GRILLE D'ACCUMULATION 2D POUR VEHICULE AUTONOME

Quentin SCHULTZ, Mathieu GROSSARD
Professeur encadrant : Thomas JOSSO-LAURAIN
ENSISA Mulhouse 12 Rue des Frères Lumière, 68093 Mulhouse
prenom.nom@uha.fr

Résumé—**Nous avons développé un algorithme pour le traitement de données 3D provenant d'un lidar afin de les convertir et de créer une grille 2D d'accumulation. Celle-ci affichant une couleur en fonction de la présence ou non d'un obstacle et de la proximité du véhicule par rapport à celui-ci. Pour cela, nous avons utilisé la base de données Kitti qui possède des jeux de données lidar. Dans le but de pouvoir développer notre algorithme, dans un premier temps sur Matlab pour commencer à manipuler et à appréhender le sujet puis sur RTMaps pour l'utilisation finale. Nous avons pu le tester sur le véhicule à l'ENSISA. Notre algorithme est capable de traiter les données lidar et permet la détection des obstacles par le véhicule. Il permet également de sélectionner les obstacles présentant une proximité pouvant être considérée comme dangereuse. Cet algorithme a pour vocation d'être utilisé pour des usages plus courants du véhicule (conduite en ville ou campagne par exemple) et l'appliquer à d'autres contextes de conduite autonome avec l'enjeu de la gestion de la vitesse.**
Mots-clés— **algorithme, LIDAR, grille d'accumulation**

I. INTRODUCTION

Dans la conduite autonome, il est essentiel de détecter et d'éviter les obstacles pour assurer la sécurité des passagers et des autres usagers de la route. Les données lidar sont une source importante d'informations pour cette tâche, mais leur traitement peut être complexe et coûteux en termes de calculs. Dans ce travail, nous avons développé un algorithme pour traiter les données 3D provenant d'un lidar et les convertir pour pouvoir créer une grille 2D d'accumulation affichant une couleur en fonction de la présence ou non d'un obstacle et de la proximité qui le sépare du véhicule. Notre objectif était de créer une visualisation facilement lisible des données lidar, afin d'aider à la détection et à l'évitement des obstacles par le véhicule.

Au début de notre travail, nous avons effectué une recherche sur le sujet pour comprendre les approches existantes et les défis à relever, ainsi que la base de données à utiliser pour le développement. Une fois que nous avons compris notre sujet, nous avons commencé à développer différents algorithmes pour créer notre grille que nous présenterons plus en détail dans la partie développement. Lorsque nous avons créé notre grille, nous nous sommes posé la question sur la modification de la vitesse selon la distance entre le véhicule et l'obstacle. Pour conclure, nous avons sélectionné la grille la plus adaptée pour la tester sur le véhicule Artemips de l'université. Ces tests ont eu lieu sur le parking et nous ont permis de constater les améliorations à effectuer.

II. Recherche

1. Grille d'accumulation

Pour comprendre notre sujet et plus principalement la grille d'accumulation, que l'on a choisie d'utiliser, nous nous sommes intéressés à la thèse de Julien Moras [1]. Cette thèse est une très bonne ouverture pour la compréhension d'une grille à partir de laquelle découle notre compréhension du sujet.

Une grille d'accumulation est donc une grille de pixels utilisée pour représenter des données, dans notre cas provenant de données 3D, converti en 2D de manière visuelle. Elle est souvent utilisée dans le domaine de la robotique et de la vision par ordinateur pour représenter des informations sur la distance, la présence d'objets, etc.

Pour utiliser une grille d'accumulation, on commence par définir un espace de travail en 2D, généralement en utilisant des coordonnées cartésiennes ou polaires. On divise cet espace en une grille de pixels de taille fixe pour correspondre à un nombre de lignes et de colonnes définies. Chaque pixel de la grille est associé à une valeur, qui peut être un entier ou un réel. Lorsqu'on souhaite ajouter des données à la grille, on incrémente la valeur du pixel correspondant à la position des données dans l'espace de travail. Ainsi, chaque pixel de la grille représente l'accumulation de toutes les données qui se trouvent dans sa zone de couverture dans l'espace de travail.

Une fois que toutes les données ont été ajoutées à la grille, on peut utiliser une palette de couleurs pour visualiser les valeurs de chaque pixel de la grille. Par exemple, on peut utiliser une palette de couleurs allant du vert (pour les valeurs les plus faibles) au rouge (pour les valeurs les plus élevées). Ainsi, on peut facilement repérer les zones de la carte qui contiennent des obstacles.

Deux possibilités se présentaient pour détecter les objets :

Une grille d'occupation qui consiste à assigner une valeur à chaque cellule de notre grille. Les valeurs proches de 1 représentent une forte probabilité que la cellule contienne un obstacle. Les valeurs proches de 0 représentent une forte probabilité que la cellule soit inoccupée et sans obstacle.

Ou le principe de la grille d'accumulation qui, quant à elle, utilise un principe fréquentiste : plus une cellule est vue comme occupée, plus elle a de chances de l'être.

Nous avons choisi d'utiliser la grille d'accumulation d'une part, car nous n'avons pas à faire face à une incertitude dont fait preuve la grille d'occupation Cette incertitude concerne la présence ou non dans une cellule d'un obstacle. D'une autre part, car nous avons pu choisir des valeurs dites décisives qui nous permettent de considérer une cellule comme étant occupée ou non. Il faut aussi savoir que les grilles d'accumulation sont utilisées pour réaliser la cartographie d'un environnement extérieur en utilisant par exemple des lidars ce qui correspond à nos besoins et nous a confortés aussi dans notre choix.

2. Kitti

Nous avions besoin pour notre développement de l'algorithme, dont nous allons parler plus en détail par la suite, d'avoir une base de données sur laquelle travailler. Nous nous sommes orientés vers la base de données Kitti [2] (KITTI dataset) qui est une référence dans le domaine de la vision par ordinateur et de la conduite autonome. Elle a été mise en place par l'Institut de technologie de Karlsruhe (Allemagne) et est disponible en ligne gratuitement pour les chercheurs et les développeurs.

La base de données Kitti contient des enregistrements de données réelles acquises au bord de véhicules équipés de capteurs de haute précision, tels que des caméras et des lidars. Les enregistrements couvrent différents types de scénarios de conduite, tels que la ville, la campagne, les autoroutes, etc. Ils sont accompagnés de données de référence, telles que des images de caméras, des profils de hauteur et des annotations de scène, qui permettent de valider les algorithmes de traitement de l'image et de la profondeur.

La base de données Kitti est largement utilisée dans la recherche et le développement de systèmes [3] de vision par ordinateur et de conduite autonome, car elle offre une grande variété de données de qualité et de référence fiables.

III. Développement

Pour le développement de notre algorithme, nous sommes passés par différentes parties que nous allons expliquer. Dans un premier temps, nous avons eu à sélectionner le jeu de données adéquat afin de pouvoir faire des simulations se rapprochant le plus possible des cas réels de la vie quotidienne. Ce jeu de donnée nous servira tout au long de notre phase d'essai et nous sert de base pour le développement et la mise en place de notre première version de grille. Dans un second temps, il s'agira de réaliser et de mettre en place des algorithmes pour récupérer un affichage complet des données lidar. Nous allons donc détailler les différentes problématiques rencontrées vis-à-vis des choix des obstacles à identifier en priorité, à savoir ceux présentant le plus de risques et donc les plus proches du véhicule. Nous avons eu deux approches différentes que nous verrons plus en détail par la suite. La première qui se termine sur la première colonne de détection puis le second cas qui fait une différence jusqu'à une certaine distance. Nous finirons par une explication sur le fonctionnement de la modification de la vitesse du véhicule que nous avons choisi d'utiliser afin de le faire ralentir jusqu'à l'arrêt.

1. Sélection du jeu de données

La toute première étape est donc de charger les données 3D venant du lidar, cette étape, aussi anodine soit elle, est importante puisque c'est à partir de ces données que l'on crée notre grille. Ainsi, il est important de respecter les dimensions entre les données que l'on peut recevoir. Les données de Kitti sont en mètres et les données lidar de la voiture sont en centimètres. Cette étape est importante puisque, lors de la simulation, si les dimensions ne sont pas à la même échelle par rapport aux tests, les essais seront faussés.

Une fois que l'on obtient nos trois vecteurs X, Y et Z dans les dimensions que nous souhaitons, l'étape suivante est de filtrer, selon la hauteur des Z, les valeurs X et Y que nous gardons pour l'utilisation de notre grille. Sachant qu'un point est défini en (X, Y, Z), les valeurs Z sont utilisées sur la hauteur pour garder une hauteur définie de données selon la position du lidar et jusqu'à une hauteur légèrement supérieure à un trottoir pour éviter les erreurs de mesures lors du passage en 2D dans l'hypothèse d'avoir une route plane et peu ou pas pentue.

2. Grille d'accumulation 3D vers 2D

Maintenant que nous avons nos X et Y dans la plage en Z que nous souhaitons vérifier, il faut créer notre

grille d'accumulation. Pour cela, on choisit de faire une image qui sera quadrillée avec des cases d'une certaine taille selon la vitesse du véhicule. Les tailles, que l'on a définies sont du 25x25cm pour une vitesse inférieure à 10 Km/h, du 50x50cm pour une vitesse comprise entre 10 et 19 Km/h et finalement pour toutes vitesses supérieures ou égale à 20 Km/h, on a du 1x1m.

Lorsque l'on obtient les tailles de nos cases qui correspondent donc au pixel d'une image, il reste à vérifier si la case est occupée ou non. La première étape que l'on a effectuée est de parcourir une image que l'on a créée pour ressembler à notre map de ligne en ligne. Si dans une des cases, nous détectons un certain nombre de points sur cette zone, nous mettons en premier cette case de couleur blanche sur du noir.

Voici un exemple avec le cas des cases de 1x1m. Si un objet est détecté, la valeur doit être d'au moins 10 :

3	5	4	12	30	1	5	2	1	1
3	0	1	2	0	0	2	3	3	5
0	0	0	15	1	10	21	3	9	2
0	1	4	9	5	64	1	23	1	1

Figure 1 : Exemple Grille d'accumulation

Nous avons donc décidé de changer la couleur d'affichage de notre grille selon le nombre de points détecté dans une zone définie par la taille de la case.

Maintenant qu'on a pu créer notre première version d'une grille d'accumulation, nous allons chercher à donner différentes couleurs selon si l'objet détecté est dans la zone 1, la zone 2 ou encore la zone 3. Ces trois zones sont définies à partir du calcul de la distance de freinage.

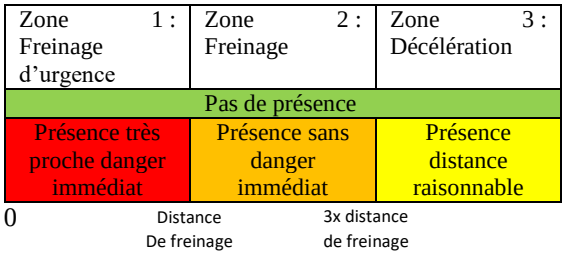

Figure 2 : Définition des zones

Ainsi nous obtenons cette représentation :

Zone 1		Zone 2				Zone 3	
2	15	5	10	15	2	11	3
3	4	5	1	2	0	0	0
0	15	0	0	0	0	15	2
0	9	1	23	4	6	4	2
1	3	4	8	5	3	4	0
0	10	4	6	7	9	1	5

Figure 3 : Grille d'accumulation

Une bonne visualisation est obtenue sur la distance des obstacles présents sur la route que la voiture risque de rencontrer. Malheureusement avec cette technique, on avait beaucoup de retard au niveau des calculs, c'est pourquoi nous avons choisi de nous arrêter à partir de la première détection.

3. Première détection

Il est préférable d'accélérer le traitement en évitant de vérifier toute la carte, puisque l'obstacle le plus proche détecté devrait être suffisant pour régler la vitesse du véhicule par la demande de freinage. Ainsi la première difficulté a été de recréer la grille d'accumulation en faisant une visualisation par colonne au lieu de ligne pour la création. Ensuite, nous devons créer l'affichage et forcer l'arrêt des calculs pour la détection si un obstacle a été détecté. En se basant sur la figure précédente, nous allons étayer notre explication.

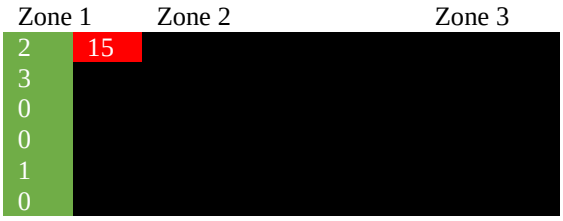

Figure 4 : Grille d'accumulation unique v1

En se basant sur cette grille, nous sommes censés uniquement ralentir à une certaine vitesse et ne pas arrêter le véhicule. Puisque l'on souhaite ralentir et passer à côté de l'obstacle, s'il ne se trouve pas devant, mais être capable de ralentir assez pour pouvoir freiner dans le cas où il se déplace vers l'avant. Ainsi, nous avons décidé de vérifier la colonne en entier au lieu de s'arrêter dès le premier obstacle détecté. Nous pourrons alors modifier la vitesse à retirer selon l'importance de la position d'obstacle sur la grille.

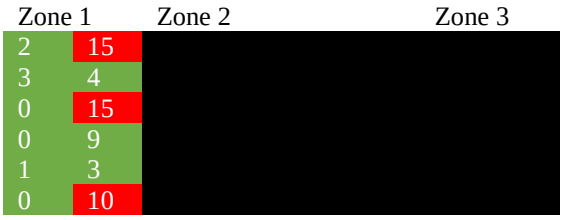

Figure 5 : Grille d'accumulation unique v2

Maintenant avec cette version, on sait que l'obstacle ayant la proximité la plus importante est la deuxième valeur détectée dans cette colonne, car il s'agit de la valeur possédant le risque le plus proche. La vitesse résultante que l'on doit retirer est alors la plus importante. L'avantage de cette technique se porte sur la vitesse de calcul en temps réel pour la détection d'obstacle. En contrepartie, lorsque nous

sommes sur une route bondée, comme une ruelle piétonne, nous avons des soucis pour la sélection du plus grand risque puisque si on détecte dès la première colonne quelqu'un sur les côtés, on ne va pas vérifier si, devant, il y a quelque chose ou quelqu'un qui est un bien plus grand risque.

4. Détection Multiple

Nous avons donc vu jusqu'à présent deux méthodes. Chacune avec leurs avantages et inconvénients respectifs. La première avec l'affichage complet de la map, nous avions la possibilité de choisir parmi toutes les détections laquelle est la plus risquée, mais cela se traduisait par un temps de calcul assez long pour créer la grille.

La deuxième méthode avec l'affichage jusqu'à la détection du premier obstacle et on s'arrête à la fin de sa colonne, ce qui nous permettait d'avoir un temps de calcul beaucoup plus rapide et permettait de traiter plus de données. Par contre, son inconvénient est que lorsque le véhicule se trouve dans des situations où il y a une forte région d'activité, celui-ci ne sera pas optimal pour la correction de sa vitesse. Ainsi la meilleure des solutions a été de faire un mélange des deux solutions pour avoir un temps de calcul respectable et ne pas avoir à calculer toute la grille.

Pour réaliser cela, nous partons sur le principe que si le premier obstacle détecté se trouve dans la zone 1, nous allons continuer à créer la grille d'accumulation jusqu'à une partie de la zone 2. En se basant sur une équation de la distance qui est :

$$da = dist_{arret} + \left(\frac{vit}{10}\right)^2 + \frac{vit*2000}{3600} \quad (1)$$

Cela dans le but, en prenant en compte la vitesse du véhicule (vit) et $dist_{arret}$ représente l'espace que l'on souhaite avoir entre le véhicule à l'arrêt et l'obstacle. Da nous permet de bien sélectionner la vitesse à retirer du véhicule selon la formule que nous expliquerons à la suite. Une fois cette colonne atteinte et finie, nous nous arrêtons de créer la grille pour gagner du temps. Ainsi en partant de la même grille d'exemple :

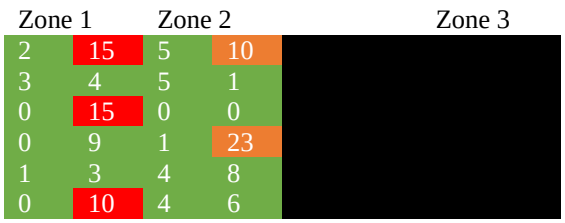

Figure 6 : Grille d'accumulation multiple zone 1

Pour modifier la vitesse nous commençons par calculer l'angle α, qui nous permettra de choisir l'importance de la décélération en fonction de la vitesse à retirer avec :

$$\alpha = \text{atan}\left(\frac{y}{x}\right) * \frac{180}{pi} \quad (2)$$

Grâce aux formules de vitesse que nous présenterons par la suite, ainsi dans le cas précis, ce serait bien l'obstacle sur le 15 rouge avec un angle d'environ 5 °. Dans le cas où un obstacle est détecté plus loin que la distance nécessaire au temps d'arrêt, nous arrêtons de calculer une fois la colonne finie, car le ralentissement à appliquer à ce moment sera déjà respecté et nous n'avons pas besoin de vérifier la suite. Ainsi en modifiant la grille pour correspondre à l'exemple :

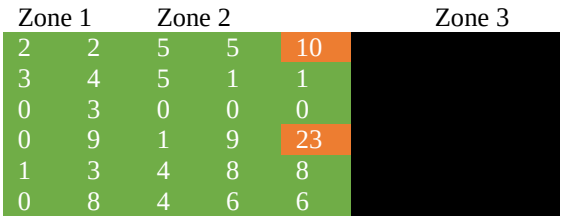

Figure 7 : Grille d'accumulation multiple zone 2

Comme on peut le voir, nous nous sommes arrêtés à la colonne de détection. C'est cette détection (la valeur 23) qui nous donnera la priorité et qui nous donne la vitesse à appliquer pour ralentir.

5. Calcul de la vitesse

Nous nous sommes donc posé la question de la modification de la vitesse à appliquer selon la distance entre le véhicule et l'obstacle. Ainsi, nous avons une distance en x et y à prendre en compte. Nous voulions utiliser un angle (alpha) entre la voiture et l'obstacle en utilisant une formule de trigonométrie afin de pouvoir détecter l'obstacle le plus proche.

Avec y et x la position de l'obstacle sur la grille. Et le fait que nous souhaitons la valeur en degrés au lieu de radian. Nous souhaitons ne pas différencier angle positif et négatif donc nous récupérons la valeur absolue de l'angle.

Ensuite, nous avons choisi deux formules qui calculent la vitesse à retirer dans la suite. La première formule ne respecte pas vraiment les unités, mais celle-ci est fonctionnelle puisque l'on aura une vitesse selon des degrés. Cette formule trouvée est donc la suivante :

$$VKm_{hd} = \frac{vit_{reference} * dist_{arret}}{\frac{\alpha}{val} + x} \quad (3)$$

Nous avons donc besoin pour cette formule d'une « vitesse de référence » qui sera la vitesse maximale que le véhicule devra atteindre lors de la conduite. Il

y a aussi la distance d'arrêt qui concerne l'espace que l'on souhaite avoir entre le véhicule à l'arrêt et l'obstacle. Soit obtenue en faisant :

$$dist_{arret} = dist_{souhaité} + position_{lidar} \quad (4)$$

L'angle « α » est à diviser par une valeur « val » à définir empiriquement. Pour les tests réalisés sur le parking avec un choix de 15 km/h pour la vitesse de référence, on trouve val est égal à 14,5 et finalement « x » qui est la distance entre le lidar et l'obstacle. Cette formule nous retourne toujours une valeur positive, mais pouvant être supérieure à la vitesse de référence, c'est pourquoi on doit imposer une limite.

Notre deuxième version respecte beaucoup plus les dimensions des unités, mais doit avoir plus de valeur définie empiriquement et passer par deux calculs. Le premier calcul à appliquer concerne le calcul de l'importance de l'angle à apporter dans la formule :

$$K = K_{max} - K_{min} * \frac{\alpha}{\alpha_{max}} \quad (5)$$

Nous devons donc définir la valeur de K_{max} et K_{min} que nous souhaitons appliquer. Ces valeurs doivent être choisies empiriquement et après plusieurs test et erreur. Soit avec les mêmes conditions de tests que précédemment, K_{max} est égal à 3,1 et K_{min} est égal à 0,5. Nous devons aussi choisir le α_{max} mais celui-ci est plus simple à choisir puisqu'il s'agira de 90°, la valeur maximale que l'on peut obtenir avec la grille. Maintenant que nous possédons la valeur K, nous pouvons calculer la vitesse :

$$Vkm_h = abs(K * \frac{x - da}{dist_{freinage} - da}) \quad (6)$$

On récupère la valeur absolue qui sera toujours limitée par la vitesse de référence dans le cas où le résultat est supérieur à la vitesse de référence. L'équation de la distance (da) est nécessaire ainsi que la distance de freinage. La distance de freinage est obtenue en appliquant les dizaines de la vitesse multipliées par 6. On ne prend pas en compte la tenue de route pour le moment, mais cela est une caractéristique à prendre en compte et à améliorer.

Maintenant que nous avons les deux façons de calculer la vitesse, il ne nous reste plus qu'à choisir celle à utiliser en fonction des tests réalisés préalablement. Une fois choisie, on fait une différence entre la vitesse de référence et la vitesse calculée pour obtenir la vitesse que le véhicule doit avoir. Ainsi, il ne s'agit que de faire :

$$Vitesse = vit_{reference} - vit_{choisie} \quad (7)$$

Nous finissons avec un tableau comparatif des méthodes.

	Affichage complet	Affichage première détection	Affichage distance minimum
Vitesse de traitement	Lente	Très rapide	Rapide
Observation des résultats	Totale	Insuffisante	Suffisant

Tableau 1 : Comparaison des résultats

IV. ARTEMIPS

Le premier obstacle auquel nous nous sommes confrontés est un problème de dimension (d'unité) retourné par les deux lidars du véhicule de l'uha. En effet, les données récupérées par Kitti étaient directement en mètre donc nous nous sommes basés sur des mètres pour créer la grille. Comme sous-entendu, les lidars de l'uha ne retournaient pas des données en mètre, mais en centimètre. Nous avons donc fait le passage de centimètre à mètre.

Nous avons par la suite été confrontés à un problème de référentiel lors des tests. Lors de la simulation, les lidars qu'utilisait la base de données Kitti étaient placés avec un référentiel X, Y, Z tel que :

Figure 8 : Lidar et référentiel véhicule Kitti

Ce qui n'était pas le cas des lidars du véhicule de l'ENSISA, qui lui avait les lidars montés à l'envers et un référentiel X, Y, Z tel que :

Figure 9 : Lidar gauche et référentiel véhicule uha

Nous avons procédé à des changements de référentiel pour correspondre avec notre algorithme en inversant la valeur des X et Y et en changeant le signe du Z. Ce qui nous donne :

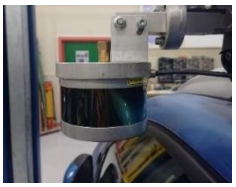

Figure 10 : Lidar gauche et référentiel véhicule uha adapté

Pour le lidar de droite, le référentiel véhicule uha est le même que celui de la figure 8, mais avec x et y dans le sens contraire. Nous avons adapté le référentiel de la même manière que celui de gauche, mais en changeant le signe des X, Y et Z.

Nous avons ensuite effectué des tests pratiques avec lesquels nous avions la possibilité de sélectionner le lidar à utiliser. Pour ces tests, nous avons utilisé la dernière version de la grille, soit la grille multiple puisque celle-ci est la plus adaptée pour un usage réel. Ces tests ont été réalisés avec le logiciel RTMaps. Nous ne pouvons pas montrer les résultats mais nous allons les décrire.

Nous avons remarqué avec les premiers tests que notre grille fonctionnait et se mettait à jour. Par contre, nous n'étions pas satisfaits du temps de calcul qui influait sur l'affichage de la grille. Celui-ci étant conséquent avec environ une seconde de décalage entre l'affichage et ce qui se déroulait dans l'environnement. Plusieurs suppositions ont été faites pour de possibles améliorations, dont une, que nous avons eu le temps de vérifier.

Notre première hypothèse a été d'améliorer le filtrage spatial que l'on obtient des lidars en filtrant en fonction de nos Z, X et Y que l'on souhaite utiliser pour l'affichage, ainsi que de supprimer les points déjà détectés sur une cellule pour ne pas avoir à la reparcourir. Avec cette hypothèse que nous avons eu le temps de tester, nous passons de 131 grilles à 310 grilles, soit une amélioration d'un rapport de 2,36 soit, 136 % correspondants à 179 grilles supplémentaires. Cette solution est déjà plus acceptable, mais manque toujours de réactivité. C'est pourquoi, au lieu d'utiliser deux vecteurs distincts pour former notre X et Y, nous avons décidé d'utiliser un vecteur qui contient la paire des X et Y. Avec cette modification, on passe de 310 grilles à 535, soit une amélioration d'un rapport de 1.72, soit 72 %, correspondants à 223 grilles supplémentaires.

La deuxième hypothèse serait d'effectuer un filtrage spatial selon les trois zones pour gagner du temps de calcul.

L'hypothèse suivante consisterait à séparer l'affichage de la grille et la commande sur la grille en passant par un tableau représentant notre grille à afficher et serait envoyé à un autre module pour créer la grille. Finalement, on pourrait utiliser des threads pour accélérer l'accumulation de la grille.

V. Conclusion

Le but de ce projet se portait principalement sur la perception de l'environnement. La mission principale était d'acquérir les données des lidars présents sur la voiture, de les traiter et de les représenter pour pouvoir ensuite être capable de détecter des obstacles et agir sur la vitesse selon la proximité pour ralentir, voire arrêter le véhicule. Pour y arriver, nous avons utilisé le principe de la grille d'accumulation qui a occupé une place primordiale. Il s'agissait de découper les mesures des lidars en une grille 2D afin de déterminer si une cellule est occupée ou non. Un découpage en trois zones a donc été effectué. Une zone de proximité immédiate, une zone de ralentissement et une zone sans danger immédiat. Nous avons été capables grâce à l'implémentation d'algorithme de proposer un moyen de gérer la vitesse et le ralentissement du véhicule.

Nous sommes satisfaits du travail fournit. Cependant, il reste quelques points à améliorer ou à tester. En effet, la réactivité de l'affichage reste à améliorer comme nous l'avons souligné et il nous resterait à tester l'arrêt du véhicule à une distance donnée (grâce au calcul) entre le véhicule et l'obstacle avec la modification de la vitesse en utilisant la commande de freinage.

Référence

[1] Julien MORAS. Grilles de perception évidentielles pour la navigation robotique en milieu urbain. Autre[cs.OH]. Université de Technologie de Compiègne, 2013. Français. ffNNT : 2013COMP2057ff. fftel00866300

[2] Karlsruhe Institute of Technology, KITTI Vision Benchmark, https://www.cvlibs.net/datasets/kitti/, 12 Janvier 2023

[3] Ruddy THEODOSE, Dieumet DENIS, Thierry CHATEAU, Vincent FREMONT, Paul CHECCHIN. A Deep Learning Approach for LiDAR Resolution-Agnostic Object Detection. IEEE Transactions on Intelligent Transportation Systems, 2021, pp.1-12. ff10.1109/TITS.2021.3130487ff. ffhal-03485613