et objectifs du stage :

Tout système de perception de l'environnement nécessite un apport régulier et fréquent de données sur le milieu ambiant, données qui sont générées par une suite de capteurs implantés sur le véhicule.

Chaque type de capteur (caméra, télémètre LiDAR, RADAR, etc.) admet par nature certaines forces et certaines faiblesses, que ce soit en termes de coût, de précision, d'étendue de mesure, de fréquence, etc. Une caméra vidéo classique par exemple est un capteur très polyvalent en pleine journée, mais perds de son utilité dans des environnements peu éclairés. De plus, une application sur véhicule nécessite une fiabilité maximale, toute défaillance pouvant avoir des conséquences désastreuses sur un véhicule en fonctionnement.

Les objectifs d'un système de capteurs multi-capteurs sont donc multiples : l'utilisation de plusieurs capteurs fournit non seulement une redondance nécessaire à la mise en place d'un système sécuritaire tout en élargissant le spectre de fonctionnement du système.

L'obstacle majeur à l'implantation d'un système multi-capteur est l'exploitation adéquate de données fournies par une multitude de sources hétérogènes, et donc la mise en correspondance des données générées par l'ensemble des capteurs présents.

La plateforme expérimentale de l'IUT d'Orsay, détaillée par la suite, possède plusieurs capteurs implantés avant le début du stage. Notre mission au cours de ce stage est donc la suivante :

- Comprendre le fonctionnement des systèmes multi-capteurs
- Développer et implanter des algorithmes exploitant plusieurs capteurs pour la mise en correspondance des données.

4. Le plan de ce rapport

La suite de ce rapport sera divisée en plusieurs parties. Dans un premier temps, nous parlerons des moyens mis à notre disposition, en détaillant la plateforme expérimentale et les capteurs embarqués. Nous parlerons ensuite brièvement de l'architecture logicielle que nous exploiterons pour cette mission.

Dans un second temps nous parlerons des contributions apportées au cours du stage en termes d'étalonnage du système, travail nécessaire à l'implantation d'algorithmes de perception.

Dans un troisième temps, nous aborderons l'application de ces résultants dans l'objectif de mettre en correspondance les capteurs embarqués. Nous détaillerons entre autres les algorithmes mis en place démontrant l'exploitation du système multiplicateurs. Chaque algorithme fera l'objet de sa propre partie-mission. Nous aborderons ensuite les résultats obtenus pour chaque algorithme.

Enfin, nous parlerons des retombées de ce travail, puis nous conclurons.

Comme indiqué précédemment, un véhicule à conduite automatisé est modélisable à l'aide d'une chaîne de fonctionnement contenant plusieurs blocs, dont un bloc de perception. C'est dans le cadre de ce bloc de perception que nous avons apportés nos contributions au cours du stage, que ce soit en termes d'étalonnage du système de capteurs ou de mise en correspondance inter-capteurs. L'ensemble des contributions et leurs relations dans le cadre de ce bloc de perception sont illustrés en *Figure 1* :

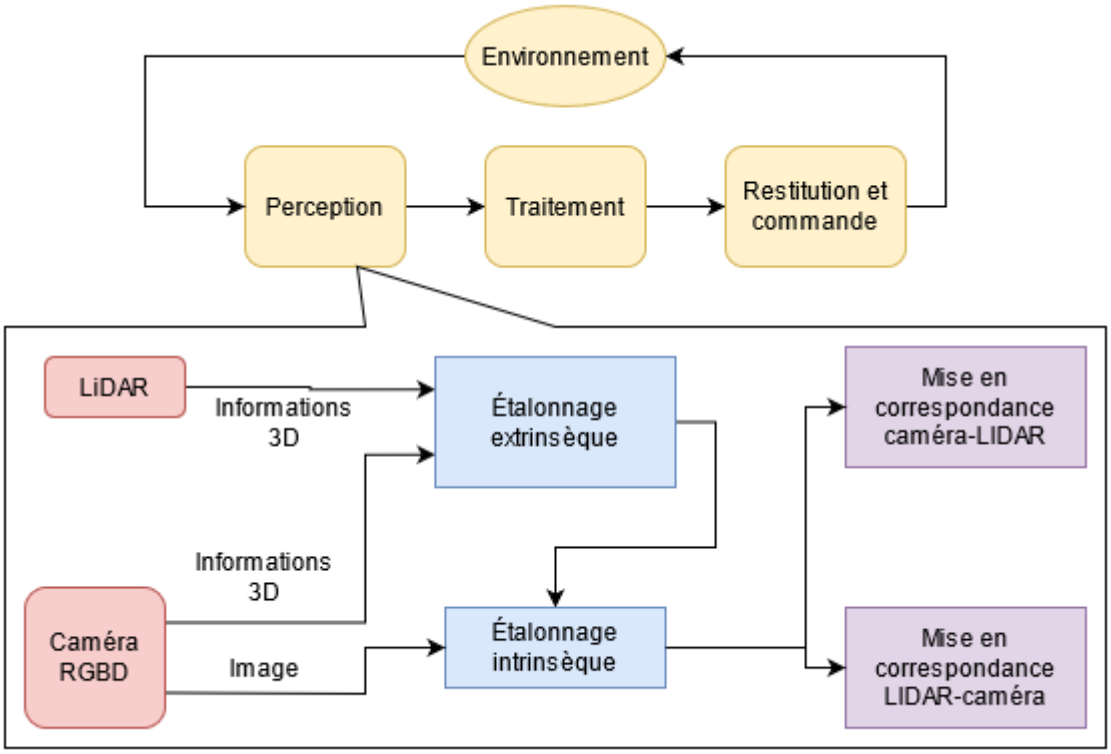

Figure 1: Schéma bloc d'un véhicule à conduite automatisé et contributions en stage

II. Le matériel

L'IUT d'Orsay a fait l'acquisition d'un véhicule dans le but d'étudier les systèmes de perception. C'est sur ce véhicule que nous mettrons en place les blocs logiciels nécessaires à l'exploitation de données multi-capteurs.

1. Le véhicule

La plateforme expérimentale utilise une voiture de série : une **Renault Twizy**. Ce véhicule électrique à deux places est classé comme un quadricycle lourd, et est destiné à une utilisation en milieu urbain et proche-urbain.

Ce véhicule (visible dans la *Figure 2)* affiche une autonomie d'une centaine de kilomètres et une vitesse maximale de 90km/h.

Figure 2: le véhicule utilisé comme base de la plateforme de perception, muni de ses capteurs (photo prise le 12/05/22 à Linas-Montlhéry)

2. Le matériel embarqué

Une variété de capteurs sont embarqués sur le véhicule, capteurs qui seront exploités dans le cadre de notre mission. Ces capteurs sont attachés au véhicule par l'intermédiaire d'une barre métallique, elle-même fixée rigidement au toit du véhicule.

On distingue plusieurs capteurs sur le véhicule (visibles en *Figure 3)* :

- Un télémètre LiDAR de marque **Velodyne Puck16**. Ce capteur, utilisant un LASER (classe 1), est capable de fournir des mesures de distance à 360° autour du véhicule sur 16 nappes verticales, jusqu'à 20 fois par seconde.
- Deux caméras RGBD de marque **Intel Realsense D435**. Ces caméras capables de fournir des images 3D, utilisant la vision stéréoscopique et un procédé de vision à base de lumière structurée afin de fournir une carte de profondeur en conjonction avec une image couleur classique. Une seule de ces caméras RGBD est utilisée dans le cadre de notre stage.

Figure 3: (à gauche) Télémètre LIDAR Velodyne, (à droite) Caméra RGBD Intel

Le traitement de données générées par les capteurs embarqués est effectué par un PC portable fonctionnant sous le système d'exploitation Linux (Ubuntu 20.04 LTS 64bit). L'écran affiche en temps réelle l'ensemble des données issue du système de perception.

Le véhicule possède également un écran de 10 pouces fixé au tableau de bord. Cet écran affiche les données recueillies par le système de perception, à destination du conducteur. Ces informations peuvent également être transmises vers un affichage déporté à l'aide d'un transmetteur radio HF.

3. L'architecture logicielle

Le système multi-capteurs est coordonné à l'aide d'une couche logicielle existante nommée Robot Operating System (ROS). ROS est une couche logicielle dont la tâche est de récolter et diffuser les données générées par les différents capteurs du système afin d'assurer leur disponibilité.

A l'instar du protocole MQTT, la circulation d'informations inter-procès sous ROS se fait sur la base de « topics » classant les données en fonction de la provenance et de la nature des informations communiquées. Les différents blocs logiciels implantés peuvent « souscrire » à ces topics afin de recevoir toute nouvelle information de ce type dès leur apparition, ou au contraire peuvent « publier » des nouvelles informations sur différents topics afin de mettre ces données à disposition. Ces informations peuvent être de nature variée : des images, des nuages de points, des rapports de détection…

Toute nouvelle donnée générée par les différents capteurs est publiée afin d'être mise à disposition des différents blocs logiciels du système pour un traitement éventuel. Le développement de blocs logiciels sera effectué en python, langage qui nous ai relativement bien connu. Le langage de programmation python joui également d'une communauté massive, ce qui met une énorme variété de librairies à notre disposition. Il nous permettra donc une mise en application relativement rapide des principes explicités dans ce document.

III. Communication inter-capteur, mise en correspondance d'informations

1. Prérequis théoriques

Le partage d'informations entre capteurs sur un système multi-capteurs nécessite de pouvoir « fusionner » les données d'un capteur avec les données d'un autre capteur, dans l'attente d'exprimer l'ensemble des données dans un repère commun.

Un exemple concret est celui des caméras de recul installées sur bon nombre de véhicules récents, qui font apparaître des informations tel que la distance séparant le véhicule d'un obstacle, et ceci directement à l'écran. Ce type de système prélève des informations provenant d'une multitude de sources (d'un radar/sonar et d'une caméra couleur par exemple) et les fait apparaître dans le flux de données (incrustation) généré par un capteur source (ex : l'image générée par la caméra).

Chaque capteur fournis des mesures exprimés dans un repère qui lui est propre, et dont la dimension dépend de sa nature. Une bonne mise en correspondance des informations nécessite donc de transposer les données générées par chaque capteur dans un repère commun (tel que dans l'image acquise par une caméra par exemple).

a. Passage entre repères de même dimension

Transposer des données d'un repère vers un autre repère de même dimension nécessite de quantifier précisément la transformation rigide encodant la position et l'orientation relative de ces deux repères. Cette transformation rigide d'un repère U vers un repère V, que nous noterons ${}^{U}[S]_{V}$, se décompose en deux transformations élémentaires : un mouvement de translation et un mouvement de rotation (Corke, 2011).

Afin de rendre possible ces mouvements, nous exprimons les coordonnées de tout point de l'espace à l'aide d'un système de coordonnées *homogène.* Ce système de coordonnées est détaillé en annexe (Annexe A).

Dans le cas des repères cartésiens 3D, ces transformations prennent la forme suivante :

Translation (différence de position entre les origines des repères) :

$$T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Rotation (différence d'orientation des repères) :

$$R = R_x(\alpha) * R_y(\beta) * R_z(\theta)$$

Où $R_x, R_y \; et \; R_z$ sont des rotations élémentaires selon les axes $x, y \; et \; z$ respectivement, définis comme suis :

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}, \quad R_y(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}, \quad R_z(\gamma) = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

La matrice finale ${}^{U}[S]_{V}$ est la composition de ces deux mouvements, et sous forme de coordonnées homogènes peut s'exprimer comme suit :

$${}^{U}[S]_{V} = \begin{pmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1 \end{pmatrix}$$

L'application de cette transformation permet de transposer les points exprimés dans un repère U propre à un capteur, vers un repère V propre à un autre capteur. Cela, à condition de caractériser correctement les coefficients contenus dans chacune de ces matrices. Le procédé permettant l'estimation des coefficients de cette matrice est dénommé **étalonnage extrinsèque.**

b. Passage entre repères de dimensions différentes : le cas du modèle sténopé

Comme dans le cas précédent, le fait de transposer les points d'un repère vers un autre passe par une transformation mathématique dont les coefficients sont à déterminer. Cependant, le passage d'un espace d'une dimension vers un espace d'une autre dimension admet une matrice de transformation qui n'est à priori pas de forme carrée, et n'est donc pas rigoureusement inversible. Tandis que le passage d'un espace de dimension N vers un espace de dimension K (où $K < N$) est possible, cette opération entraîne fondamentalement une suppression d'informations. L'opération inverse, si elle est possible, ne restitue pas l'information originale dans son intégralité.

Cette transformation est d'intérêt tout particulier pour nous car notre véhicule possède une caméra embarquée, dont les mesures (des pixels de couleur) sont exprimées dans un repère 2D (l'image générée par la caméra). Nous voulons en effet pouvoir estimer la transformation mathématique permettant d'associer un point de l'espace autour de la caméra (un repère 3D) avec un pixel de l'image.

L'exercice consiste donc à modéliser la caméra utilisée comme un objet mathématique appliquant cette transformation. Dans cette démarche, nous choisissons de modéliser cet appareil grâce au modèle **sténopé** (Orteu, 2008)**.** Cette modélisation permet de décrire le processus projectif de formation des images par une caméra, dont la lentille est supposée comme respectant les conditions de Gauss.

La matrice associée à la modélisation sténopé, dénommée « matrice de projection », est composée de coefficients qui ne sont autre que les paramètres dits « intrinsèques » de la caméra, à savoir la distance focale horizontale et verticale (x et y respectivement) de la lentille et le centre optique de l'image (de coordonnées x et y). Nous la noterons $[K]$, et elle prend la forme suivante :

$$[K] = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Une bonne estimation des paramètres de la caméra permet de déterminer, pour tout point de l'espace dont les coordonnées sont exprimées par rapport au repère 3D de la caméra, les coordonnées du pixel correspondant sur l'image. Nous appelons le procédé permettant l'estimation des coefficients de cette matrice l**'étalonnage intrinsèque.**

c. Composition de transformations

Les matrices explicitées ci-dessus permettent de transposer les points d'un repère dans un autre repère. Il en découle donc que l'application de transformations successives entre différents repères peut permettre de lier deux repères via des repères intermédiaires sans qu'il ne soit nécessaire de connaitre la transformation les liant directement.

Soit A, B et C trois repères et soit ${}^{A}[\beta]_{B}$ et ${}^{B}[\beta]_{C}$ les transformations permettant de transposer les points de A vers B et de B vers C respectivement. Nous pouvons obtenir la transformation permettant de transposer les points de A vers C par composition des deux premières transformations :

$$ {}^{A}[\beta]_{C} = {}^{A}[\beta]_{B} \oplus {}^{B}[\beta]_{C} $$

Où $\oplus$ est l'opérateur de composition de ces transformations. Lorsque ces transformations sont modélisées par des matrices, cette opération prend la forme d'une multiplication matricielle.

Dans le cas du système de capteurs considéré, nous pourrons obtenir la matrice permettant de transposer les points LIDAR dans l'image générée par composition de la matrice transposant les points LIDAR dans l'espace caméra $[S]$ et de la matrice de projection $[K]$, tel que la matrice finale $[\xi]$ s'écrive :

$$ [\xi] = [K] * [S] $$

2. Application

La mise en place d'un système multi-capteurs nécessite une bonne connaissance des transformations permettant de transposer les mesures d'un capteur dans le repère d'un autre capteur. L'acquisition de ces paramètres est donc un première partie-mission que nous devons accomplir avant de pouvoir mettre en place une association et fusion de données.

Dans le cas du système de capteurs implanté sur le véhicule de l'IUT, il nous sera donc nécessaire de quantifier :

- Les paramètres intrinsèques de la matrice de projection de la caméra embarquée permettant de projeter les points du repère 3D associé à la caméra sur l'image générée par celle-ci.
- Les paramètres extrinsèques associés à la matrice permettant de transposer les points du repère 3D associé au télémètre LIDAR dans le repère 3D associé à la caméra.

Nous envisageons ici d'estimer manuellement les paramètres de ces transformations, à savoir :

- Les angles associés aux rotations élémentaires selon x, y et z.
- Les coordonnées selon x, y et z de la translation (qui n'est autre que la séparation spatiale des deux capteurs).

Les paramètres intrinsèques ont été obtenus déterminés précédemment dans le contexte d'un projet tutoré: leur acquisition ne sera pas abordée.

Une première étude du véhicule nous permet d'estimer les paramètres nécessaires. Nous développons une application dont l'objectif est de projeter les mesures du télémètre LIDAR embarqué sur le flux vidéo de la caméra embarquée. Les paramètres rendant cette opération possible (qui sont les paramètres recherchés cités ci-dessus) seront modifiables par l'utilisateur, permettant un alignement manuel des données LIDAR avec l'image. Ce travail est effectué à l'aide d'un enregistrement contenant les données générées par les capteurs embarqués, dont une série d'image couleurs et une séquence de nuages de points LiDAR.

Le premier obstacle rencontré consiste à appairer les données générées par nos capteurs : les capteurs fonctionnant à des fréquences d'acquisition différentes (10 Hz et 30 Hz pour le télémètre et la caméra RGBD respectivement) l'émission de données par chaque capteur (et donc son stockage dans l'enregistrement horodaté) n'est pas régulière Nous appairions donc chaque image à la donnée LiDAR la plus récente, grâce à l'horodatage associé à chaque mesure.

Le protocole que nous visons à mettre en place est le suivant :

1) Les paramètres que nous cherchons à estimer sont stockés dans un fichier. Ce fichier est lu et les matrices nécessaires à la projection sont changées en mémoire.
2) En considérant un couple de données LiDAR/caméra associés, nous regardons l'ensemble des points LiDAR dans le nuage de point :
    - Nous éliminons les points susceptibles d'être en dehors du champ de vision de la caméra. Seuls les points dont l'azimut est compris dans un intervalle prédéterminé réglable sont conservés, limitant le nombre de points à traiter.
    - Les points restants sont chacun projetés dans le repère de l'image en appliquant la transformation $[\xi]$ aux coordonnées homogènes de chaque point. Nous plaçons un point de couleur sur l'image aux coordonnées obtenues. La couleur de ce point sera choisie en fonction de la distance mesurée, les points bleu correspondant à des mesures à courte distance tandis que les points rouges sont plus distants.

    Les coordonnées homogènes du point à l'écran sont obtenues via l'application de l'opération suivante (où P représente les coordonnées du point) :

$$P_{2D_{im}} = [\xi] * P_{3D_{LiDAR}}$$

    - L'image est affichée à l'écran, et le traitement du prochain couple de données commence. Si l'utilisateur commande un changement de paramètres, les contenus du fichier sont modifiés et les matrices sont réinitialisées.

3. Résultats

L'étalonnage entrepris nous permet de déterminer de façon approximative les paramètres recherchés. L'image finale obtenue permet de visualiser les mesures faites par le télémètre LiDAR. La cohérence entre la profondeur des points projetés et la scène à l'écran nous assure un réglage qualitatif des paramètres extrinsèques.

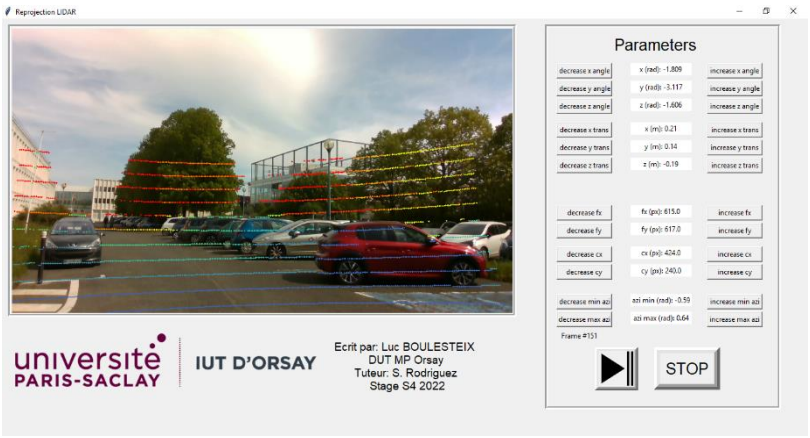

Figure 4: Alignement de données LiDAR sur une image

On notera cependant que ce procédé manuel est long et peut induire l'utilisateur en erreur : à plusieurs reprises nous avons trouvé une série de paramètres dont le rendu à l'écran semblait bon pour l'image considérée, mais qui donnait une image complètement incohérente une fois que la scène avait changée (ambiguïté perspective).

Figure 5: Un exemple de paramétrage incorrect

Critiquement, on notera que cette modélisation est fondamentalement limitée par le fait qu'elle suppose que les capteurs du système sont entièrement solidaires les uns par rapport aux autres. Or ce n'est pas strictement le cas : ces derniers bougent les uns par rapport aux autres lorsque le véhicule est en mouvement, par vibration par exemple (faible déplacement). De plus, l'asynchronisme des deux capteurs peut engendrer des erreurs dans la reprojection selon la dynamique de la scène.

Un alignement automatique de l'image avec les données LiDAR à l'aide d'un algorithme d'alignement de nuages de points nous permettrai d'évaluer l'incertitude associé à cet alignement, et pourrait constituer une perspective d'avenir. Ce travail n'est cependant pas entrepris dans le contexte de ce stage.

4. Vérification d'hypothèses : mesure de l'impact de variations de paramètres sur la reprojection

Le travail rapporté au chapitre suppose que les paramètres du système ne varient pas au cours du temps. Or ceci n'est pas strictement le cas :

- Les capteurs sont fixés au véhicule à l'aide de supports en plastique PLA. Ce plastique devient malléable à partir de 40-50°C, températures qui sont atteignables en plein soleil.
- La lentille sur la caméra est éventuellement sensible à des changements dans sa géométrie selon les conditions de température et d'humidité, ce qui entraînerait des répercussions sur la distance focale de la lentille.

Pour vérifier que les hypothèses effectuées sont plausibles, nous réalisons le protocole suivant :

1) Nous construisons un point dans l'espace LiDAR situé à une certaine distance du véhicule (ex : 10m) et le projetons sur l'image de la caméra.
2) Nous faisons ensuite varier individuellement chaque paramètre (rotation, translation, distance focale et centre optique) de façon aléatoire dans un intervalle centré autour de la valeur obtenue lors de l'étalonnage. Dans chaque cas, nous construisons une nouvelle matrice à l'aide de ces paramètres modifiés et nous projetons ce même point sur l'image.
3) La transformation LIDAR-image ayant changé, les coordonnées du point projeté sur l'image auront changés. Nous calculons la distance entre ces deux coordonnées.
4) Cette opération est répétée pour une même distance et pour un même paramètres une cinquantaine de fois de sorte à modéliser des états possibles des paramètres du système. Nous relevons ensuite la distance (en pixels) moyenne et répétons cette opération pour un autre paramètre, puis ensuite pour une autre distance.

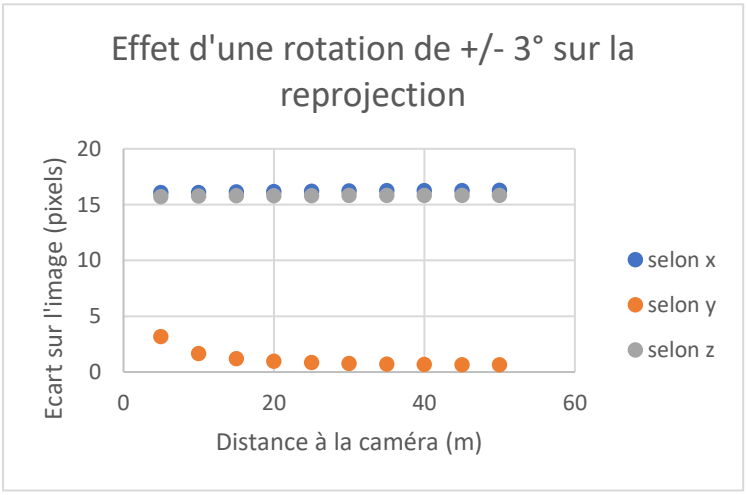

Figure 6: Effet d'un mouvement de rotation sur la reprojection

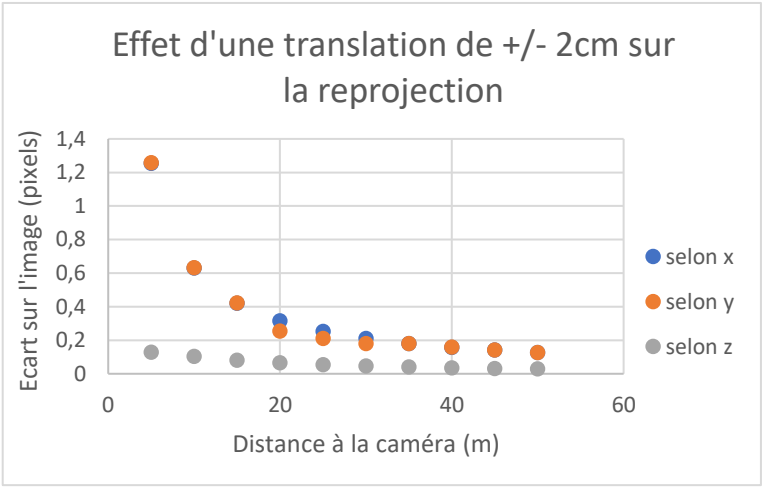

Figure 7: Effet d'un mouvement de translation sur la reprojection

L'objectif est ici de juger si des variations considérées comme « normales » des paramètres du système (exemple : mouvement de quelques millimètres entre les capteurs) a un impact significatif sur la reprojection que l'on obtient. A ces fins, nous décidons d'évaluer l'effet d'un mouvement de rotation de $\pm$ 3° d'un capteur par rapport à l'autre sur la reprojection, d'une translation de $\pm$ $2cm$, ainsi qu'une variation de $\pm$ 10 pixels de la distance focale de la caméra.

Les résultats obtenus se démarquent selon 2 cas distincts (et sont visibles en *Figure 6, 7 et 8*) :

- Les variations qui n'ont que peu d'impact sur la reprojection (écart d'un pixel au plus)
- Les variations ayant un large impacte sur la reprojection (écart de plusieurs pixels).

Les variations de distance focale entrent dans ce premier cas, avec des variations de 10 pixels sur la distance focale de la lentille entraînant des erreurs de reprojection de l'ordre de 1 pixel. Ces variations ne sont pas détectables sur une image haute résolution.

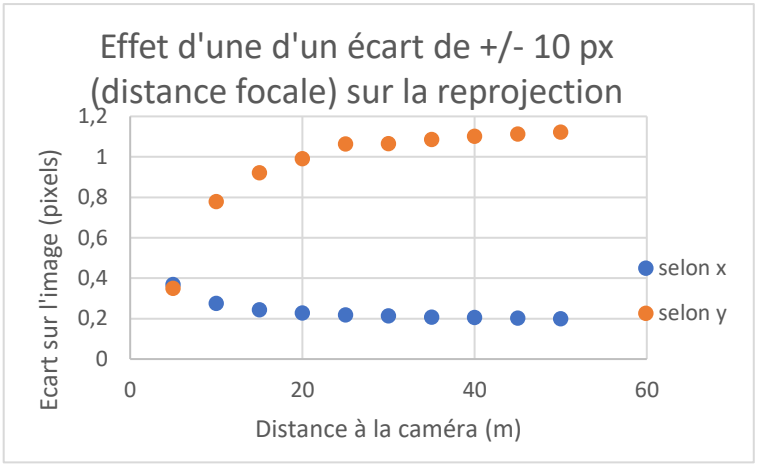

Figure 8: Effet d'une variation de distance focale sur la reprojection

Les mouvements de translation n'ont également pas un impact significatif sur la qualité de reprojection, avec des mouvements de plusieurs centimètres (on rappelle que le plastique utilisé est tout de même relativement rigide) n'entraînant que des variations à hauteur de 1 ou 2 pixels, et ceci seulement à courte portée. Ces variations sont également non perceptibles sur une image haute résolution.

On remarque cependant que les variations dues à la rotation des capteurs ont un effet particulièrement délétère sur la qualité de la reprojection, avec des rotations de quelques degrés entraînant des erreurs de reprojection de 10 ou 20 pixels, ce qui est visible sur l'image obtenue, comme on l'observe sur la *Figure 9.*

Figure 9: Erreurs de reprojection suites à la rotation des capteurs. En rose : point centrale sans distorsions, La couleur utilisée dépend de la distance du point au véhicule (bleu = lointain)

En raison de ces résultats, nous devons recommander que le véhicule ne soit pas garé au soleil pour des durées prolongées, car ceci pourrait modifier les paramètres du système. En l'absence de travaux visant à quantifier le potentiel de déformation des support utilisés, nous ne pouvons cependant pas emmètre de recommandations concernant le changement de ces supports.

IV. Algorithmes de traitement multi-capteurs :

La quantification des paramètres intrinsèques et extrinsèques nous permet de procéder à l'objectif principal de notre mission, à savoir la mise en place de blocs logiciels visant à effectuer la mise en correspondance d'informations entre capteurs en temps réel.

Au cours de ce stage, nous avons entrepris de démontrer plusieurs types de mise en correspondance d'informations inter-capteurs :

- La projection de données LiDAR dans le champ de vue de la caméra
- La détection du plan de route et des bords de la chaussée dans le but de surligner la route dans le flux vidéo

- L'utilisation de systèmes de vision par intelligence artificielle/CNN pour démarquer les utilisateurs de la route dans l'espace 3D LIDAR.

Chacune de ces démonstrations sera détaillée dans cet ordre en faisant l'objet de sa propre sous-partie. Les résultats obtenus seront détaillés par la suite.

1. Projection LIDAR-image

Le travail abordé dans le cadre du chapitre III nous a permis de développer les outils nécessaires à la projection de données LIDAR sur un flux vidéo. Nous souhaitons étendre ce travail à une application en temps réelle : le véhicule doit être capable de générer l'image composite (scène + LIDAR projeté) à l'aide des données générées par le système de capteurs.

Le principe opératoire est ici principalement le même que dans le chapitre précèdent, à l'exception de la synchronisation des données image et LIDAR. Précédemment, nous appairions les données capteurs avant de procéder au traitement de celles-ci, en fonction de leur horodatage. La synchronisation est maintenant assurée par un bloc « synchroniseur » qui assure la tâche de coupler des données dont l'écart temporel est relativement faible (moins de 50ms)

La détection de données LiIDAR et l'image synchronisées servira de condition nécessaire au traitement du couple de données en question. Nous suivrons ensuite le protocole suivant :

1) Les paramètres du système sont chargés depuis un fichier stocké sur l'ordinateur, servant à former la matrice $[\xi]$ permettant de transposer les données LIDAR sur l'image.
2) En considérant l'ensemble des points dans le nuage de points :
    a. Nous éliminons les points susceptibles d'être en dehors du champ de vue de la caméra, c'est-à-dire l'ensemble des points dont l'azimut est en dehors d'un intervalle prédéterminé.
    b. Nous appliquons la transformation $[\xi]$ aux coordonnées des points restants afin de calculer leurs coordonnées dans l'image. Nous plaçons ensuite un point coloré en ces coordonnées.

$$P_{2D_{im}} = [\xi] * P_{3D_{cam}}$$

3) L'image formée est rendue accessible pour affichage sur l'écran embarqué du véhicule.

Le développement de ce bloc logiciel rends possible la reprojection des données LIDAR dans un flux vidéo et représente un premier exemple de mise en correspondance de données inter-capteurs.

2. Détection de chaussée

Nous nous proposons d'exploiter le travail effectué précédemment dans l'optique de détecter la chaussée sur laquelle roule le véhicule et de la démarquer dans le flux vidéo.

Cette détection est effectuée sur 3 stratégies :

- Détection du plan routier à l'aide de la transformée de Hough

- Détection des bords routiers
- Acquisition de statistiques sur le plan routier

L'idée envisagée ici est d'utiliser ces trois stratégies pour tracer le profil de la route à l'écran (analyse de scène augmenté). Il s'agit d'une technologie existant depuis quelques années sur les véhicules de série.

a. Détection du plan routier avec la transformée de Hough

Dans un premier temps, nous souhaiterions pouvoir détecter le plan routier par l'intermédiaire des données générées par notre système de capteur. On envisage dans un premier temps de détecter dans un nuage de point le sous-ensemble de points coplanaires à la route (hypothèse : planéité de la route dans un horizon limité).

Nous ne pouvons pas nous appuyer sur les simples coordonnées de ces points en raison du tangage du véhicule et autres mouvements de suspension. C'est pour cela que nous allons utiliser une méthode de détection basée sur la transformée de **Hough** (R Fisher, 2003)**.** La transformée de Hough propose un espace cumulatif permettant de détecter différents types de formes géométriques dans une image via une détection de paramètres associés. Nous utilisons ici cet algorithme pour détecter une droite, qui représente ici la projection du plan de route sur le plan longitudinal du véhicule. Des détails supplémentaires sur la transformée de Hough sont fournies en annexe B.

L'intersection du plan du sol avec le plan longitudinal du véhicule se traduit par une droite. En conséquence, la présence d'un plan dans le nuage de point généré par le télémètre sera traduite par l'apparition d'une accumulation de points sur le plan coupant le véhicule dans le sens longitudinal, formant une droite.

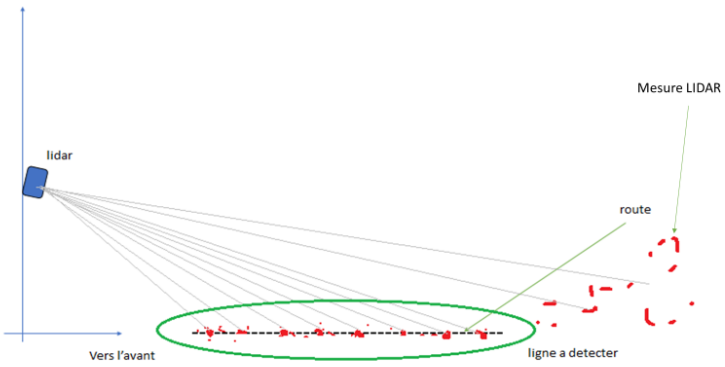

Figure 10 : Illustration d'une méthode de détection de plan via la détection de droite

Si l'on suppose que la route représente le plan principal dans à moins de 20m du véhicule et que celle-ci est plate, il devient possible de détecter la droite associée au plan routier à l'aide de la transformée de Hough.

Par la suite, si l'on souhaite déterminer si tout point du nuage de point LiDAR appartient au plan du sol, il suffit de vérifier si ce point, une fois projeté dans le plan longitudinal, est lui-même arbitrairement proche de la droite détectée. Tout point LiDAR respectant ces conditions peut ensuite être inféré appartient au plan du sol.

Cette détection de plan utilise le protocole suivant :

1) Nous considérons le sous-ensemble de points LIDAR présents dans le champ de la caméra, à la fois par soucis de performance et pour ne détecter que la route faisant face au véhicule. On

élimine également les points dont la distance vis-à-vis du véhicule est trop grande, l'hypothèse d'une route plane ne pouvant tenir à grande distance.

2) Les points retenus sont projetés sur une image monochrome, en fonction de leurs coordonnées sur le plan longitudinal du véhicule. Nous appelons cet espace « espace plan ».
3) En appliquant la transformée de Hough sur cette image, nous retenons la ligne avec le plus grand nombre de votes. Les paramètres (ρ, θ) de la droite correspondante sont utilisés pour calculer les coordonnées d'une paire de points (x_1, y_1) et (x_2, y_2) contenus dans la droite (avec k est un réel réglable, ici k = 1000).

$$\begin{cases} x_1 = \rho.\cos(\theta) - k.\sin(\theta) \\ y_1 = \rho.\sin(\theta) + k.cos(\theta) \\ x_2 = \rho.\cos(\theta) + k.\sin(\theta) \\ y_2 = \rho.\sin(\theta) - k.cos(\theta) \end{cases}$$

La droite est ensuite tracée sur l'image à des fins illustratives, comme aperçu en *Figure 10.*

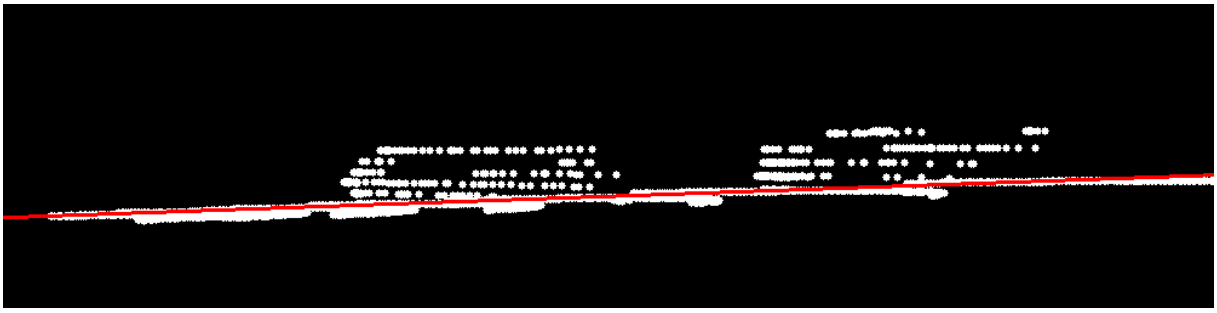

Figure 10: Détection du plan routier via détection de ligne droite. La droite rouge traduit le plan routier détecté.

4) Nous pouvons considérons à nouveau le nuage de point, et nous projetons chaque point sur l'espace plan. Nous évaluons ensuite la distance entre tout point $M\ (x, y)$ de l'image à la droite à l'aide de la relation suivante :

$$D_d(M) = \frac{|(x_2 - x_1)(y_1 - y) - (x_1 - x)(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

Cette distance est comparée à un seuil prédéterminé (ici 5 pixels). Si elle est inférieure à ce seuil, le point LiDAR considéré appartient au plan du sol. Nous pouvons projeter les points LiDAR sur l'image finale si nous le souhaitons.

Cette détection affiche le risque de faussement détecter les trottoirs si ceux-ci sont proches de la route, et risque également de générer des faux positifs si la route n'est que peu visible par les différents capteurs, ce qui motive notre choix de détecter les bords de la route également.

De plus, cette méthode de détection ne fait que chercher des plans dans le nuage de points généré par le télémètre : nous n'avons aucun moyen de nous assurer que ce plan correspond réellement au plan routier et non à une autre surface qui aurait été favorisée.

b. Détection des bords routiers

L'unique détection du plan correspondant au sol n'est parfois pas suffisante pour assurer une bonne détection de la chaussée : en effet, lors de tests nous constatons entre autres que le trottoir est souvent détecté au même titre que le plan routier dans le cas où les bords de la chaussée sont peu marqués. La détection des bords de la chaussée fournit une manière supplémentaire de cerner la chaussée dans l'image générée par la caméra vidéo embarquée.

La méthode utilisée ici présuppose que le véhicule avance de manière relativement parallèle aux bords de la route. Si c'est effectivement le cas, nous devrions détecter dans le nuage de points LiDAR une accumulation de points de coordonnée latérale (x) similaire, correspondant à une interception du faisceau laser du télémètre avec le bord de la route.

Une distribution des coordonnées latérales de l'ensemble des points dans le champ de la caméra fera apparaître des pics dont les coordonnées correspondent à la position latérale des bords de la route. Nous pourrons utiliser ces coordonnées pour tracer dans une paire de lignes démarquant les bords de la route dans l'image finale. La détection de la chaussée ne dépendra ensuite que de notre capacité à détecter l'ensembles des points satisfaisant nos deux stratégies.

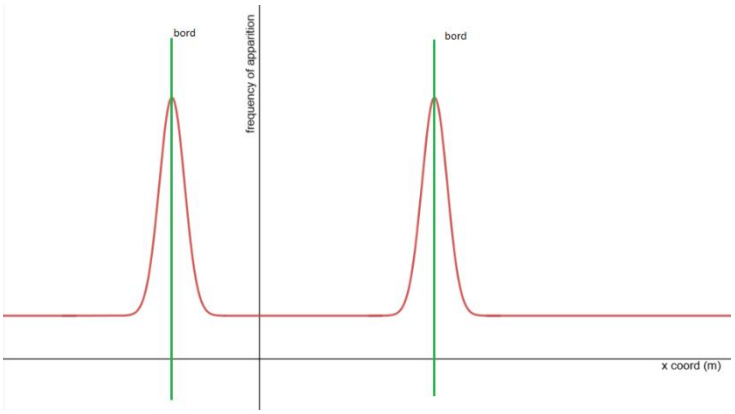

Figure 11:Distribution attendue des coordonnées latérales dans le nuage de point. Les pics correspondent aux bords de la chaussée

Le protocole suivant est mis en place :

1) Dans un premier temps, nous définissons un certain nombre de « bins » (ou seaux) discrets qui servirons à définir la distribution des coordonnées x des points dans le nuage de points, ainsi que la distance maximale (selon x) des points considérés (rien ne sert de scruter les bords routiers à des dizaines de mètres latéralement au véhicule).
2) En ne considérant que le sous ensemble du nuage de point présent dans le champ de vue de la caméra, nous délestons chaque point de leurs coordonnées y et z pour construire la distribution recherchée.
3) En considérant la fréquence d'apparition de points dans chaque bin, nous obtenons la distribution des coordonnées latérales du sous-ensemble du nuage de point considéré. Nous utilisons ensuite une fonction de détection de pics, tels *signal:findpeaks()* de la libraire scientifique *scipy* pour détecter les pics présents et leurs coordonnées
4) Les résultats de cette analyse sont répartis selon trois cas :
    a. Le nombre de pics est strictement inférieur à 2 : nous n'avons pas réussi à détecter des bords routiers et les résultats de l'opération précédente ne sont donc pas exploitables
    b. Le nombre de pics est exactement égal à 2. Ces pics sont donc conservés en l'état et leurs coordonnées latérales sont notées.
    c. Le nombre de pics est strictement supérieur à 2. Dans ce cas nous ne conservons que les pics de part et d'autre de x = 0m dont les coordonnées sont les plus proches de cet

axe central. Nous présupposons ici que la route ne devrait pas engendrer un pic le long de sa surface, et que toute multiplicité de pics de part et d'autre de l'axe centrale est une conséquence de la présence d'objets sur le trottoir. Les coordonnées des pics retenus sont relevées.

5) Nous construisons ensuite pour chaque pic une paire de points dans l'espace (illustré dans la *Figure 12)* de position latérale correspondant au bord détecté. Un premier point est placé à la hauteur du véhicule tandis que le deuxième est placé à une vingtaine de mètres devant celui-ci.

6) Chaque point est ensuite projeté sur l'image couleur de la caméra, puis chaque paire est reliée de façon à faire apparaître le bord routier.

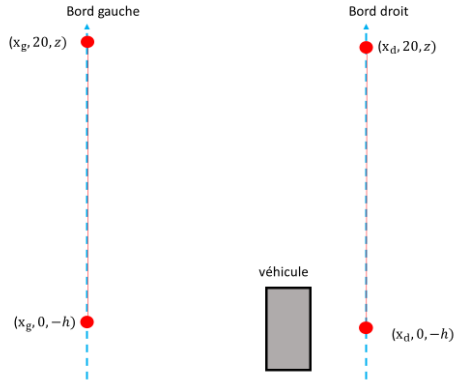

Figure 12: Construction de points conformes au bord routier

Le travail entrepris ci-dessus nous permet de rendre visible les bords routiers. Ce système possède cependant des limitations : comme pour la détection de plan, il est « aveugle » : il n'y a aucun moyen de vérifier que les bords détectés sont réellement les bords de la chaussée.

c. Démarquage heuristique de la chaussée

Le travail entrepris précédemment, nous as permis de cerner la chaussée sur laquelle le véhicule roule (espace navigable). Nous souhaitons la faire apparaître dans le flux vidéo en le surlignant.

La détection de plan et de bords entrepris précédemment exploitait sur le flux de données venant du télémètre LiDAR embarqué. Il est maintenant possible de déterminer si un point LiDAR donné intercepte la chaussé ou non. Nous voulons étendre cette capacité pour pouvoir déterminer si un pixel de l'image, arbitrairement choisi, appartient à la chaussée. Nous proposons le protocole suivant :

1) Nous souhaitons établir des statistiques sur la couleur du plan routier. Nous considérons le sous-ensemble de points LiDAR appartenant au plan routier. Les données valides sont ensuite projetées dans l'image, et nous relevons ensuite la couleur du pixel correspond aux coordonnées obtenues après la projection.
2) En considérant les données récoltées, nous calculons la moyenne et l'écart type associée à chaque composante (bleu, vert et rouge) couleur.
3) Nous considérons ensuite l'ensemble des pixels susceptibles d'appartenir au plan routier. La couleur de chaque pixel est relevée et comparée aux statistiques récoltés précédemment. Si la couleur obtenue est arbitrairement proche des statistiques récoltées nous plaçons un pixel de couleur en ces coordonnées pour démarquer ce que l'on infère être la chaussée.

Cette technique est pour le moment la seule considérée dans l'objectif de surligner la route dans le flux vidéo. Cependant, elle souffre de certains problèmes impactant sa fidélité. Fondamentalement, cette technique ne voit *que* la couleur des pixels. Elle est donc capable de démarquer des pixels n'appartenant pas à la route bien que de couleur similaire (roues de voitures, murs…) ou ne pas

démarquer des objets qui appartiennent au plan routier sans pour autant être de la même couleurs (les marquages routiers par exemple). La combinaison de ces trois critères de détection (détection de plan, de bords, de couleur) nous permet de développer un bloc logiciel capable de démarquer la route dans l'image couleur générée par la caméra embarquée.

3. Démarquage d'obstacles dans l'espace 3D-LiDAR

La mise en commun de données explorée au cours des deux dernières sous-parties représentait une transposition des données générées par le télémètre LiDAR dans l'espace image de la caméra RGBD implantée. Nous devons maintenant mettre en œuvre un exemple de corrélation dans l'autre direction : à savoir une transposition des données générées par la caméra dans l'espace LiDAR.

Nous nous proposons ici d'utiliser un algorithme de vision par ordinateur pour repérer les usagers de la route présente dans le champ de vue de la caméra, pour ensuite démarquer ces obstacles dans un environnement 3D centré sur le télémètre. De plus, nous voulons que les amers que nous plaçons reflètent la nature et l'orientation des utilisateurs détectés.

a. L'espace 3D LIDAR

Les exemples précédents étaient tous capables de nous fournir en sortie une image couleur contenant l'ensemble de la télémétrie du système de capteurs. Ceci était une conséquence naturelle du sens de parcours des informations (du télémètre vers la caméra). La mise en correspondance ayant lieu dans le sens inverse ici, le bloc logiciel que nous mettrons en place fournira en sortie un nuage de point constitué de points de coordonnées 3D, ainsi qu'une série d'amers 3D destinés à être dessinés dans ce même espace. Cette télémétrie doit être visionnée à l'aide d'un logiciel spécialisé, nommé *RVIZ (visible en Figure 13).*

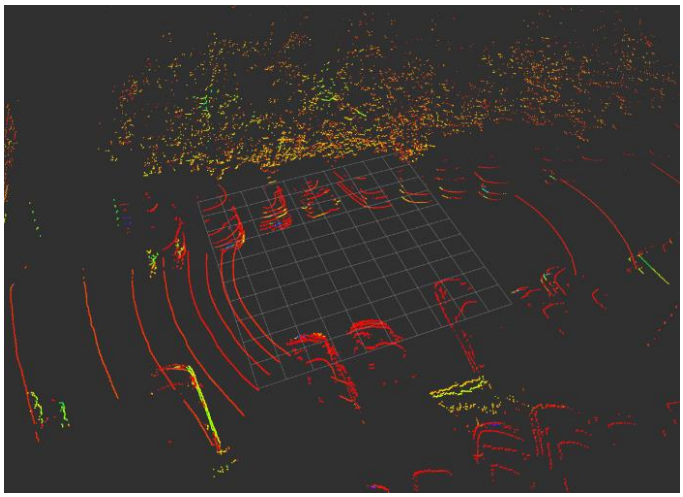

Figure 13: Visualisation de nuages de points avec RVIZ

Les amers en question sont des formes géométriques simples de taille, de position et d'orientation à définir que nous placerons dans l'espace propre au LIDAR après avoir déterminé leurs coordonnées dans ce repère.

b. Détection d'objets automatisée par CNN

Le travail envisagé ici repose sur notre capacité à distinguer différents types d'obstacles dans l'image générée par la caméra. Nous utiliserons un système de vision nommé *MobileNet*, qui exploite l'intelligence artificielle. MobileNet utilise un réseau de neurones convolutionnel pour détecter différentes classes d'objets dans une image.

Figure 14: Détection d'objets avec MobileNet

Les types d'objets à détecter sont définis avant l'implantation, via un procédé nommé *apprentissage profond*. Lors de ce procédé, MobileNet à accès à une large banque d'images sur lesquelles figurent des objets à détecter, dans le but d'apprendre au système de vision l'apparence physique des objets que l'on cherche à détecter. Une fois l'apprentissage fini (c'est l'état du système tel qu'on le reçoit en début de stage), MobileNet est ainsi capable de détecter ces objets dans une image. Ces objets sont démarqués à l'aide d'une boite englobante (exemple illustré dans la *Figure 14),* dont les coordonnées sont rendues disponibles en conjonction avec une variété d'informations annexes.

c. Projection dans l'espace LiDAR

Les exemples précédents touchaient à la projection de données LiDAR dans l'espace image de la caméra RGBD. Cette projection était effectuée en transposant les points du repère 3D LiDAR dans le repère 3D de la caméra à l'aide d'une matrice ${}^{U}[S]_{V}$, suivi d'une projection de ces points sur l'image finale à l'aide d'une matrice $[K]$. Nous voulons réaliser l'opération inverse : transposer des points de l'image vers l'espace 3D caméra, puis vers l'espace 3D (repère) LiDAR.

Il est facile de réaliser la transformation inverse ${}^{U}[S]_{V}$, notée ${}^{V}[S]_{U}$:

$$ {}^{V}[S]_{U} = \begin{pmatrix} R_{3x3}^{T} & -R_{3x3}^{T} * t_{3x1} \\ 0_{1x3} & 1 \end{pmatrix} $$

Cette matrice est capable de transposer les points de l'espace 3D (repère) caméra dans l'espace LiDAR. Elle utilise les mêmes paramètres extrinsèques que la matrice permettant de transposer les données LiDAR dans l'espace caméra.

Il ne nous reste plus qu'à projeter les points de l'image dans l'espace 3D de la caméra. Lors du chapitre 3A, nous avions évoqué que la projection d'informations sur l'image représente une destruction d'informations. En effet, le passage d'un espace de dimension 3 vers un espace de dimension 2 représente une perte d'informations : il existe théoriquement une infinité de points de l'espace 3D qui se projettent sur un unique point de l'image.

Concrètement, la projection de points sur l'image entraîne une perte de l'information de profondeur associés à ces points. L'opération inverse est donc capable de définir une droite sur laquelle se trouve le point original, mais ne peut pas spécifier la position du point sur cette droite (la distance du point à

la caméra). La mesure de profondeur effectuée par la caméra RGBD nous laisse cependant retrouver cette coordonnée manquante.

La transformation image-espace 3D caméra est réalisée à l'aide des paramètres intrinsèques de la caméra. Dans un premier temps nous définissons un vecteur normé pointant dans la direction du point dans l'espace à l'aide des coordonnées (u, v) du pixel considéré.

$$[Obj] = normalize\left(\begin{bmatrix} \frac{u}{f_x} \\ \frac{v}{f_y} \\ 1 \end{bmatrix}\right)$$

Les coordonnées dans l'espace 3D caméra sont ensuite retrouvées à l'aide de la mesure de profondeur RGBD :

$$P_{3D_{cam}} = distance * [Obj]$$

On peut ensuite retrouver les coordonnées de ce point dans l'espace LIDAR :

$$P_{3D_{LiDAR}} = \begin{pmatrix} R_{3x3}^T & -R_{3x3}^T * T_{3x1} \\ 0_{1x3} & 1 \end{pmatrix} * P_{3D_{cam}}$$

Où R, T sont les matrices rotations et translation définies au chapitre 3.

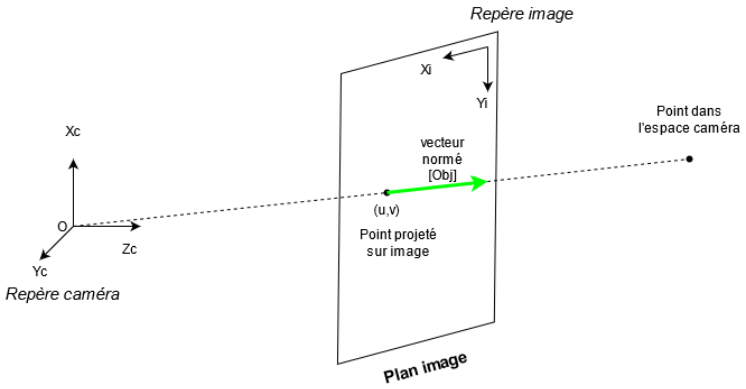

Figure 15: Projection inverse image-camera

d. Exploitation

Le principe opératoire visé est le suivant : lorsque MobileNet détecte des objets dans une image, nous récupérerons les coordonnées (sur l'image) des boites englobantes des objets détectés En considérant uniquement le centre de cette boite englobante, nous calculerons les coordonnées de ce point dans l'espace LIDAR. Ensuite, selon le type d'objet en question, nous placerons un amer spécifique en ces coordonnées. De plus, si l'objet est un véhicule, nous tenterons d'estimer son orientation avant de placer l'amer correspondant.

L'ensemble des amers générés par cette opération sera ensuite publié pour être affiché par RVIZ, en conjonction avec le nuage de points généré par le télémètre.

L'estimation de l'orientation des véhicules détectés est entièrement basée sur la forme des boites englobantes. Nous considérerons le rapport entre la hauteur de la boite englobante et sa largeur comme indice sur l'orientation du véhicule à l'écran. Nous constatons que les véhicules faisant face (ou dos) à la caméra admettent des boites englobantes où ce rapport est d'environ 1.0 (des boites de forme carrée) tandis que les véhicules placés avec leurs côtés face à la caméra admettent des boites englobantes où ce rapport est entre 2 ou 3, comme nous l'illustrons en *Figure 16*.

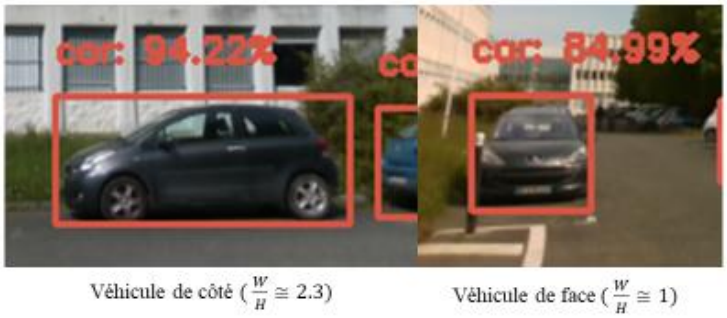

Figure 16: Véhicules détectés sous différents angles

Ce rapport est borné, ne dépassant guère une valeur de 3 (pour les véhicules les plus longs), et ne passant pas sous 0.8 (les véhicules les plus fins). En considérant un angle de 0° comme étant l'angle correspondant à un véhicule faisant dos à la caméra (auquel cas le rapport W/H sera faible), et 90° correspondant à un véhicule lui montrant son côté (auquel cas le rapport W/H sera grand), nous pouvons construire un mécanisme associant les dimensions d'une boite englobante à l'angle du véhicule.

Par symétrie des amers, on se restreint à l'intervalle [0°, 90°], ce qui simplifie le problème considéré. Nous appliquons donc le protocole suivant :

1) Soit $WH_{max}\ et\ WH_{min}$ les bornes supérieurs et inférieurs attendues pour le rapport des dimensions des boites englobantes. Nous considérons une première boite englobante et calculons le rapport entre sa largeur et sa hauteur. Si elle n'est pas comprise dans l'intervalle fixée par ces deux bornes (trop élevée ou trop faible), nous la ramenons à la valeur autorisée la plus proche.
2) Nous normalisons ce rapport entre 0 et 1 à l'aide de la formule suivante :
$$WH_{norm} = \frac{WH - WH_{min}}{WH_{max} - WH_{min}}$$
3) L'angle final est obtenu : $\theta = WH_{norm} * 90°$

L'orientation des amers selon un unique axe (l'axe z) est suffisante pour l'application envisagée ici. Cette information, si elle est calculée (elle ne l'est pas pour les piétons, où nous utiliserons un amer cylindrique) est ensuite utilisée, en conjonction avec la position dans l'espace de l'objet détecté, pour placer un amer dans le repère du LiDAR. Cette opération est ensuite effectuée pour l'ensemble des objets détectés par MobileNet.

4. Résultats

Les principes explicités plus haut sont mis en application pour créer différents blocs logiciels utilisant Python. Les retombées de ces travaux, qu'ils soient logiciels ou autres, seront explicités ici.

a. Projection LIDAR-image en temps réel

L'application visant à reprojeter les données LIDAR sur l'image générée par la caméra embarquée exploitant bon nombre d'outils développés dans le cadre du travail abordé dans le cadre du chapitre III, elle est la première à être implantée sur le véhicule.

Cette application est capable de manifester sur le flux vidéo les mesures effectuées par le télémètre, tout en faisant apparaître la distance de chaque point à l'aide d'un code couleur (bleu signifiant une faible distance et rouge signifiant une distance plus élevée). Plusieurs essais sont réalisés sur le plateau du Moulon durant lesquels des enregistrements sont effectués.

Figure 17: Projection en temps réel des données LIDAR

La qualité de la projection est à la hauteur de nos attentes. Cependant, la lenteur associée au langage Python, couplée au grand nombre d'opérations à effectuer, limite la fréquence de rafraîchissement du flux vidéo final (image + points reprojetés). Une perspective possible serait d'implémenter ce logiciel sous une langage plus rapide, tel que C++. L'augmentation de la fréquence d'acquisition du télémètre pourrait également être envisagée afin d'améliorer la qualité de la reprojection, à supposer une puissance de calcul suffisante.

b. Projection camera-LiDAR

Le bloc logiciel assurant la projection de données dans l'espace LIDAR à l'aide de MobileNet est le deuxième bloc logiciel à être implanté. Ce bloc ne possède initialement pas la capacité d'estimer l'orientation des véhicules détectés par notre système de vision. Cette capacité sera ajoutée ultérieurement.

Des essais sont effectués pour constater du bon fonctionnement du bloc logiciel en question. Nous notons plusieurs défauts avec ce bloc logiciel :

- La détection des objets n'est pas parfaitement assurée : certains usagers ne sont pas détectés.

- L'estimation de la profondeur des objets détectés reste perfectible, générant des erreurs quant au placement des amers. C'est pour cela que nous n'affichons que les amers proches du véhicule (moins de 30m).
- Le placement des boites englobantes n'est pas toujours idéal : MobileNet détecte parfois difficilement l'extrémité de certains véhicules, faussant l'orientation de l'amer.

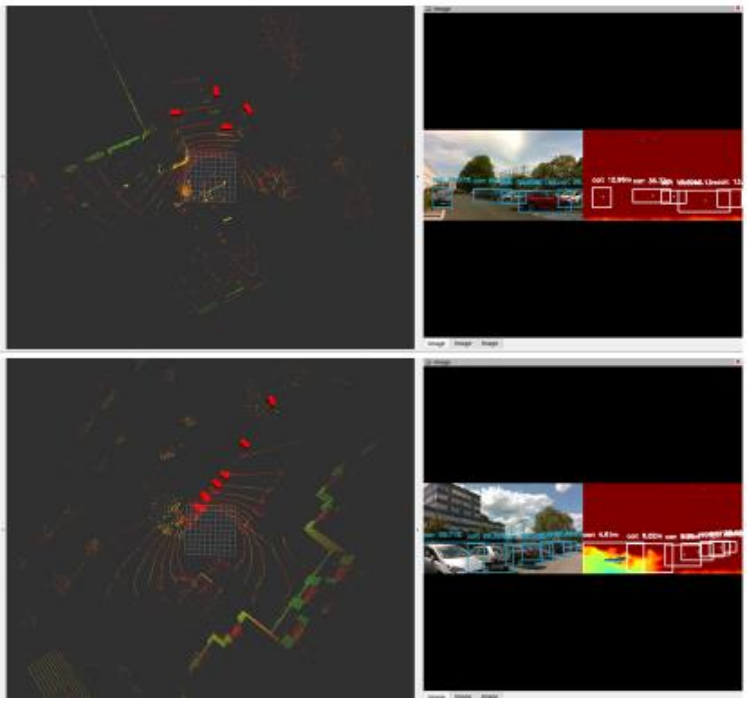

Figure 18: exemples de placement d'amers dans l'espace 3D LIDAR (gauche), détection d'objets MobileNet (droite)

Bien que les résultats de cette partie-mission sont globalement satisfaisants, nous jugeons que l'implémentation de MobileNet utilisée ici limite la qualité de la reprojection que nous tentons d'accomplir. Une meilleure détection des objets dans une scène et une meilleure estimation de leur distance serait donc souhaitable. La fréquence de rafraîchissement offerte n'est cependant pas limitante. L'évaluation qualitative des prédictions émises par MobileNet est possible à l'aide d'un data-set de référence, tel que des prédictions émises par un algorithme de vision suppos plus fiable. A l'aide d'une telle référence, les prédictions de MobileNet peuvent être évaluées grâce à un indicateur nommé Intersection Over Union (le rapport des surfaces d'intersection et d'union des boites englobantes pour MobileNet et la référence). Ce travail, débuté en cours de projet tutoré mais non poursuivi en stage, permettrai d'évaluer le taux de correspondance entre MobileNet et la référence choisie.

c. Retombées : Challenge UTAC 2022

L'UTAC, un organisme de validation de véhicules, en coopération avec la Société des Ingénieurs de l'Automobile, organise en 2022 la première édition de son concours, nommé « UTAC Challenge ». Ce concours, destiné aux universités et écoles d'ingénieurs, permet aux établissements participants de présenter des projets de véhicules automatisés réalisés par des groupes d'étudiants. Le concours présente différentes catégories distinctes (Parcours Urbain Automatisé, Parking Automatisé et Épreuve Libre) dans lesquels il est possible de participer.

Figure 19: présentation du projet devant le jury (c'est moi !)

Au cours du concours, les participants présentent leur projet (qui peut être un véhicule physique ou une simulation de véhicule) dans le cadre d'un « pitch » oral court, suivi d'une démonstration

technique durant laquelle l'équipe peut apporter des précisions sur le travail abordé. Le jury présent est constitué d'industriels travaillant dans le secteur automobile.

C'est dans le cadre de ce concours que l'IUT d'Orsay décide de présenter la plateforme de perception sur laquelle nous travaillons. Le véhicule n'étant pas robotisé, l'IUT postule dans la catégorie « Épreuve Libre ». Le challenge accueille une variété d'écoles réparties sur l'ensemble des catégories disponibles : ESTACA, ESIGELEC, ENSTA Paris, ENSTA Bretagne, IPSA, UTBM, ECE, ESME et Polytech Orléans.

Au cours de ce concours, nous avons présenté la somme totale du travail réalisé en stage jusqu'au jour de l'évènement (le 12 mai), qui comprenant les sections **a.** et **b.** de cette sous partie ($IV.D$), ainsi que le travail réalisé par le groupe « Perception Mobile » de Projet Tutoré cette année, qui a permis d'introduire certaines notions techniques nécessaires au travail réalisé en stage.

Figure 20: L'équipe de l'IUT d'Orsay présente au concours UTAC après notre victoire

C'est dans le cadre de ce concours que l'IUT d'Orsay remporte la première place dans la catégorie « Épreuve Libre ». Cette victoire permet à l'IUT de remporter un trophée ainsi que des places pour différents évènements automobiles.

d. Détection du plan de route

La détection de la route est une application effectuée en plusieurs temps. Dans un premier temps, pour démontrer le concept, nous développons un bloc logiciel dont le seul objectif est de détecter le sol, pour ensuite projeter sur l'image les données LiDAR qui interceptent ce plan. Ce bloc est mis en place afin d'évaluer les capacités d'un tel système.

Ce bloc logiciel est déployé sur le véhicule. Ce premier prototype nous permet de voir que la technique utilisée reste puissante à condition que les hypothèses émises lors de sa conception sont respectées. Cependant, il reste très sensible à ces hypothèses. Notre choix de contraindre la détection de la route via une détection des bords de la chaussée était donc justifiée.

Figure 21: Détection du sol via transformée de Hough (les points colorés correspondent à des données LIDAR).

Une méthode alternative aurait été d'utiliser des algorithmes de détection de plan. Cependant, cette approche est jugée surdimensionnée et n'est pas exploitée dans le cadre de ce stage. L'implantation d'un algorithme de détection de sol utilisant une détection de plan est donc une perspective d'évolution possible.

Ce bloc représente une preuve de concept et nous fournis assez d'assurance pour continuer à exploiter cette stratégie de perception.

Nous nous mettons au travail pour modifier le bloc précédent, via l'ajout de code capable de détecter les bords de la route, ainsi que de code capable de surligner cette route, une fois détectée.

Nous constatons que ce bloc est également sensible aux hypothèses émises, fournissant une bonne démarcation de la chaussée lorsque celle-ci est bien visible et avec des bords bien démarquées, tel qu'on le constate dans les clichés a) et b) de la *Figure 22*. Cependant, comme nous l'avions anticipé, nous constatons que l'analyse routière basée sur la couleur n'est parfois pas capable de détecter certaines parties de la route lorsque celles-ci sont de couleur trop différente.

Les marquages routiers visibles dans a) et b) ne sont entre autres pas surlignés. L'ombrage sur la route visible dans l'image c) de la Figure 22 entraîne une non-détection du reste de la route non-ombragée, sa couleur étant trop différente. La récolte de statistiques dans un format couleur plus sensible à la luminosité de chaque pixel (tel que YCbCr) pourrait permettre une récole de statistiques plus adaptées, mais ce travail n'est pas entrepris pour le moment.

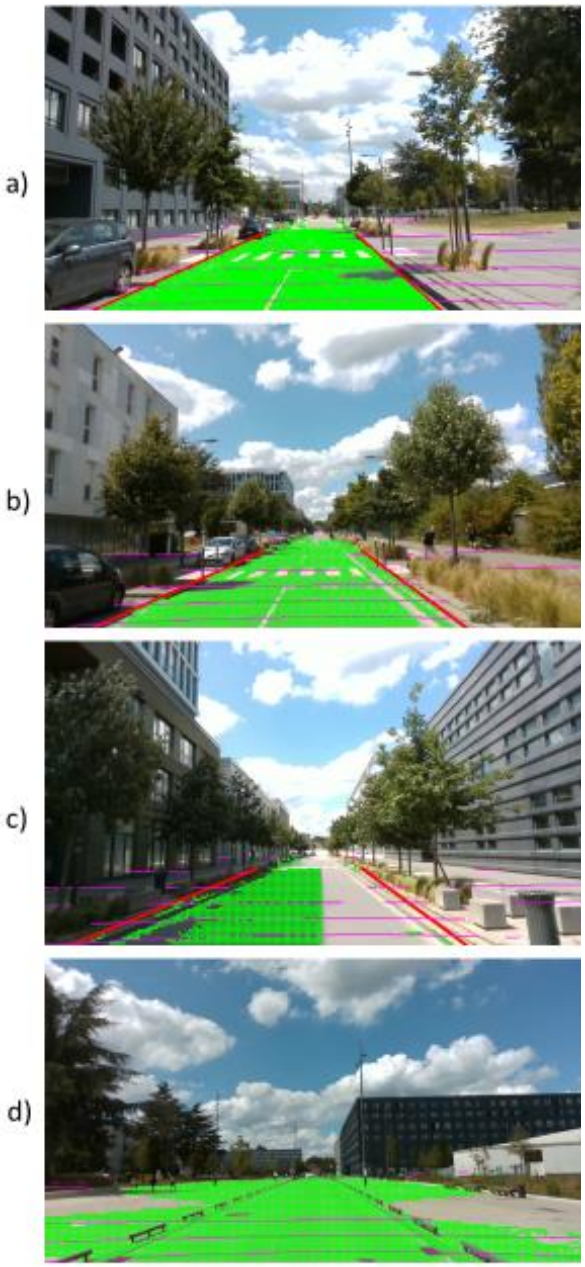

Figure 22: Détection et marquage de la route (VERT = Route détectée, ROUGE = bords de la route détectés, VIOLET = mesures LIDAR)

Dans certains cas, les bords de la route ne sont pas détectés, comme dans le cliché d) de la Figure 22. Lorsque c'est le cas, le logiciel bascule vers une simple détection de sol, qui inclut les trottoirs bas. Un facteur limitant pour cette détection est la résolution de notre télémètre : une résolution accrue permettant une plus grande accumulation de points sur les bords routiers.

Une potentielle piste d'amélioration à considérer est de cerner la route à l'aide de mécanismes supplémentaires : nous pourrions par exemple concevoir d'un système exploitant plusieurs stratégies de détection routières, où la détection de la route nécessiterai qu'une majorité de ces stratégies rapportent des résultats positifs et cohérents. L'évaluation de la qualité des résultats à l'aide d'une

matrice de confusion (notamment à l'aide d'un data-set de référence permettant de vérifier la conformité des résultats émis par ce bloc) est une perspective d'avenir possible, mais qui n'est pas abordée dans le cadre de ce stage.

e. Retombées : interface graphique de commande

C'est dans l'optique de simplifier les commandes du système de perception que notre tuteur, Mr. Rodriguez, développe une interface graphique simple à partir de laquelle nous pouvons lancer les différents blocs de perception implantés sur le véhicule. Au début du stage, cette interface contient des commandes pouvant activer les différents capteurs du système, lancer un enregistrement, montrer l'état du système, et même activer le détecteur d'objet MobileNet.

Au cours de notre stage, nous avons modifié cette interface afin qu'elle puisse commander les blocs de perception que nous avons développé. L'évolution de l'interface est visible en constatant les changements entre les deux interfaces de la *Figure 23* :

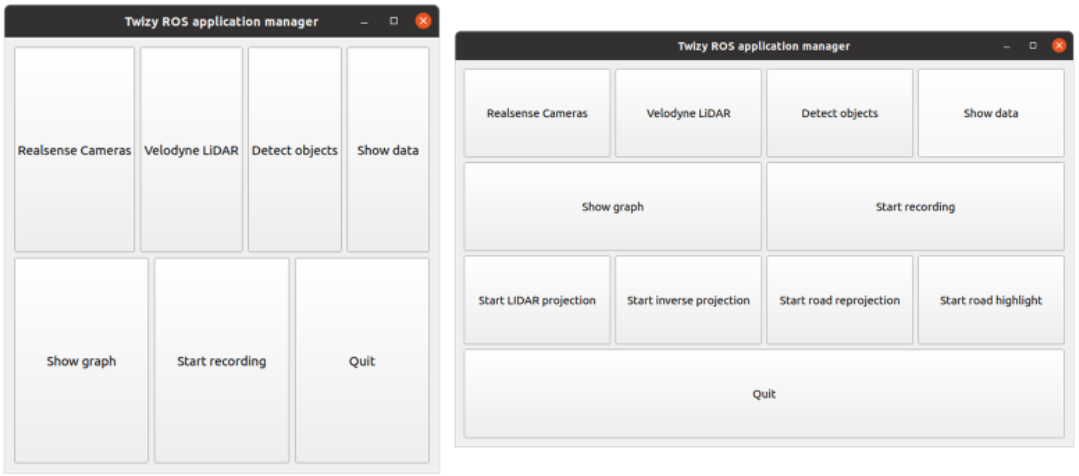

Figure 23: Evolution de l'interface de commande du système de perception (GAUCHE : en avril 2022, DROITE : en juin 2022)

Cette interface permet à l'utilisateur de lancer simplement les différents blocs de perception via une interface graphique facilement navigable. Ceci permet de s'affranchir de la complexité associée aux invités de commande.

V. Conclusion

L'objectif principal qui nous a été confié en avril 2022 était d'instrumenter un système de perception multi-capteurs, en vue d'implanter sur la plateforme une série de blocs logiciels exploitant des données provenant de sources multiples.

Bien que les résultats de ces blocs de perception restent perfectibles, nous jugeons qu'ils représentent une preuve de concept démontrant la faisabilité des différentes techniques méthodes explicitées au cours de ce document. De plus, nous jugeons que dans son état actuel, le système de perception implanté sur le véhicule apporte des informations qui pourraient s'avérer utiles à un conducteur averti. Nous citons par exemple le bloc logiciel de reprojection LiDAR, qui pourrait complémenter la vue du conducteur dans des mauvaises conditions lumineuses.

Nous jugeons aujourd'hui que les objectifs visés sont atteints : les différents blocs de perceptions développés au cours du stage démontrent une mise en commun de données provenant des différents capteurs installés sur le véhicule. De plus, cette mise en correspondance est assurée dans l'ensemble des directions possibles, démontrant une réelle mise en correspondance entre les capteurs du système.

VI. Bilan

1. Bilan personnel

Cette première expérience professionnelle, bien qu'ayant lieu dans l'IUT où j'effectue ma formation, a représenté pour moi un changement significatif par rapport à mon parcours jusqu'à là de nature purement scolaire. Ceci a provoqué une certaine inquiétude de ma part en début de stage.

Cependant, je qualifierai très positivement cette première expérience : le fait de pouvoir travailler plus ou moins en autonomie sur un projet à la fois intéressant et relativement complexe est très valorisant à mes yeux, et représente une des raisons principales motivant mon choix de continuer dans la voie de l'ingénierie. Ce stage m'as donc permis de me conforter dans mon choix de poursuite d'études et de projet professionnel.

2. Bilan professionnel

Le travail entrepris au cours de ce stage m'as naturellement permis d'acquérir de nouvelles compétences et connaissances qui me seront utiles à l'avenir.

L'utilisation de Python pour le développement de blocs logiciels m'as offert une meilleure maîtrise de ce langage ainsi qu'une meilleure connaissance de principes de programmation que j'avais jusqu'à présent ignorés, tel que la programmation orienté objet. Le développement de logiciels incorporant des centaines de lignes de code m'a également forcé à revoir ma façon d'organiser et de développer mon code.

Ma vitesse de frappe sur clavier semble également avoir légèrement augmentée.

Au cours du stage j'ai pu observer de plus près les techniques utilisées par l'industrie automobile dans les systèmes ADAS contemporains dans l'optique de les implémenter sur le véhicule de l'IUT. Ceci représente pour moi mon aspect préféré de ce sujet de stage : le travail entrepris ici n'est ni obsolète ni inutile et pourra éventuellement être appliqué dans ma carrière future.

VII. Bibliographie :

Ressources :

- [P. Corke, 2011] : « *Robotics, Vision and Control : fundamental algorithms in MATLAB* »
- [J. Orteu, 2008] : « *Modélisation et calibrage d'une caméra* » (*http://www.optique-ingenieur.org/fr/cours/OPI_fr_M04_C01/co/Grain_OPI_fr_M04_C01_2.html*)
- [INTEL] : « *Projection, Texture-Mapping and Occlusion with Intel RealSense Depth Cameras* »

- [Wikipedia] : « Rotation Matrix » (https://en.wikipedia.org/wiki/Rotation_matrix)
- [Wikipedia] : « Hough transform » (https://en.wikipedia.org/wiki/Hough_transform)
- [S. Nayar, 2021] : « *First principles of computer vision : Hough Transform | Boundary detection » (Youtube)*
- [R. Fisher, S. Perkins, A. Walker, E Wolfart, 2003]: « Hough Transform » (https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm)

Documentation :

- https://docs.opencv.org/4.x/dc/da5/tutorial_py_drawing_functions.html
- https://docs.python.org/3/library/tkinter.html
- https://realpython.com/python-gui-tkinter/
- https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html
- http://wiki.ros.org/rviz/DisplayTypes/Marker

Images et ressources graphiques :

- Logo de l'IUT d'Orsay : https://www.iut-orsay.universite-paris-saclay.fr/
- Interview challenge UTAC : https://www.youtube.com/watch?v=B9J_mRO_NnA

VIII. Annexes

1. Les systèmes de coordonnées homogènes

L'expression des coordonnées d'un point dans un espace de dimension n est classiquement effectué à l'aide d'un n-uplet de scalaires, représentant en agrégat les coordonnées du point. Par exemple, nous pouvons exprimer les coordonnées d'un point dans un espace 3D (associé à un repère constitué de 3 vecteurs non colinéaires) à l'aide d'un triplet de nombres. Nous allons cependant montrer que ce système n'est pas adapté pour une manipulation efficace des points de l'espace.

Ce système de coordonnées est adapté lorsque l'on souhaite transformer ce point par rotation. Soit $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ et $X' = \begin{pmatrix} x_1{}' \\ \vdots \\ x_n{}' \end{pmatrix}$ les coordonnées d'un point dans un espace de dimension avant et après une rotation par le vecteur $R = \begin{pmatrix} r_{1,1} & \dots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{n,1} & \dots & r_{n,n} \end{pmatrix}$.

L'opération effectué est :

$$X' = RX$$

Cette opération est possible car X est de même hauteur que la largeur de R.

Considérons maintenant une opération de translation $T = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$ dans ce même espace. Soit $X" = \begin{pmatrix} x_1" \\ \vdots \\ x_n" \end{pmatrix}$ les coordonnées du point considéré après translation et rotation. Les coordonnées du point sont donc obtenues avec l'opération suivante :

$$X" = X' + T$$

$$X" = RX + T$$

Ce calcul nécessite d'effectuer deux opérations matricielles. Nous pouvons cependant combiner les opérations les opérations de rotation et de translation en construisant une nouvelle matrice $[B]$:

$$B = \begin{pmatrix} r_{1,1} & \dots & r_{1,n} & t_1 \\ \vdots & \ddots & \vdots & \vdots \\ r_{n,1} & \dots & r_{n,n} & t_n \end{pmatrix}$$

Remarquons que l'opération :

$$X' = BX$$

…est maintenant impossible en raison de dimensions incompatibles de B et de X. Pour qu'elle soit possible, il est nécessaire de modifier le vecteur X en y concaténant une ligne de compensation. On peut par exemple noter Y ce vecteur :

$$Y = \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix}$$

Il nous est maintenant possible de multiplier B par Y pour obtenir les coordonnées du point X".

Il est cependant possible de trouver une alternative au fait de concaténer systématiquement un 1 à chaque vecteur colonne que l'on souhaite transformer. C'est dans cette optique que l'on introduit le **système de coordonnées homogènes** (l'expression de Y ci-dessus est d'ailleurs homogène, illustrant une manière d'exprimer des coordonnées sous cette forme). Dans ce système, chaque point est exprimé à l'aide d'un vecteur colonne constituée de $n + 1$ lignes. L'opération rotation + translation peut donc s'exprimer de façon compacte à l'aide de la matrice suivante :

$$B' = \begin{pmatrix} r_{1,1} & \cdots & r_{1,n} & t_1 \\ \vdots & \ddots & \vdots & \vdots \\ r_{n,1} & \cdots & r_{n,n} & t_n \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}$$

L'opération suivante fournit donc un vecteur colonne Y" de coordonnées homogènes.

$$Y" = B'Y$$

Il est ensuite possible de retrouver les coordonnées « classiques » associées au point Y (que l'on notera U) :

$$U = (u_{i,j})_{1 \le i \le n\ ,j=1} \rightarrow u_{i,j} = \frac{x_i}{x_{n+1}}$$

Ce système de coordonnées sera exploité pour l'ensemble des opérations matricielles utilisées.

2. Transformée de Hough

La transformée de Hough est un algorithme de détection de droites. Cet algorithme, appliqué à une image constituée de pixels discrets, et capable de détecter les droites présentes dans l'image.

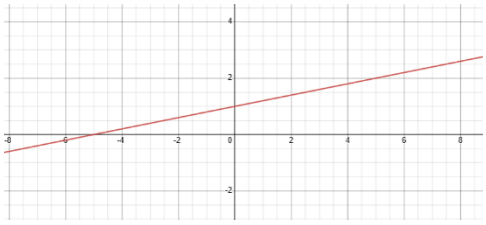

Figure 24: Droite affine y = .2x +1

Pour illustrer son mode de fonctionnement, considérons la fonction affine tracée en *Figure 24*. Son expression est de forme :

$$y = mx + b$$

Nous souhaitons déterminer expérimentalement les valeurs de m et b.

Considérons ensuite un point m de coordonnées (x_0, y_0) appartenant à la droite tracée par cette fonction. Nous prendrons comme exemple le point de coordonnées (-2, 0.6). Nous pouvons réarranger l'expression de notre fonction tel que :

$$b = y_0 - mx_0$$

Ceci est maintenant une fonction de variable m. Il est possible de tracer cette fonction au même titre que notre droite affine (visible en *Figure 25)* sur un plan appelé *espace paramétrique.* Nous obtenons une droite : il existe après tout une infinité de droites traversant le point (x_0, y_0).

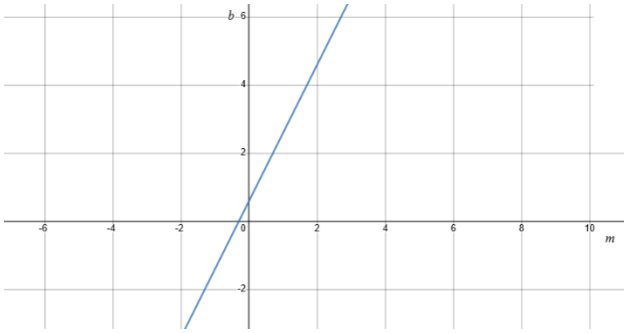

Figure 25: traçage de la droite b = y0 -mx0

Choisissons maintenant un autre point de la droite affine de coordonnées (x_1, y_1), tel que $(2, 1.4)$. Remarquons ce qu'il se produit lorsque nous utilisons ces coordonnées pour tracer la droite paramétrique associé à ce point (voir *Figure 26).*

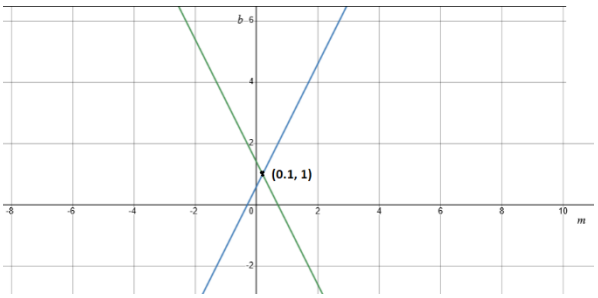

Figure 26: Traçage des droites paramétriques associées au deux points sélectionnés

Nous remarquons un point d'intersection entre les deux droites, correspondant aux couples de paramètres (b, m) satisfaisant les deux points que nous avons choisis. Ce sont les paramètres recherchés !

La généralisation de ce procédé à l'ensemble des points de la droite fera apparaître un unique point d'intersection indiquant les paramètres de la droite passant par l'ensemble des points. Sur une image constituée de pixels, contenant plusieurs lignes, nous remarquerons une multitude de points d'intersection de droites paramétriques, chacune correspondant à une droite utilisant approximativement ces paramètres.

L'approche ci-dessus pose cependant le problème que la pente de la droite considérée est à priori non bornée, pouvant prendre une infinité de valeurs. Une implémentation pratique de la transforme de Hough utilisera donc plutôt une autre paramétrisation de la droite sous forme polaire :

$$x.\sin(\theta) - ycos(\theta) + \rho = 0$$

Où ρ, θ sont les paramètres de la droite. θ est ici astreint à l'intervalle $[0, \pi]$ contrairement à m. Ceci provoque un changement majeur : tandis que les couples (b, m) de paramètres décrivant une droite passant par un point donné forment une droite dans l'espace paramétrique, les paramètres (ρ, θ) sont quant à eux distribué sur une sinusoïde. Le principe reste cependant exactement le même que précédemment et nous permet de déterminer le couple de paramètres (ρ, θ) associés à chaque droite.

Une implémentation pratique de la transformée de Hough utilise un système de votes pour déterminer les couples de paramètres les plus utilisés de façon plus précise que « trouver un pseudo-point d'intersection entre les droites paramétriques ». Pour mettre un tel système en place, on conçoit un tableau de valeurs nommé *accumulateur* qui discrétise l'espace paramétrique. En bref, des sous-ensembles de couples (ρ, θ) proches sont groupés par « bins ».

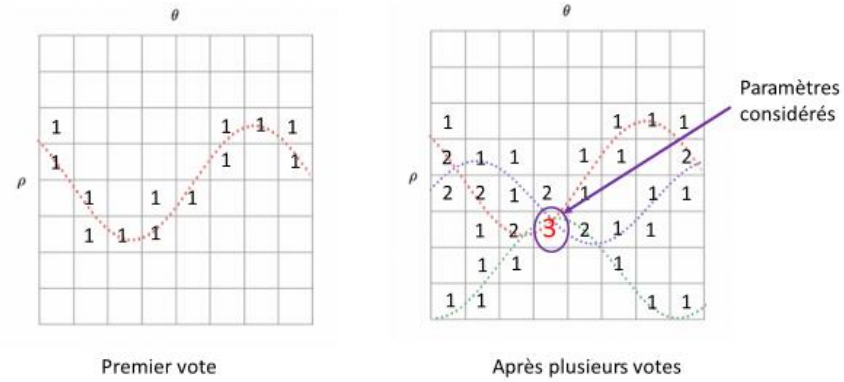

Figure 27: Procédure de vote dans l'accumulateur. Les nombres dans les cases correspondent au nombre de votes dans chaque bin, où le nombre de sinusoïdes intersectant cette case. Le bin avec le plus grand nombre de votes correspond à un couple de paramètres satisfaisant une grande quantité de points et décrivent à priori une droite dans l'image d'origine.

Lorsqu'un point (ex : un pixel d'une image) est considéré et que la droite paramétrique est obtenue, un vote est ajouté dans chaque bin contenant des paramètres possibles. Une fois avoir considéré l'ensemble des points de l'image, nous pouvons consulter les bins contenant un grand nombre de votes pour en extraire des paramètres de droites plausibles dans l'image.

L'implémentation de la transformée de Hough sous OpenCV (utilisé pour nos travaux) retourne l'ensemble des couples de paramètres ayant obtenus un nombre de votes supérieur à un seuil prédéterminé.