

Apport de l'intelligence artificielle dans la perception LiDAR, la création et le suivi de trajectoires

Rapport de stage universitaire
Master Perception Artificielle et Robotique

Réalisé par Clément Chupin

INRAe

UCA
UNIVERSITÉ
Clermont
Auvergne

Directeur de stage	Roland Lenain
Directeur de Master	Romuald Aufrère
Encadrants de stage	Dimia Iberraken / Cyrille Pierre
Référent universitaire	Benoit Thuilot

Résumé : Ce projet s'inscrit dans le cadre d'un stage de fin d'études du Master Perception Artificielle et Robotique, formation orientée recherche. L'objectif principal du stage est, par l'immersion en milieu professionnel, de mobiliser diverses compétences autour d'un projet, afin de développer une expérience sur des domaines spécifiques.

L'objectif de ce projet sera d'évaluer le potentiel apport de l'intelligence artificielle dans le domaine de la perception en milieu agricole. Ce projet se construira autour de trois axes essentiels permettant le contrôle de robot en milieu agricole : le traitement de données LiDARs, l'intelligence artificielle, et enfin l'automatique. Ces domaines scientifiques nous permettront de réaliser une architecture logicielle, permettant de percevoir, de comprendre et d'interagir avec l'environnement.

Par les récentes recherches dans le domaine, on constate que l'intelligence artificielle peut apporter de nombreuses solutions technologiques permettant une plus grande robustesse et adaptabilité des algorithmes face à une situation inconnue. Cet outil est également pertinent au sein de l'équipe Roméa de l'INRAE, spécialisée dans la robotique agricole, où la diversité et l'évolution constante des données agricoles pose des défis particuliers qui pourraient être traités par l'utilisation de l'intelligence artificielle, notamment dans le domaine de la perception, de la prise de décisions, et du contrôle.

Au cours du projet, nous avons exploré et évalué l'apport de l'intelligence artificielle en tant que solution technique pour améliorer notre algorithme de détection et de suivi de rangées de plantes, ce qui nous a conduits à envisager d'autres solutions. En effet, pour que notre algorithme final soit performant, il doit être entraîné sur une base de données reflétant la réalité. Malheureusement pour nous, la mise en place d'une base de données d'entraînement proche du réel s'est avéré être une tâche complexe et chronophage, ce qui rend impossible la prise en compte des moindres changements ou nouveautés que le cadre applicatif réel propose. Ces limitations nous ont finalement dissuadé d'utiliser des réseaux de neurones dans ce projet portant sur la perception.

Suite aux limitations de l'utilisation de l'intelligence artificielle, nous avons exploré l'approche probabiliste de la perception. Tout d'abord, pour la détection de plantes, nous avons adopté une approche géométrique de la détection de plantes, nous permettant d'assurer une fiabilité de perception de l'environnement en de multiples circonstances. Par la suite, nous avons étudié la création d'une trajectoire initialement peu robuste aux fausses détections. Par son fonctionnement prenant en compte chaque point d'intérêt, il était souvent amené à faire des choix irréversibles et rester coincé dans une mauvaise configuration. On a donc cherché à identifier, par mixtures de gaussiennes, une représentation générale et probabiliste de la répartition du couvert végétal. On fera également intervenir un filtre de Kalman pour permettre de prendre en compte l'incertitude de mesure et d'état de cet algorithme.

Ce projet nous aura permis d'explorer les limitations de l'algorithme initial de perception et de contrôle, permettant difficilement d'assurer robustesse et autonomie sur l'ensemble des cas réels (nécessité d'un réglage manuel, erreurs de détections, erreurs de trajectoires....). Par l'utilisation de l'intelligence artificielle, on a vu que l'on pouvait potentiellement améliorer nos algorithmes, cependant par ses diverses limitations, l'utilisation de l'intelligence artificielle dans le cas réel compromet la robustesse générale de notre algorithme.

Sommaire

1 Contexte général du stage	2
1.1 Agroécologie	2
1.2 INRAE : institut national de recherche pour l'agriculture, l'alimentation et l'environnement	2
1.3 Unité de recherche TSCF : Technologies et systèmes d'information pour les agro-systèmes	3
1.3.1 Roméa : RObotique et Mobilité pour l'Environnement et l'Agriculture	4
1.3.2 Projet TIARA : Collaboration entre ROMEA et Sabi Agri	4
1.4 Objectif du stage	5
2 État de l'art	6
2.1 Perception LiDAR dans le milieu agricole	6
2.2 Furrow : algorithme de perception et de suivi de trajectoire	7
2.3 Limitations du Furrow	9
2.4 Apport de l'intelligence artificielle	11
3 Mise en place de l'environnement de simulation	13
3.1 Création de base de données	14
3.2 Méthode d'évaluation de nos algorithmes	17
3.2.1 Évaluation de robustesse	17
3.2.2 Évaluation de précision	18
3.2.3 Enregistrement de données et expérimentation réel	19
4 Architecture 1 : Détection de configuration agricole	21
4.1 Traitement nuage de points	22
4.2 Choix du réseau de neurones	23
4.3 Expérimentations	25
4.4 Conclusion	27
5 Architecture 2 : Détection de plantes	29
5.1 Segmentation du nuages de points	29
5.2 Détection de points d'intérêt	33
5.3 Expérimentations	34
5.4 Conclusion	36
6 Architecture 3 : Suivi de trajectoire par réseau de neurones	38
6.1 Expérimentations	39
6.2 Conclusion	40
7 Architecture 4 : Suivi de trajectoire par modèle probabiliste	42
7.1 Recherche des points d'intérêt par modèle gaussien	44
7.1.1 Définition du modèle gaussien	45
7.1.2 Fonctions de pertes	49
7.2 Filtre de Kalman	51
7.3 Calcul de trajectoire	52
7.4 Expérimentations	54
7.5 Conclusion	57
8 Conclusion de stage	59
8.1 Apport de l'intelligence artificielle dans la robotique agricole	59
8.2 Analyse Forces-Faiblesses et Risques-Opportunités du projet mené	59
8.3 Perspectives	60

1 Contexte général du stage

1.1 Agroécologie

Au cours des dernières décennies, le secteur agricole a été profondément influencé par des transformations démographiques, des évolutions des habitudes alimentaires et des avancées technologiques. En effet, d'après l'OCDE, «L'un des principaux défis que doit relever le secteur agricole consiste à nourrir une population mondiale en expansion tout en réduisant son empreinte écologique et en préservant les ressources naturelles pour les générations futures.». Face à une demande croissante en produits alimentaires, l'agriculture doit naviguer entre la nécessité d'accroître la production et les préoccupations environnementales. Cette situation requiert des approches qui permettent de maximiser l'efficacité des ressources tout en minimisant les effets négatifs sur l'écosystème.

Actuellement, l'agriculture est largement orientée vers une production à grande échelle et à faible coût, avec une attention limitée portée aux conséquences environnementales. D'après l'INRAE, «Les questions autour des pesticides de synthèse, de leurs usages et de leurs impacts sont récurrentes dans notre société. Lorsqu'ils sont massivement utilisés pour protéger les cultures, les produits phytosanitaires finissent par se retrouver aussi dans les sols, les eaux, l'air et dans notre alimentation, avec des impacts sur la biodiversité et sur notre santé». Cette approche a souvent conduit à l'utilisation intensive de produits chimiques et de pesticides, engendrant des problèmes environnementaux et sanitaires.

Bien que diverses alternatives aient été proposées, notamment la promotion de la production locale, l'adoption de pratiques agricoles respectueuses de l'environnement et l'encouragement de l'agroécologie, par la diversité et la complexité des problématiques rencontrées, il en est donc difficile de définir une solution universelle. La transition agroécologique apparaît alors comme un compromis nécessaire qui combine différentes solutions technologiques, sociales et environnementales afin de créer un modèle agricole plus durable et respectueux de l'environnement.

Les objectifs concrets se concentrent autour des 5 principes initiaux de l'agroécologie (Altieri [12]) :

- Permettre le recyclage de la biomasse
- Garantir les conditions de fonctionnement du sol
- Minimiser les pertes de ressources non renouvelables, réduire l'usage d'intrants externes
- Favoriser la diversification
- Permettre les interactions biologiques

1.2 INRAE : institut national de recherche pour l'agriculture, l'alimentation et l'environnement

L'INRAE est un institut national de recherche publique qui a pour mission de développer de nouvelles approches de production agricole :

«INRAE a pour mission de réaliser, d'organiser et de coordonner, à son initiative ou à la demande de l'État, tous travaux de recherche scientifique et technologique dans les domaines de l'agriculture, de l'alimentation, de la forêt, de l'environnement, de l'eau, de la biodiversité, de la bioéconomie, de l'économie circulaire, de la gestion durable des territoires et des risques dans les champs de compétence précités. »

Extrait du décret fondateur de INRAE N°2019-1048 du 10 octobre 2019.

Les sujets de recherche vise à explorer différentes voies pour trouver des solutions innovantes en vue d'une agriculture plus durable et respectueuse de l'environnement. Il est souligné sur leur site : « Nos recherches visent à :

- Préserver les milieux cultivés
- Réduire l'utilisation d'intrants de synthèse (pesticides, antibiotiques, engrais) grâce aux régulations biologiques
- Diversifier les productions agricoles à toutes les échelles, du champ à l'assiette.»

Sur ses 18 centres, INRAE est engagé dans 33 sites universitaires



Figure 1: Carte d'implantation de l'INRAE sur le territoire français

Pour mener à bien sa mission, l'INRAE compte dans ses effectifs environ 11 500 personnes réparties dans une vingtaine de sites à travers toute la France (figure 1), avec 268 unités de recherche. Grâce à un budget d'environ un milliard d'euros, l'institut vise à devenir une référence internationale de la recherche en agriculture. En effet, plus de la moitié de leurs publications sont réalisées en collaboration avec d'autres pays. Cette structure solide et cette ouverture à la collaboration internationale permettent à l'INRAE de jouer un rôle de premier plan dans l'avancement des connaissances et des solutions pour une agriculture plus durable.

Dans le contexte de multiples défis écologiques, techniques, sociaux et politiques, l'INRAE assume la responsabilité de forger de nouvelles approches en matière de production agricole. Cet engagement se matérialise par une recherche approfondie dans des domaines variés, mettant en avant notamment la biologie, la chimie et la géologie. Cependant, cette recherche demeure ouverte à une multitude de solutions potentielles, parmi lesquelles la robotique, l'électronique et la mécanique occupent une place notable.

1.3 Unité de recherche TSCF : Technologies et systèmes d'information pour les agro-systèmes

L'unité de recherche Technologies et systèmes d'information pour les agrosystèmes (TSCF) est rattachée au Centre INRAE Clermont Auvergne-Rhône-Alpes qui conduit des recherches de pointe dans des secteurs

clés de l'agriculture, de l'environnement et de l'alimentation. Cette unité est composée de 3 équipes qui rassemblent 60 agents, est implantée sur 2 sites : le Pôle scientifique et universitaire des Cézeaux à Aubière (63) et le Site de recherche et d'expérimentation de Montoldre (03).

Elle mobilise les sciences pour l'ingénier et les sciences et technologies de l'information et de la communication pour conduire des recherches sur les méthodes et outils pour une ingénierie des systèmes agro-environnementaux. Par exemple le développement d'outils de supervision de flottes de robots. Elle conduit également des activités de recherche, d'expertise et d'essai dans le domaine des performances et de la sécurité des agroéquipements pour contribuer à l'amélioration de la sécurité en agriculture et à la réduction des pollutions d'origine agricole.

Grâce à ses travaux de recherche technologique, elle apporte des réponses concrètes aux besoins d'une agriculture productive écologiquement responsable et de la gestion de l'environnement.

1.3.1 Roméa : RObotique et Mobilité pour l'Environnement et l'Agriculture

L'équipe ROMEA, au sein de laquelle se déroule le stage, mène des recherches en robotique pour le développement d'engins autonomes en milieux naturels pour accompagner la transition écologique de l'agriculture. En effet, la pression sur la main-d'œuvre est considérable et la pénibilité des métiers agricoles importante. De plus, le maintien des niveaux et de la qualité de production constituent également une contrainte. Le déploiement de robots, capables de réaliser des travaux agricoles de manière autonome, émerge ainsi comme un moyen de concilier ces problématiques opposées.

Cependant, le développement de ces technologies nécessite de pouvoir garantir un haut niveau de précision et de sécurité, malgré la diversité des situations rencontrées, les incertitudes dans les interactions entre les robots et l'environnement, et la multiplicité des tâches à accomplir. En outre, la nature des travaux à réaliser dans le domaine agricole requiert la coopération avec l'humain et implique que les robots puissent s'adapter aux actions humaines.

L'équipe développe ainsi des approches de perception et de commande permettant aux robots de s'adapter à leur environnement de façon sûre, quel que soit le contexte d'évolution. Pour ce faire, les travaux menés au sein de ROMEA couvrent différents aspects, allant du développement de nouveaux capteurs pour la caractérisation des interactions entre le robot et son environnement, jusqu'au développement d'algorithmes de reconfiguration de comportements robotiques, en passant par la coopération de robots avec l'humain, et le contrôle de manipulateurs mobiles.

1.3.2 Projet TIARA : Collaboration entre ROMEA et Sabi Agri

Le projet TIARA (Toward Intelligent Adaptable Robots for Agriculture) est un laboratoire commun en collaboration avec l'entreprise privée Sabi Agri, le but est de développer un tracteur électrique agricole avec plusieurs niveaux d'autonomie, la recherche se concentre actuellement sur le suivi de trajectoire multi-capteurs (GPS, LiDAR), couplé avec des recherches en parallèle sur la segmentation des données LiDAR pour mieux appréhender son environnement, ainsi que la traversabilité des obstacles.

L'objectif du LabCom TIARA est d'accélérer le développement d'une architecture de commande versatile sur les tracteurs électriques de Sabi Agri, permettant de travailler de façon autonome en interaction multi-machines, multi-capteurs, multi-stratégies, et le tout supervisé humainement sur place et/ou à distance.



Figure 2: Tracteur électrique POM de la société Sabi Agri

Nos simulations et expérimentations lors de ce stage se concentreront sur le tracteur électrique POM, anciennement ALPO (figure 2). De manière générale, les recherches menées peuvent être adaptées à n’importe quel engin équipé d’un capteur LiDAR. L’objectif ultime est de tester nos algorithmes sur différentes machines, en utilisant des capteurs abordables pour faciliter la diffusion de nos solutions technologiques. En adoptant cette approche, nous cherchons à rendre nos innovations plus accessibles et à favoriser leur adoption dans un large éventail d’applications, contribuant ainsi à l’avancement de la robotique et de la technologie LiDAR dans divers domaines industriels et environnementaux.

1.4 Objectif du stage

Notre projet a pour but de mener une évaluation approfondie de la valeur ajoutée de l’intelligence artificielle dans le domaine spécifique de la perception artificielle. À travers ses avancées ininterrompues et son vaste étendue d’applications, nous entretenons l’hypothèse que le domaine de l’intelligence artificielle détient le potentiel nécessaire pour apporter une contribution significative aux défis rencontrés dans le contexte de la robotique agricole. En d’autres termes, notre démarche vise à améliorer l’autonomie, la robustesse et la précision de nos algorithmes en leurs conférant la capacité d’analyse avancée des données capteurs, afin de mieux appréhender leurs environnements.

Finalement, au sein de l’équipe Roméa, nous souhaitons poursuivre l’intégration et l’utilisation de l’intelligence artificielle en tant que nouvel outil au sein de la robotique agricole, pour à la fois capitaliser les travaux et algorithmes précédent, tout en intégrant des améliorations et de nouvelles fonctionnalités.

2 État de l'art

2.1 Perception LiDAR dans le milieu agricole

Au sein de l'équipe Roméa, la solution technologique privilégiée est le capteur LiDAR. Ce dispositif fonctionne en émettant des faisceaux lasers pour mesurer les distances et ainsi percevoir la géométrie de l'environnement. On mesure le temps de vol de chaque rayon renvoyé au capteur pour déterminer la distance parcourue, ce processus de télémétrie laser nous permet d'obtenir une représentation de notre environnement via un scan de profondeur. Il existe des LiDARs mono-nappe qui nous donne une représentation 2D de l'environnement, ainsi que des LiDARS multi-nappes qui nous donne une représentation 3D.

La perception LiDAR présente des avantages significatifs pour le milieu agricole. Contrairement à une caméra RGB classique, nous pensons que le LiDAR est beaucoup plus robuste face aux variations environnementales telles que la poussière, l'humidité, la luminosité, les intempéries et le brouillard. Il représente également un défi technologique intéressant, étant donné que de nombreuses recherches ont déjà été effectuées dans le domaine du traitement d'images et de vidéos, notamment pour des applications industrielles, médicales et militaires, les recherches autour du LiDAR se placent autour de problématiques moins étudiées et de domaines d'applications innovants.



Figure 3: Capteur LiDAR SICK LMS-151

Dans ce projet, nous utiliserons un capteur LiDAR SICK LMS-151 à une nappe (figure 3), la nappe est horizontale par rapport au LiDAR, et le LiDAR est généralement incliné à 45° par rapport au tracteur, ce qui nous permet à l'aide de l'odométrie d'obtenir une reconstruction 3D du terrain agricole (figure 4).

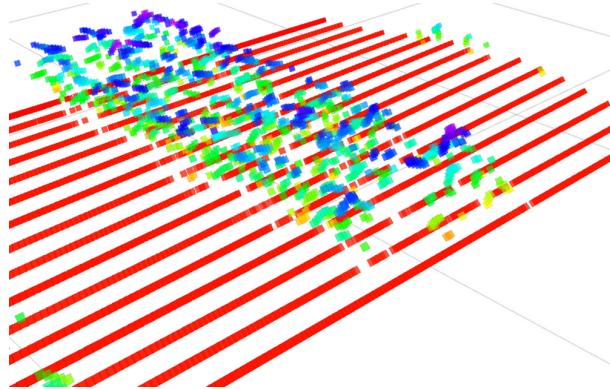


Figure 4: Reconstruction 3D basé sur l'accumulation de scan mono-nappe

Par l'accumulation de scans LiDAR mono-nappe, ainsi que la prise en compte du déplacement du véhicule (l'odométrie), on est capable de former une reconstruction 3D où chaque scan LiDAR est replacé dans le repère du robot au fur et à mesure de son avancement dans les rangées de plantes. Ce processus nous permet d'avoir une représentation géométrique de l'environnement agricole.

Le contenu technique de notre travail s'inscrit dans le domaine de la robotique mobile agricole, où un des objectifs principaux est de permettre aux engins agricoles de se localiser et d'appréhender leur environnement afin d'y exécuter des tâches en milieux non-structurés. Concrètement, le véhicule doit être en mesure de se guider en utilisant les données GPS et de percevoir son environnement au moyen de divers capteurs. Cette capacité lui permet de se déplacer dans un contexte agricole sans causer de dommages aux plantes, et plus largement aux biens et aux personnes. En résumé, l'algorithme de contrôle ne s'appuie pas uniquement sur la perception de l'environnement, une composante essentielle de la localisation du tracteur est le basculement entre différents paradigmes de localisation.

Pour permettre le suivi de trajectoire, il faut tout d'abord définir ce que devra fournir l'algorithme de perception aux lois de contrôle. L'algorithme fournira un écart angulaire et latéral par rapport à la trajectoire formée par les plantes (figure 5). La loi de contrôle utilisé prendra en compte les roues avant indépendamment actionnées en angle de braquage, ainsi que la configuration générale de notre véhicule [5].

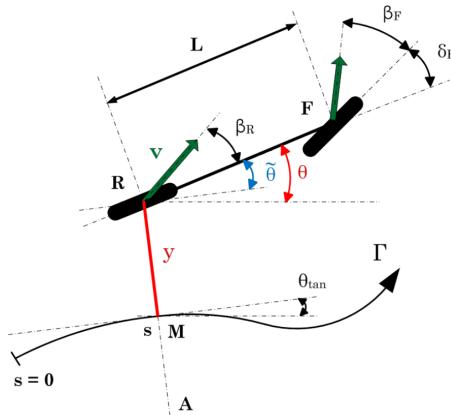


Figure 5: Expression de l'écart angulaire et latéral, modèle cinématique étendu [5]

Nous définissons une trajectoire de référence basée GPS RTK (Global Positioning System, Real Time Kinematic), ce sera notre connaissance *a priori* de notre environnement. Ensuite intervient le suivi spécialisé, il se basera sur la perception de l'environnement et permettra d'obtenir un positionnement précis et adaptée à l'environnement pour effectuer notre tâche agricole (par exemple, un désherbage mécanique au plus proche de la plante). Finalement, notre guidage se basera sur le GPS, si notre perception identifie une trajectoire à proximité comportant suffisamment de plantes, une transition automatique sera effectuée vers cette trajectoire, garantissant ainsi un suivi à la fois robuste et adapté à notre environnement (Pierre C. et al., 2022 [1]). Notre algorithme de perception n'a alors pas pour but de fonctionner en situation de perte de vue, de décalage ou de confusion. Son rôle est d'assurer le suivi précis de la trajectoire lorsque l'engin agricole est aligné avec les rangées de plantes à suivre. Ainsi, l'algorithme de perception garantit un suivi de trajectoire précis pour la réalisation de diverses tâches agricoles, telles que le désherbage mécanique, le semis de graines et la pulvérisation.

2.2 Furrow : algorithme de perception et de suivi de trajectoire

À l'origine, ROMEA a conçu un algorithme de suivi de sillons de roues basé LiDAR, par la suite, nous appellerons l'algorithme "Furrow" pour sa fonction principale, suivre un ou plusieurs sillons ("Furrow following").

Puis, par la suite, l'algorithme a été étendu pour suivre des rangées de plantes en milieu agricole (Iberraken D. et al., 2022 [4]). On détecte via le capteur LiDAR les rangées de plantes dans le scan courant, puis, par la commande aux actionneurs, on contrôle le tracteur afin de suivre la planche¹ formée par les rangées de plantes (figure 6).

¹planche = ensemble des rangées de plantes qu'enjambe un engin agricole, une largeur généralement entre 50cm et 200cm

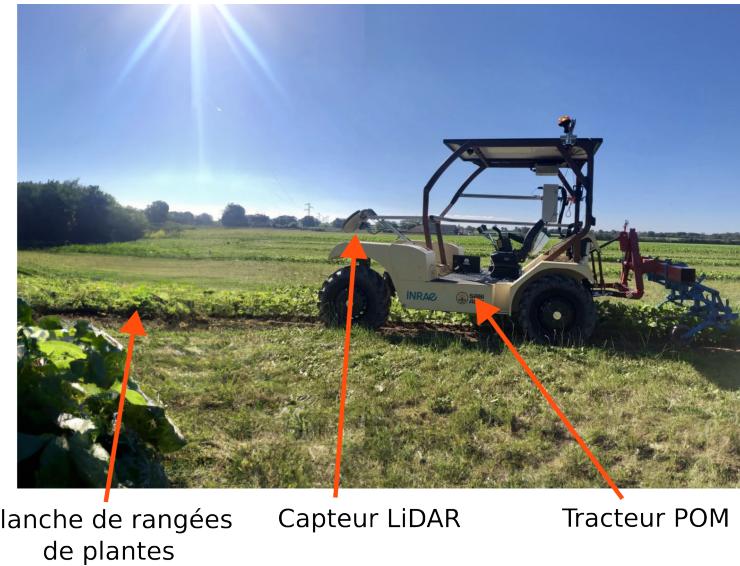


Figure 6: Présentation des différents éléments de l'environnement agricole

Pour réaliser le suivi de plantes, l'algorithme opère en deux étapes principales :

1. La détection : on détermine dans notre environnement la position des plantes par rapport au tracteur
2. La création et le suivi de trajectoire : à partir de la position des plantes, on construit et on suit la trajectoire formée par celles-ci

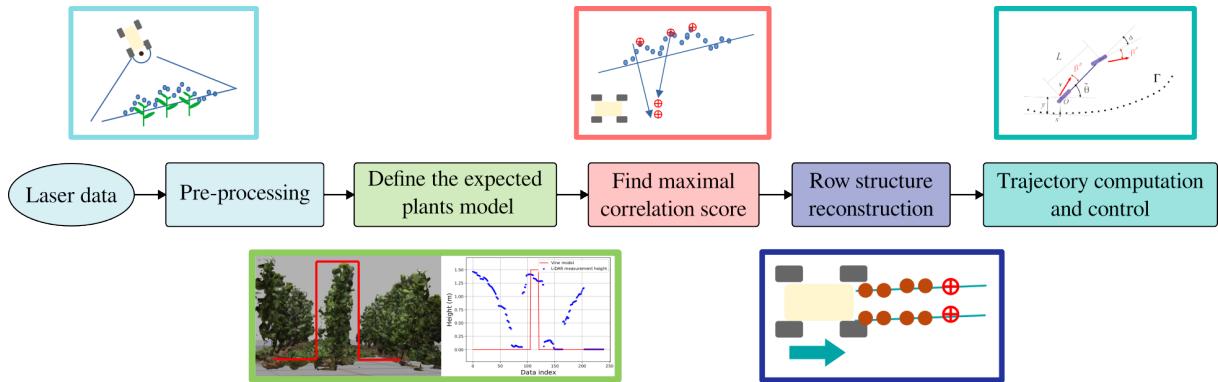


Figure 7: Architecture générale du furrow, détaillée en 5 étapes

Les étapes détaillées dans la figure 7 se déroulent de la façon suivante :

1. On reçoit la donnée LiDAR, on la borne en angle et on la convertit dans le repère du tracteur
2. On définit la modèle de plante attendu dans le scan LiDAR
3. On calcule la corrélation entre le modèle et le scan LiDAR afin de trouver les points les plus représentatifs des plantes : les points d'intérêt
4. Avec les points d'intérêt, on trouve les rangées gauche et droite qui forment la planche à suivre
5. On calcul la trajectoire à suivre par le tracteur, puis on génère les consignes d'écart latéral et angulaire

L'algorithme a un fonctionnement itératif, chaque opération dépend des opérations précédentes. Sur la visualisation de l'exécution de notre algorithme (figure 8), on peut observer les différentes étapes composant le processus.

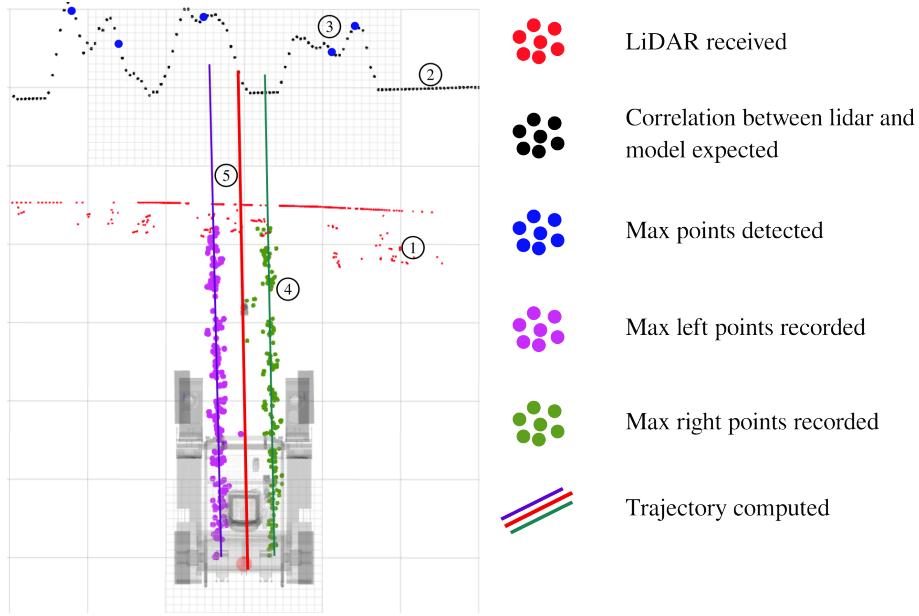


Figure 8: Visualisation des éléments composants l'algorithme du furrow, numérotation correspondant aux étapes décrites plus haut

2.3 Limitations du Furrow

L'algorithme du furrow, initialement conçu pour le suivi de traces de roues et ultérieurement adapté pour prendre en compte la présence de plantes, n'est pas optimal dans la tâche qui lui est confiée. Bien qu'il se montre globalement robuste dans des conditions réelles nominales, il n'est pas à l'abri de certaines limitations qui réduisent ses niveaux d'autonomie et de fiabilité. L'algorithme nécessite un réglage manuel fixe de paramètres, pas forcément adapté à l'entièreté de la rangée de plantes, sa détection est peu fiable quand les plantes sont petites, ou avec la présence de mauvaise herbes, et pour finir, il a tendance à cumuler ses erreurs et à changer son suivi pour une rangée de plantes adjacentes. Par ces problématiques, le but de nos travaux sera d'augmenter les niveaux d'autonomie, de robustesse et de précision du suivi de plantes.

Réglage manuel

Pour détecter les plantes, le furrow nécessite une supervision humaine, on doit lui donner les bons paramètres correspondant aux rangées de plantes à suivre. En effet, on doit définir le modèle *a priori* de ce que l'on est censé recevoir dans le scan LiDAR. On spécifie la hauteur et largeur des plantes, ainsi que la largeur de la planche à suivre. Le réglage de modèle attendu conditionnant la corrélation, et donc les plantes détectés, une configuration automatique de ces paramètres permettrait d'obtenir un algorithme plus autonome, plus robuste et plus précis (figure 9).

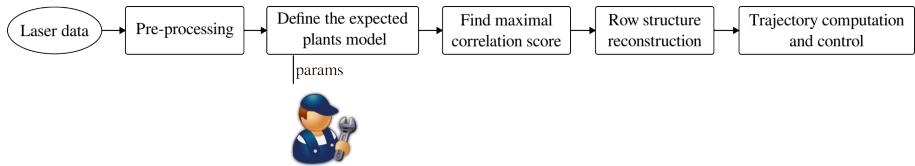


Figure 9: Nécessité de supervision humaine pour déterminer les paramètres adéquates au bon fonctionnement de l'algorithme

Cette problématique fera l'objet de l'architecture 1, où l'on demandera à un réseau de neurones de régler automatiquement les paramètres de détection du furrow, à partir du nuage de points donné par le LiDAR.

Mauvaise détection

La détection de plantes par corrélation avec un modèle statique présente plusieurs problématiques. Les fausses détections sont courantes, n'étant pas basé sur la géométrie globale de la rangée de plante, mais uniquement la corrélation d'un modèle avec la dernière nappe LiDAR disponible. Par conséquent, l'algorithme présente des difficultés à différencier les mottes de terres des plantes et à détecter les plantes trop petites dans l'environnement (figure 10). On a également l'impossibilité de prendre en compte plusieurs rangées de tailles différentes. On ne peut régler le suivi que d'un seul type de plantes, et par conséquent, si jamais on a 2 rangées différentes de plantes à suivre, uniquement une seule rangée sera correctement détectée.



Figure 10: Prise de vue réelle d'un champ, cas de petites plantes par rapport aux bosses de terre, ainsi qu'une plantation asymétrique

Cette problématique fera l'objet de l'architecture 2, où l'on demandera à un réseau de neurones de nous donner les points d'intérêt correspondant aux plantes, à partir du nuage de points donné par le LiDAR.

Accumulation d'erreurs

L'algorithme du furrow utilise une approche de création de trajectoire dite "optimiste", ce qui signifie qu'il prend en compte chaque information précédemment acquise pour la génération de la trajectoire. La classification des points d'intérêt en rangées de plantes se fait en cherchant le plus proche voisin du point

d'intérêt précédent. Cette méthode est efficace tant que des points proches des rangées de plantes peuvent toujours être trouvés. Cependant, si une anomalie se présente dans la détection des rangées de plantes, la sélection de point peut diverger et créer une trajectoire incorrecte (figure 11).

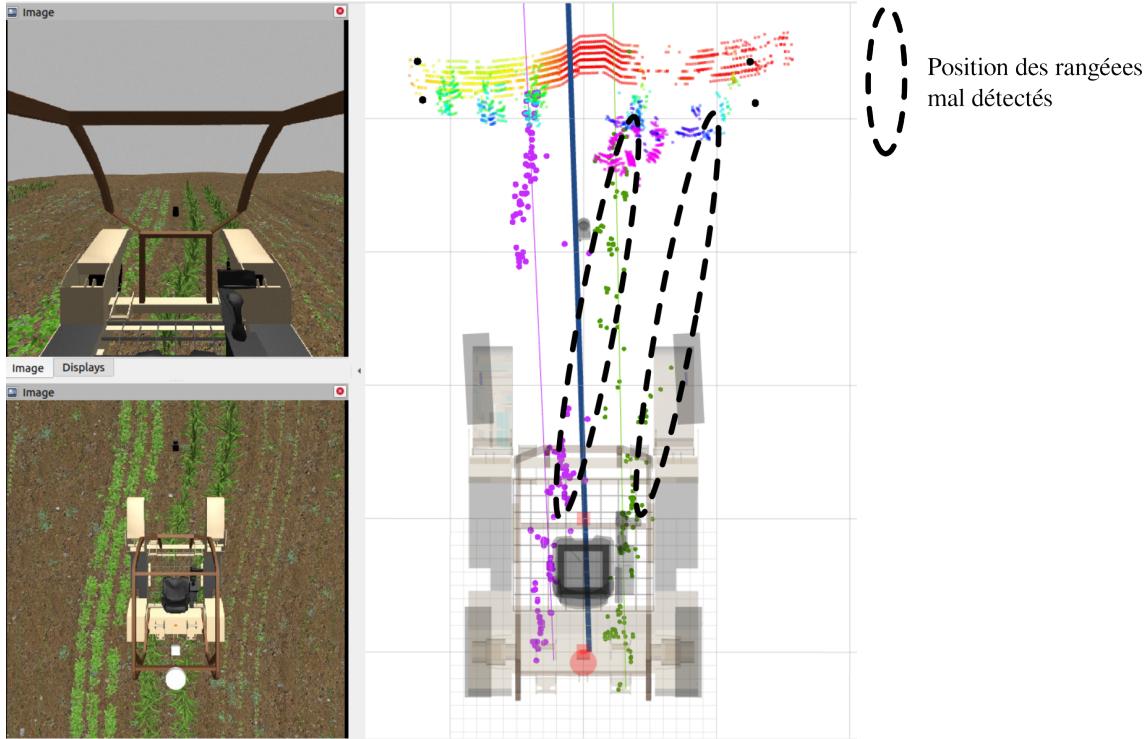


Figure 11: Cas de divergence dû à des erreurs passées, l'algorithme prend la mauvaise décision de changer de rangées

Malgré ses défauts, ce fonctionnement par boucle de rétroaction est nécessaire pour la prise en compte de l'évolution de la courbures des rangées agricoles, les nouvelles detections doivent prendre en compte les rangées précédemment détectés. Si jamais on sélectionne une rangée de plantes différentes, ce choix risque de nous mener à encore plus d'erreurs par le futur. Il n'est donc pas possible d'améliorer cet algorithme sans changer complètement son paradigme de fonctionnement.

Cette problématique fera l'objet de l'architecture 3, où l'on demandera à un réseau de neurones de nous donner directement l'écart latérale et angulaire à la trajectoire formée par les rangées de plantes, à partir du nuage de points donné par le LiDAR.

Finalement, cette problématique n'ayant pas donné de résultats concluants par l'utilisation d'un réseau de neurones, nous nous dirigerons vers une architecture 4, où la création de trajectoire se fera par une analyse probabiliste de la répartition du nuage de points d'intérêts décrivant les rangées de plantes : recherche des rangées de plantes par mixture de gaussiennes.

2.4 Apport de l'intelligence artificielle

L'introduction de l'intelligence artificielle offre un potentiel prometteur pour améliorer considérablement la perception artificielle dans le domaine de la robotique agricole. Tel qu'exposé précédemment, plusieurs problématiques peuvent être résolues afin d'optimiser l'algorithme du furrow. L'intelligence artificielle nous offre ainsi une opportunité significative d'améliorer les capacités de perception de nos systèmes robotiques agricoles, ouvrant la voie à des avancées notables dans ce domaine en constante évolution.

Le domaine de l'intelligence artificielle est vaste : dans ce projet, par l'utilisation classique de nos algorithmes d'apprentissage, nous nous concentrerons sur l'utilisation de réseau de neurones en apprentissage supervisé.

Réseau de neurones

Un réseau de neurones artificiels est un modèle mathématique inspiré du fonctionnement du cerveau humain. Il est utilisé pour résoudre des problèmes complexes d'apprentissage automatique et de traitement de données. Le réseau est composé de couches de neurones inter-connectés, où chaque neurone effectue des opérations mathématiques sur les données qu'il reçoit et transmet le résultat à d'autres neurones. Les couches sont reliées les unes aux autres par des matrices de paramètres correspondant aux poids des connexions entre les neurones, dans le processus d'apprentissage, ce sont les connexions qui se modifient afin d'obtenir l'opération adéquate.

Le processus d'apprentissage et d'utilisation d'un réseau de neurones implique deux phases principales :

- Entraînement : Pendant cette phase, le réseau de neurones est alimenté avec un ensemble de données d'entrée et les réponses attendues correspondantes. Le réseau ajuste automatiquement les poids et les biais de ses connexions pour minimiser l'écart entre les prédictions qu'il produit (noté \hat{Y}) et les réponses réelles (noté Y).
- Inférence : Une fois que le réseau est entraîné, il peut être utilisé pour effectuer des prédictions sur de nouvelles données sans étiquettes. Les données d'entrée (noté \hat{X}) sont propagées à travers le réseau, chaque neurone effectuant des calculs sur les données et transmettant les résultats à la couche suivante. La sortie finale du réseau est la prédition résultante (\hat{Y}).

Par exemple, pour un réseau de neurones de 4 couches de respectivement 3,8,8 et 2 neurones, on aura pour entrée la première couche soit 3 scalaires, en sortie la dernière couche soit 2 scalaires (figure 12).

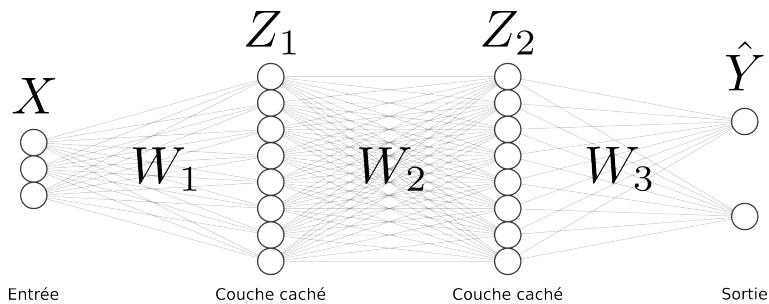


Figure 12: Réseau de neurones à 2 couches cachées (sans biais, ni fonctions d'activations)

En terme de dimensions, on aura pour entrée $\underbrace{X}_{1,3}$ et en sortie $\underbrace{\hat{Y}}_{1,2}$. Tous le processus du réseau de neurones tourne autour des connexions entre les neurones, représenté par des matrices de dimensions (Ne, Ns) avec Ne pour le nombre de neurones en entrée, et Ns le nombre de neurones en sortie. Nous aurons donc en paramètres $\underbrace{W_1}_{3,8}$, $\underbrace{W_2}_{8,8}$, $\underbrace{W_3}_{8,2}$, et enfin, l'opération permettant d'obtenir notre sortie de réseau de neurones sera donné par l'opération :

$$\hat{Y} = X * W_1 * W_2 * W_3 \quad (1)$$

Dans ce cas-ci, chaque sortie serait uniquement une somme pondérée de chaque entrée. En réalité, on a recours à l'utilisation de fonctions d'activations : elles se placent généralement à la sortie de chacune des couches et permettent d'introduire de la non-linéarité dans l'opération finale du réseau de neurones.

Leurs placements sur les 2 couches cachées de 8 neurones nous donneraient l'opération suivante (avec $\sigma(x)$ une fonction d'activation quelconque) :

$$\hat{Y} = \sigma(\sigma(X * W_1) * W_2) * W_3 \quad (2)$$

Déterminer les paramètres idéaux d'un réseau de neurones fait partie de tout l'enjeu de l'apprentissage supervisé, ce sont les paramètres acquis pendant l'apprentissage qui permettront au réseau de neurones d'avoir une prédiction proche de la réalité. Nos paramètres sont définis par la liste de matrices W, W^* correspond aux paramètres idéaux pour une tâche spécifique. De multiples techniques existent pour l'entraînement, elles partent toutes du même principe : réduire au maximum l'erreur entre la réalité et la prédiction du réseau de neurones. On définit un critère de correspondance appelé fonction de perte : elle compare la prédiction et la réalité afin de nous donner un scalaire correspondant à leur proximité, une perte minimale implique donc que la prédiction est très proche de la réalité (notre objectif).

Une fonction de perte assez répandue est l'écart quadratique moyen entre la prédiction et la réalité (Mean square loss) :

$$Loss_{mse}(\hat{Y}, Y) = \frac{1}{N} \sum_{i=0}^N (\hat{Y}_i - Y_i)^2 \quad (3)$$

Avec N correspondant au nombre de sorties du réseau de neurones, taille des vecteurs \hat{Y} et Y .

Comme décrit précédemment, l'entraînement se fait à l'aide d'une base de données des pairs entrée-sortie connues. Dans notre cas, on aura une base de données d'une multitude de pairs entrée-sortie, de 3 scalaires pour l'entrée et de 2 scalaires pour la sortie. Finalement, notre but sera d'obtenir pour un maximum d'échantillon pour la base de données une prédiction proche de la réalité :

$$Prdiction(X, W^*) = \hat{Y} \approx Y \quad (4)$$

Une manière de minimiser la fonction de perte est de faire évoluer itérativement les paramètres pour réduire la fonction de perte à chaque changement de paramètres du réseau de neurones :

$$Loss_{mse}(NN(X, W^{k+1}), Y) \leq Loss_{mse}(NN(X, W^k), Y) \quad (5)$$

Ce fonctionnement permet de rapprocher progressivement les prédictions de notre réseau de neurones de la réalité, ce qui lui permet d'acquérir au fur et à mesure une compréhension de plus en plus correcte de la donnée qu'il perçoit.

Les réseaux de neurones sont employés dans de nombreuses applications, notamment la vision par ordinateur (reconnaissance d'images, détection d'objets), le traitement du langage naturel (traduction automatique, génération de texte), les systèmes de recommandation, la modélisation prédictive et bien d'autres domaines. Ils peuvent apprendre à partir de grandes quantités de données et extraire des modèles complexes, ce qui les rend extrêmement polyvalents pour résoudre une variété de problèmes.

3 Mise en place de l'environnement de simulation

Dans ce projet, afin de mettre en place nos algorithmes, nous aurons besoin d'un environnement de simulation nous permettant l'implémentation et l'expérimentation de nos solutions logiciels. L'utilisation de la plate-forme de développement logiciel ROS [6] nous permettra d'avoir la même exécution en simulation qu'en réel. La différence se fera au niveau des capteurs et des actionneurs, en réel, ces données viendront directement du matériel du tracteur, alors qu'en exécution virtuelle, ces données seront issus de capteurs et d'actionneurs de l'environnement de simulation Gazebo [9].

L'utilisation de cette simulation nous permettra d'obtenir diverses données. Tout d'abord, il y aura les données en entrée de notre algorithme, ce sont les données issues des capteurs (LiDAR, odométrie, GPS, IMU, commande utilisateur). Nos algorithmes se concentreront en particulier sur le nuage de points qui résulte du LiDAR et de l'odométrie.

Ces données sont accessibles de la même manière en situations réelles ou simulées, on les obtient à partir des capteurs installés sur le tracteur. Nous utiliserons un jumeau numérique de notre tracteur afin de faciliter la migration de nos algorithme de la simulation vers l'expérimentation réel. Par conséquent, nos algorithmes fonctionneront de la même manière en réel qu'en simulation.

L'utilisation de jumeaux numériques dans le domaine de la robotique est largement répandue, en particulier pour les applications de perception et de contrôle, ce qui en fait un outil idéal pour le domaine de la robotique agricole ([8]). Cela permet d'effectuer de multiples tests dans un environnement contrôlé, sans endommager les potentiels prototypes ou l'environnement d'expérimentation. Ce choix technique a également été motivé par l'existence d'une implémentation préexistante du projet en simulation.

Bien que l'utilisation d'une base de données réelles aurait pu être envisageable pour confronter directement nos algorithmes à de la donnée réelle plutôt que de la donnée simulée, il existe assez peu de base de données adaptée au domaine agricole, et encore moins en perception basée LiDAR (un des seuls avec nuages de points dans le domaine utilise une simple Kinect, caméra de profondeur grand public [3]). Il y a également trop de paramètres propres à chaque engin et à chaque configuration de plantes, ce qui rend difficile la proposition d'une base de données efficace pour les chercheurs du monde entier, chaque laboratoire doit personnaliser ses données afin d'obtenir les informations spécifiques dont ils ont besoin [8]. Nous avons donc créé un environnement de simulation facilement reconfigurable, le couvert végétale et le comportement du tracteur seront générés procéduralement.

3.1 Crédation de base de données

Pour répondre à la diversité des configurations agricoles possibles et à l'exigence d'adaptabilité de notre algorithme, nous utiliserons un jumeau numérique (sous l'environnement Gazebo [9]). Cela nous permettra de générer des données dans un environnement simulé, ainsi que, comme mentionné précédemment, avoir un environnement de simulation pseudo-réaliste où l'on utilisera le même tracteur avec les mêmes algorithmes en situations réelles (figure 13).

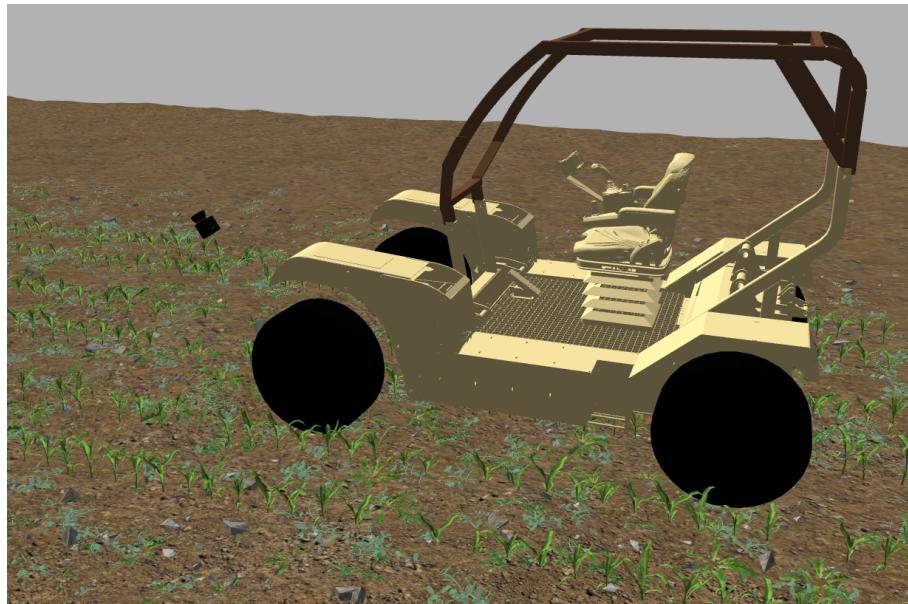


Figure 13: Environnement le plus réaliste simulé, avec mauvaises herbes, cailloux, sol irrégulier, avec des plantes de positions bruitées, simulation sous Gazebo

La simulation a été conçue pour être hautement modifiable, permettant ainsi d'adapter notre environnement, la génération des plantes et les données recueillies. L'objectif est de produire des données adaptées à l'entraînement de nos réseau de neurones.

Pour entraîner et évaluer nos réseaux de neurones, nous avons besoin de la vérité terrain annotées. Ces données sont difficiles et fastidieuses à obtenir en situations réelles (figure 14). Nous devrions mesurer manuellement la taille et la position de chacune des plantes. Nous privilégierions donc l'utilisation d'une simulation, où nous avons accès directement à ce genre d'informations (position et configuration des plantes, position du robot...).

La disponibilité de ces données est un enjeu crucial pour nos algorithmes, car contrairement à la simulation, nous n'avons pas de moyen direct d'accéder à la réalité terrain en temps réel. C'est là que réside la principale tâche de nos algorithmes de perception : prédire une réalité à partir de données brutes, comprendre et appréhender son environnement. Dans l'exécution finale, ces données ne seront pas accessibles directement, elles devront être estimées à partir du nuage de points 3D.

Pour guider la création du monde virtuel, nous nous sommes inspirés autant que possible du monde réel afin d'obtenir une simulation cohérente avec nos sites d'expérimentations (figure 14).



Figure 14: Mesure des paramètres principaux des plantes, afin d'obtenir une base de données similaires à nos sites d'expérimentations

Nous nous sommes également inspiré des mauvaises herbes et de divers débris pour pouvoir mettre en difficultés nos algorithmes afin d'éprouver leurs avantages et inconvénients. Malgré cela, notre simulation reste encore loin de la réalité, il est difficile de reproduire en temps réel un monde réaliste avec toutes les interactions que ça implique. Pour pouvoir comprendre les différences et similarités entre la simulation et le monde réel, nous avons réalisé une reconstruction 3D du milieu agricole (figure 15).

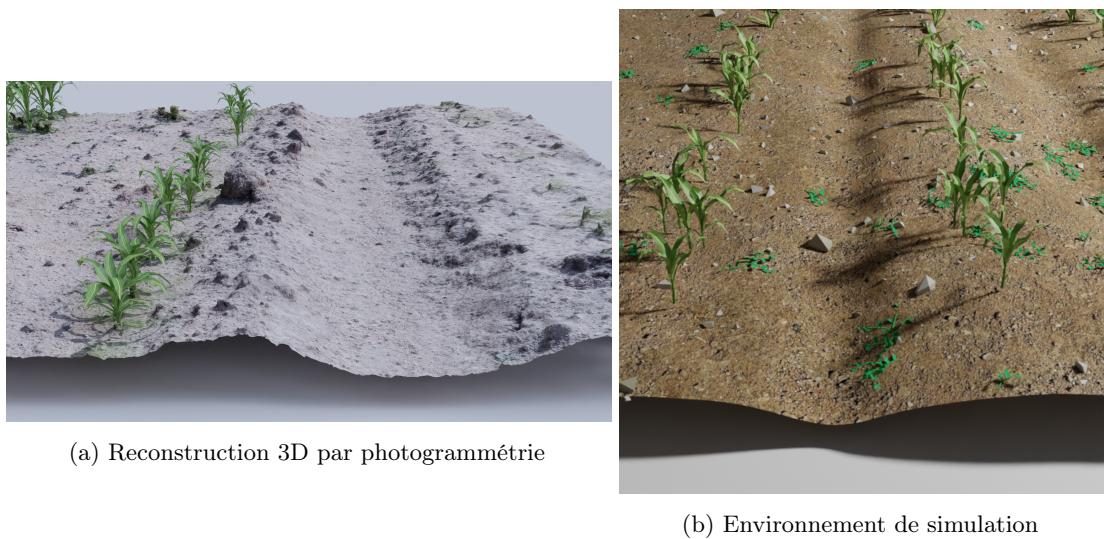


Figure 15: Comparaison entre la réalité et la simulation

Malgré les différences entre le monde réel et la simulation, l'environnement virtuelle permettra d'introduire le même type de perturbations pour nos algorithmes.

Une fois notre environnement réalisé, on simule plusieurs scénarios de guidage de tracteur afin de générer une multitude de données différentes. Afin d'avoir un comportement le plus représentatif du fonctionnement finale, on va définir deux scénarios :

- Aléatoire : le tracteur effectue des trajectoires aléatoires
- Convergence : le tracteur rejoint la trajectoire à partir d'une position donnée

Concernant la disposition des plantes, on peut faire varier la taille, le type de plantes, la largeur de planche et le nombre de rangées (figure 16). Nous avons également des paramètres de bruit, qui affecte l'apparition, la position et la taille des plantes (également visible sur la figure 16 et 17, les plantes ne sont pas parfaitement alignées ou apparues).

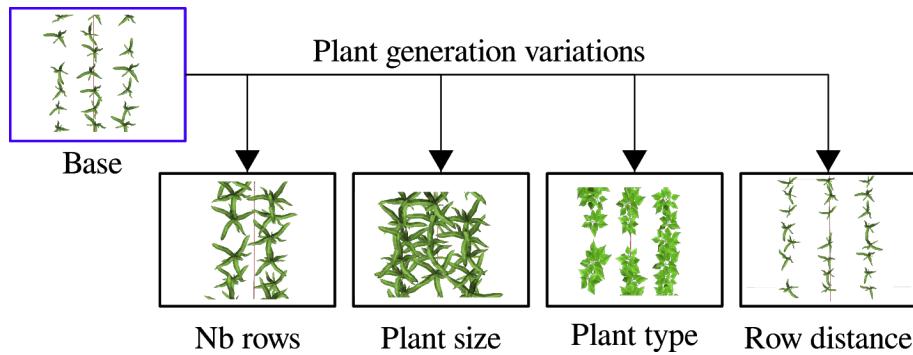


Figure 16: Paramètres utilisé pour la générations des plantes

Finalement, nos simulations de générations de données ressembleront à la figure 17 : une simulation d'un tracteur face à des plantes, avec divers scénarios de comportement de navigation et de disposition des plantes.

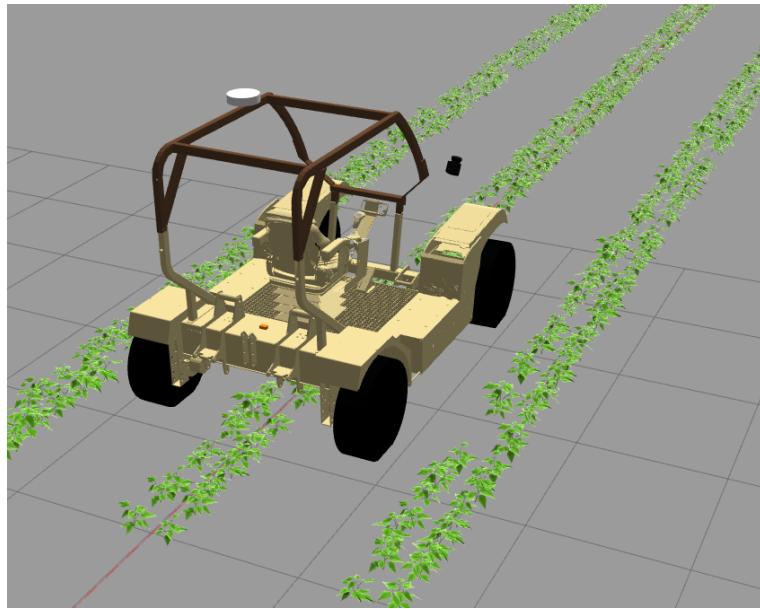


Figure 17: Simulation d'un scénario agricole, sans sol pseudo-réaliste

Lorsque nous établirons la base de données d'apprentissage, nous procéderons de manière similaire pour créer une base de validation destinée à évaluer nos réseaux neuronaux à l'aide de nouvelles données. Cette base de validation nous permettra de tester la capacité prédictive de nos réseaux de neurones face à des informations nouvelles, en dehors du cadre de leur apprentissage. L'objectif est de déterminer si ils parviennent à généraliser leur apprentissage à des éléments inconnus, le risque est que le réseau de neurones ait appris par cœur des règle pour sa base d'entraînement sans pour autant réellement comprendre la donnée, c'est le phénomène de sur-apprentissage (over-fitting).

Nous avons la possibilité de relancer la simulation avec différents paramètres ajustés ou modifiés, ce qui nous permet d'obtenir la base de données requise pour nos expérimentations. Dans le domaine du traitement de données agricoles, beaucoup de base de données issus du réel ([10], [14]) sont décrites par les paramètres de leur couvert végétal (répartition de leurs tailles en hauteur et largeur de tronc, carte du type de plante...). Dans ce projet, contrairement à ces bases de données, nous ne fournirons peu de détails sur la base de données utilisée, notre attention sera principalement portée sur la manière de générer et d'utiliser ces données pour entraîner et évaluer nos algorithmes.

3.2 Méthode d'évaluation de nos algorithmes

Nous allons mettre en place divers méthodes d'évaluations qui ont pour objectif de nous aider à comprendre les améliorations apportées lors de ce projet, afin de les exploiter au maximum et ainsi renforcer l'autonomie, la robustesse et la précision de nos algorithmes.

3.2.1 Évaluation de robustesse

Une grande partie de nos efforts se concentrera sur l'amélioration de la robustesse de nos algorithmes en matière de détection. Nous devrons donc procéder à une évaluation quantitative du nombre de rangées de plantes que l'engin agricole aura réussi à suivre.

Pour nos évaluations, nous utiliserons deux mondes spécifiques. Le monde de validation, sera constitué de différentes configurations de plantes, avec un sol relativement plat et exempt de mauvaises herbes. L'objectif de ce monde sera de vérifier le bon fonctionnement de notre algorithme dans des situations d'utilisation courantes, tout en incluant la difficulté de gérer de très petites plantes à certains moments (figure 18).

Le monde de test, sera conçu pour mettre nos algorithmes à l'épreuve. Il sera composé d'un sol irrégulier et de mauvaises herbes, recréant ainsi des conditions difficiles similaires à celles rencontrées dans des situations réelles. Ce monde mettra en évidence les défis auxquels nos algorithmes pourraient être confrontés dans des environnements agricoles réels, nous permettant ainsi d'identifier les points à améliorer pour assurer une meilleure performance dans des conditions plus complexes (figure 18).

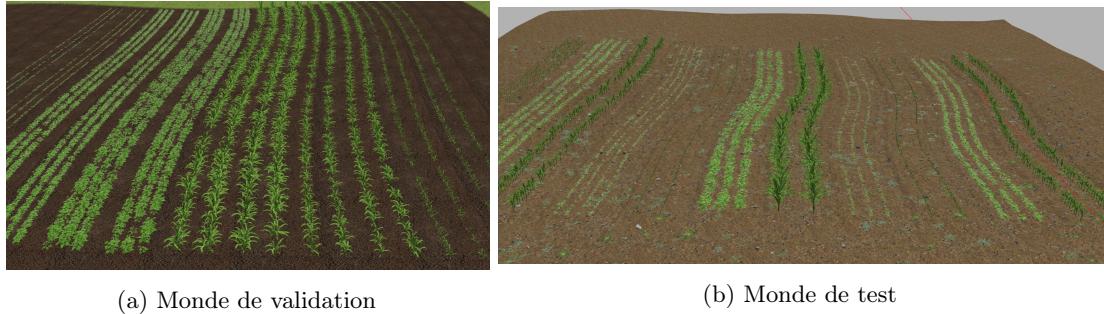


Figure 18: Environnement de test de robustesse

Notre robustesse de détection sera défini par la quantité de rangées de plantes correctement suivies. Notre taux de résolution sera défini par, pour n rangées de plantes, combien ont été suivis à 20cm près (figure 19).

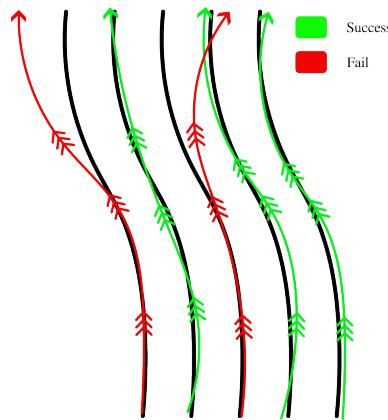


Figure 19: Évaluation en proportion des trajectoires correctement suivies

3.2.2 Évaluation de précision

Dans le cadre de l'amélioration et du réglage de nos algorithmes de création et de suivi de trajectoires, il est essentiel d'évaluer leurs précisions dans divers scénarios. Pour cela, nous devons examiner la qualité du suivi de trajectoire, c'est-à-dire évaluer dans quelle mesure notre algorithme parvient à minimiser l'écart par rapport à la trajectoire souhaitée.

Afin de représenter au mieux la diversité des configurations possibles en termes de trajectoires, nous avons créé trois scénarios distincts (figure 20). On aura pour scénario :

- La courbure légère
- La courbure forte
- L'oscillation

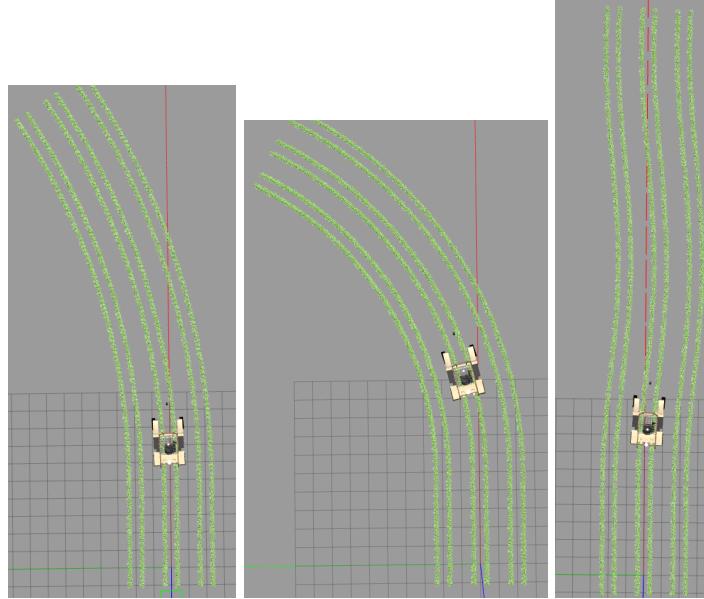


Figure 20: Schéma des différents scénarios de création de base d’entraînement, respectivement de gauche à droite, courbure légère, courbure forte et oscillation

3.2.3 Enregistrement de données et expérimentation réel

Nos expérimentations en réel ont l'avantage de pouvoir réaliser des enregistrements de données acquises en temps réel. En effet, la répétabilité des situations est difficile dans la robotique agricole, certaine situations peuvent être spécifiques à un terrain, un tracteur, un capteur, une météo, un type de sol, un type de plante.... Nous devons par conséquent d'enregistrer un maximum de données d'un maximum de scénarios différents, pour pouvoir mettre nos nouveaux algorithmes dans les situations qui les avaient mis en difficultés par le passé. Lors d'un rejet de données réelles, il est impossible d'interagir avec l'environnement, nous sommes restreint à examiner seulement les entrées et sorties des algorithmes qui ont été exécutés durant l'expérimentation réelle, ainsi qu'exécuter nos nouveaux algorithmes sur ces données (figure 21).

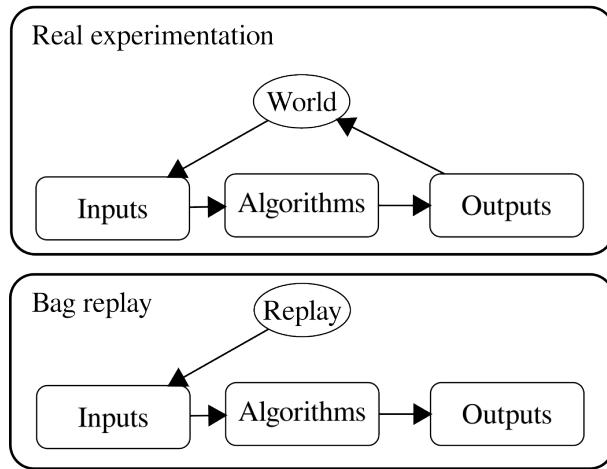


Figure 21: Schéma du fonctionnement d'un rejet de données réels

Une fois que notre algorithme fonctionne selon nos attentes dans un monde simulé, nous le soumettons à des rejeux de données acquises en temps réel pour une mise à l'épreuve dans le but d'identifier ses défauts et difficultés.

Une fois que notre algorithme a été validé en simulation et lors du rejet de données acquises en temps réel, nous le testons sur terrain réel, en conditions d'exploitation réelle (figure 22). Par la nature imprévisible de ces expérimentations, diverses mesures de sécurité sont mises en place, le tracteur se déplace à une vitesse limitée, avec un arrêt d'urgence constamment à portée de mains. Tout cela nous permet de prévoir le cas où notre algorithme rencontrera une anomalie et sortirait complètement de son fonctionnement nominal. Cette expérimentation permet d'obtenir une validation finale de toutes notre architecture de perception et de contrôle.



Figure 22: A gauche, visualisation de la perception issue de l'algorithme, à droite environnement d'expérimentations sur le site de Montoldre dans le département de l'Allier

4 Architecture 1 : Détection de configuration agricole

Pour rappel, l'algorithme de suivi de furrow à besoin d'un réglage manuel pour pouvoir fonctionner correctement. En effet, en fonction du type de végétation, de la hauteur des plantes et de leurs dispositions, nous aurons besoin de définir des paramètres afin que le furrow puisse sélectionner les points d'intérêts dans le nuage de point afin de définir une trajectoire à suivre. Ce réglage va intervenir à l'étape « Define the expected plants model » (figure 23).

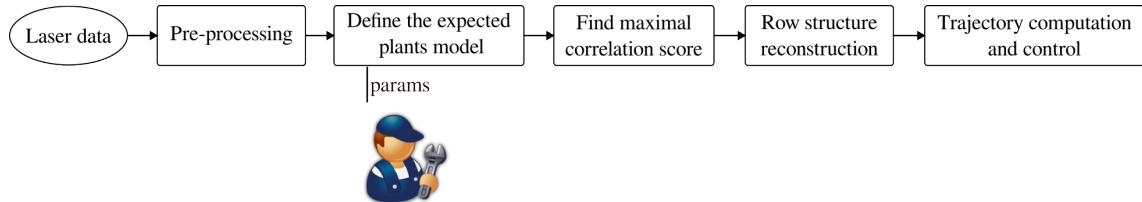


Figure 23: Architecture général du furrow, nécessitant un réglage manuel

Les paramètres nécessaires pour décrire les rangées de plantes à suivre sont la largeur et la hauteur des plantes, ainsi que la largeur de planche (figure 24).

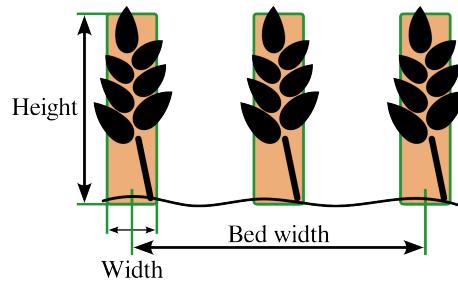


Figure 24: Modèle de réalité terrain, défini par la hauteur et la largeur des plantes, ainsi que la largeur de la planche de plantes

Ces paramètres nous permettront de définir le modèle de recherche par corrélation avec le scan LiDAR des plantes, de ce que l'on appellera points d'intérêt des plantes, points correspondants à la position la plus représentative, points perçus par le LiDAR au centre de la plante (figure 25).

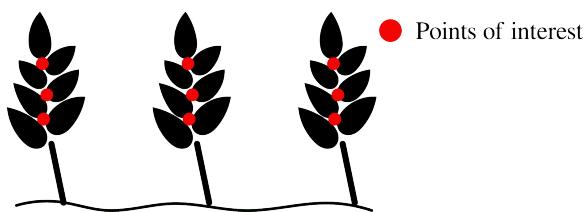


Figure 25: Points d'intérêt des plantes, représentatif de leurs centres

Dans la première architecture, nous allons nous concentrer sur l'élaboration d'un estimateur de réalité terrain par réseau de neurones. Il devra, depuis la donnée LiDAR, en déduire les paramètres décrivant la disposition et la taille des plantes. Cette estimation nous permettra de définir le modèle des plantes attendues (figure 26).

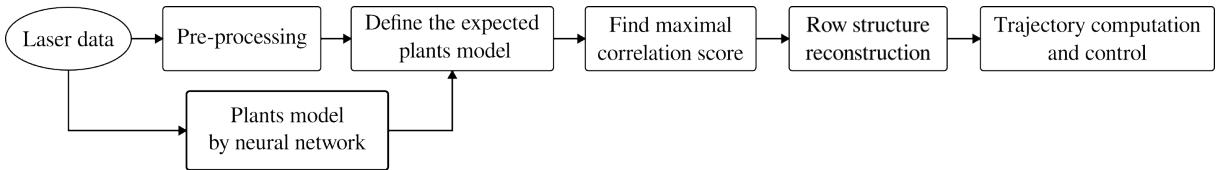


Figure 26: Fonctionnement du furrow avec l'estimation de paramètres par prédiction d'un réseau de neurones

Cette première architecture à un double objectif, dans un premier temps, la prise en mains des différentes technologies composant ce projet tel que la simulation sous ROS [6], la création d'une base de données, l'implémentation d'un réseau de neurones de traitement de nuage de points 3D. Dans un second temps, la viabilité technique de l'utilisation de l'intelligence artificielle pour l'amélioration de notre algorithme, que ce soit pour son entraînement, son exécution ou ses performances de prédiction, le tout en temps réel.

4.1 Traitement nuage de points

Nous effectuerons le traitement du nuage de points en deux étapes (figure 27), tout d'abord, nous traiterons notre entrée brute afin de l'exprimer d'une manière plus synthétique et explicite du point de vue du réseau de neurones, c'est l'étape d'extraction de caractéristiques. Puis nous traiterons ces caractéristiques afin d'effectuer notre prédiction, c'est la régression. Dans notre cas, on prédit des paramètres correspondant à la réalité terrain (figure 24).

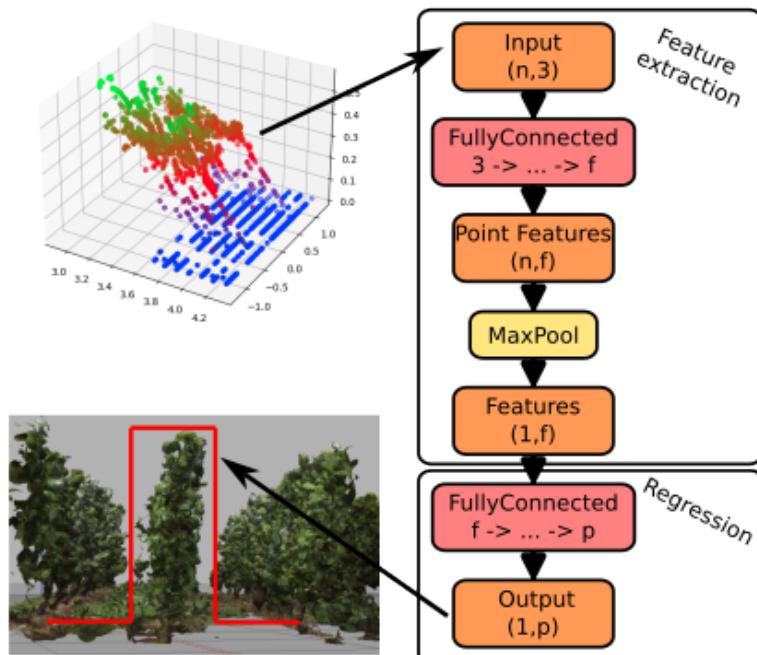


Figure 27: Présentation du processus du réseau de neurones

- Extraction des caractéristiques du nuage de points :

On traite indépendamment les N points afin de les projeter en F caractéristiques, on obtient un tableau de caractéristiques en $(N,F,)$

On prend le maximum des F caractéristiques à travers les N points, on obtient un vecteur de caractéristiques en $(F,)$

- Régression des caractéristiques en paramètres :

On traite les F caractéristiques afin d'obtenir les paramètres décrivant la réalité terrain, on obtient un vecteur de sortie en $(P,)$

Les processus d'extraction de caractéristiques et de régression sont susceptibles de varier en terme d'opérations neuronales, on peut viser à obtenir des résultats plus rapides, et/ou plus précis, et/ou plus robuste.

Par nos expérimentations détaillées dans la partie 4.2, nous nous dirigerons finalement vers une architecture à 2 couches d'extraction de caractéristiques et 2 couches de régression (figure 28).

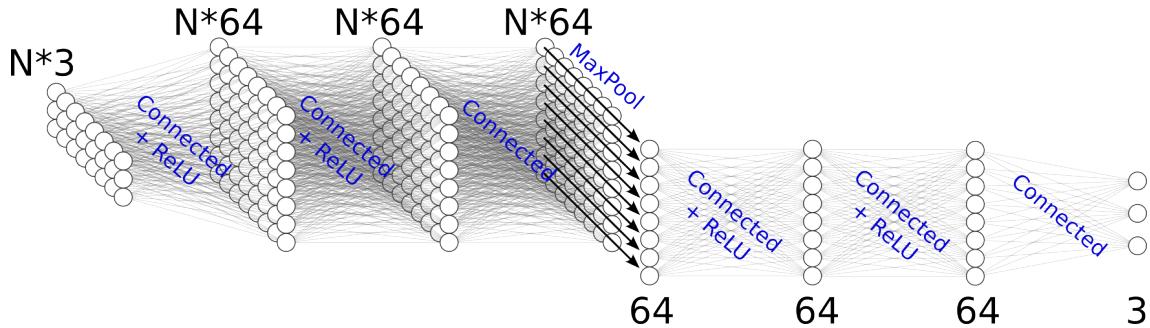


Figure 28: Architecture final du réseau de neurones utilisé, inspiré du PointNet [2], un PointNet simplifié

4.2 Choix du réseau de neurones

Pour pouvoir choisir notre réseau de neurones, nous allons évaluer la performance comparée à sa taille, le but étant d'avoir le meilleur compromis entre la rapidité d'exécution et la précision de la prédiction. La complexité sera liée au nombre de couches cachées, nous allons faire varier le nombre de couches pour l'extraction des caractéristiques, ainsi que pour la régression. Finalement, notre performance sera évaluée par la capacité de notre réseau de neurones à minimiser l'écart entre sa prédiction et la réalité, pour l'ensemble de sa base de données, on utilisera donc le résultat moyen de la fonction de perte, moyenne des erreurs absolus (figure 29).

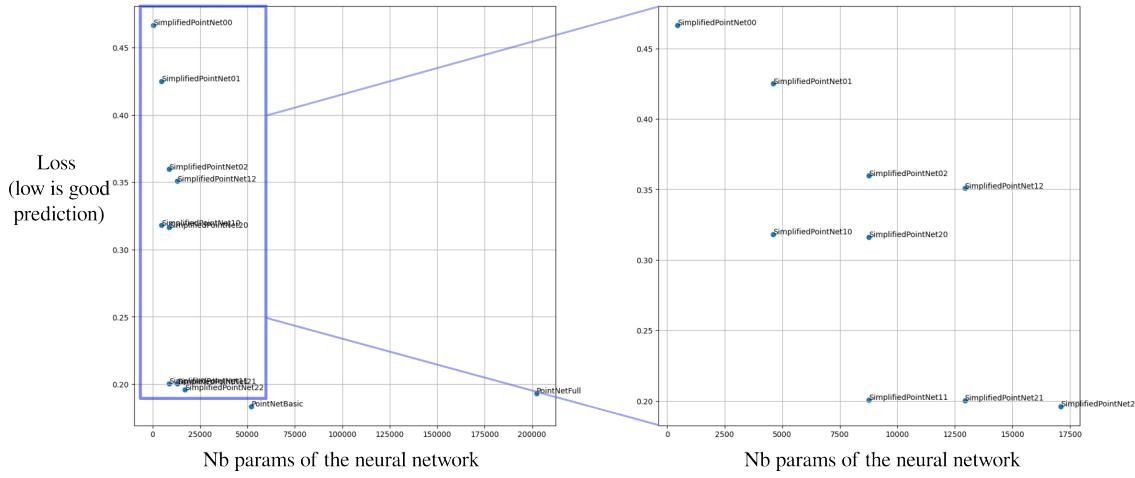


Figure 29: Comparaison des réseaux de neurones en fonction de leurs tailles et de leurs performances, à gauche, comparaison de nos réseaux de neurones avec les PointNet originaux, à droite, focus sur les réseaux de neurones étudiés dans l’architecture 1

Le nom des réseaux de neurones correspond au nombre de couches cachées pour respectivement l’extraction de caractéristiques et la régression. Le réseau de neurones qui obtiendra les meilleures performances sera le « SimplifiedPointNet22 », 2 couches cachées d’extraction de caractéristiques et 2 de régression. C’est en effet celui avec les plus de couches cachées qui réussit au mieux à former des prédictions précises à partir d’une donnée brute. Ce n’est pas une règle générale, mais pour un réseau de neurones d’opérations classiques, plus il y en a de couches cachées et de neurones, plus le réseau de neurones sera capable de comprendre une donnée complexe, et par conséquent effectuer des prédictions performantes.

Nous évaluons également des réseaux de neurones plus complexes comme le PointNetBasic et le PointNetFull, respectivement la version allégée et la version complète du PointNet [2] (figure 29).

Le PointNetBasic obtient les meilleures performances mais au prix d’un nombre de paramètres bien plus élevé. Par facilité d’entraînement et d’exécution, nous resterons finalement le SimplifiedPointNet22, qui est le meilleur compromis entre performances et poids d’exécution.

Pour l’utilisation de réseaux de neurones plus complexes tel que le PointNet++ [15] ou le VoxelNet [16], nous ne partirons pas dans cette direction pour les raisons suivantes :

- Une implémentation de ces algorithmes en temps réel plus délicate
- Notre problème à traiter ne nécessite pas une grande complexité de traitement
- Garder une implémentation simple et accessible

Une multitude de réseaux de neurones existent dans le domaine du traitement de nuages de points 3D, ils varient en type d’opération, en profondeur, ainsi qu’en nécessité de réglage manuel. Par exemple, le VoxelNet [16] va découper le nuage de points en voxels de taille prédéfinie par l’utilisateur afin d’effectuer une extraction de caractéristiques à la fois local et global. Par l’efficacité du PointNet [2], ainsi que le réglage des autres types de traitement qui limite la reproductibilité de nos expérimentations (le réglage empirique de l’algorithme ayant un fort impact sur son résultat), nous n’utiliseront pas d’autres réseaux de neurones que le PointNet simplifié.

Pour toutes ces raisons, nous ferons l’hypothèse que si notre réseau de neurones réussit à correctement répondre à nos attentes, un réseau de neurones plus complexe ne permettrait pas de mieux réaliser la tâche. Au contraire, l’algorithme d’apprentissage pourrait sur-convergé, il apprendrait par cœur les données de l’entraînement, plutôt que de comprendre réellement la végétation, il serait en sur-apprentissage.

Voici un récapitulatif des paramètres de la base de données et de l’entraînement :

Table 1: Paramètres généraux

Paramètre	Valeur
Nombre de lectures de la base de données dans le processus d'apprentissage du réseau de neurones (Epochs)	20
Nombre de point par nuage de points	1610
Nombre de nuage de points d'entraînement	128668
Durée d'acquisition de la base de données	3h40
Fréquence d'acquisition de la reconstruction 3D	10hz
Taux d'apprentissage (Learning rate)	0.001
Fonction de perte	Moyenne des erreurs absolu (MAE Loss)

4.3 Expérimentations

Pour évaluer la validité de l'architecture, nous allons comparer les prédictions de notre réseau de neurones par rapport à la vérité terrain, sur une donnée qu'il n'a jamais vue mais générée de la même manière que la base de données d'entraînement. C'est un point crucial, car notre réseau doit être en mesure de prédire les données d'un ensemble inconnu mais proche de son apprentissage (figure 30).

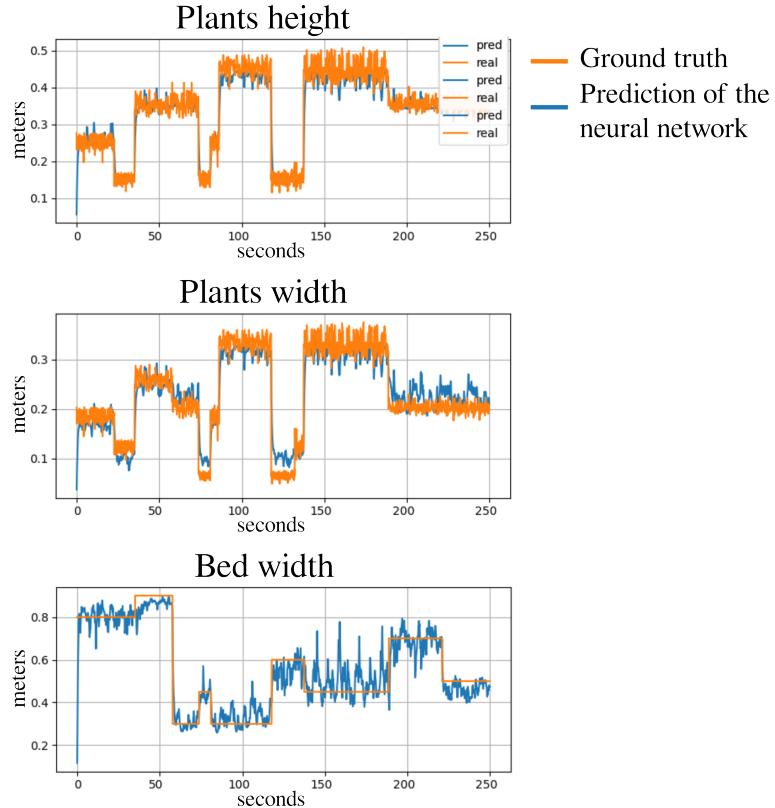


Figure 30: Prédiction d'un réseau de neurones sur une base de données de validation

Nous pouvons observer que le réseau de neurones est capable de traiter un nuage de points pour en prédire la réalité terrain, mais si jamais il est exposé à des données différentes de sa base d'entraînement, il pourrait être sensibles aux changements et avoir une prédiction faussée.

Afin d'obtenir le réseau de neurones le plus robuste au réel possible, nous réalisons l'enregistrement d'une base de données en conditions dégradées, visant à avoir une base de données d'entraînement pseudo-réaliste à l'aide d'un sol irrégulier et de mauvaises herbes (figure 31).



Figure 31: Simulation et création d'une base de données pseudo-réaliste

Suite à cela, nous évaluons notre nouveau réseau de neurones sur plusieurs configurations différentes, nous allons tester de manière indépendante l'impact du sol irrégulier et des mauvaises herbes sur la prédiction (figure 32). Nous cherchons à évaluer et à comprendre la robustesse de notre réseau de neurones selon les situations qui lui sont présentées.

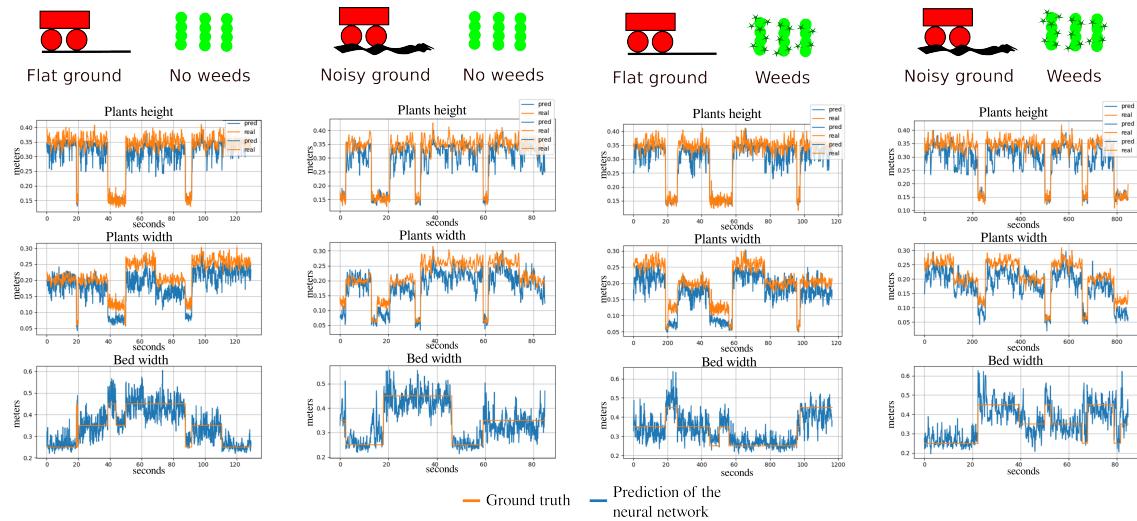


Figure 32: Prédiction du réseau de neurones d'une réalité de terrain, à travers plusieurs expérimentations évaluant l'impact d'un sol imparfait et/ou des mauvaises herbes

Nous pouvons constater que le réseau de neurones à bien réussi à avoir une prédiction proche de la réalité terrain, malgré qu'il soit victime du sol irrégulier et des mauvaises herbes, il réussit tout de même à comprendre son environnement et effectuer une prédiction proche de la réalité.

Grâce à cette prédiction, nous allons pouvoir régler automatiquement les paramètres du furrow, nous espérons avoir une meilleure adaptation générale de l'algorithme. Nous évaluons donc la robustesse de l'adaptation à différentes configurations, on mesure la proportion de rangées de plantes correctement suivies par le tracteur (figure 3).

Table 2: Comparaison entre les différentes architectures

	Validation	Test
Natif	75%	45%
Archi 1	75%	54%

Table 3: Tableau récapitulatif du taux de succès résolution des mondes de test, taux de succès exprimé en proportion de rangées de plantes correctement suivi, architecture actuelle surlignée

Malgré le réglage automatique de paramètres, l'algorithme du furrow reste grandement limité par rapports aux différentes configurations de plantes. Les résultats nous montrent une légère amélioration d'adaptation à diverses configurations de plantes. Cela vient de la méthode de détection, malgré un modèle de plantes attendu correctement paramétré, la détection des rangées par corrélation avec le scan lidar ne permet pas un suivi adapté à tout type de configurations agricoles.

4.4 Conclusion

Dans cette architecture, notre objectif était de déterminer le modèle des plantes attendues à l'aide d'un réseau de neurones, et ceci afin de permettre à l'algorithme du furrow une meilleure autonomie et robustesse.

Lorsqu'il est correctement entraîné, malgré la présence de perturbations (mauvaises herbes, sol irrégulier), le réseau de neurones réussit à prédire avec précision la réalité terrain, ce qui est prometteur pour les développements futurs en intelligence artificielle pour le traitement des nuages de points en milieu agricole.

Cependant, cette architecture est limitée par le fonctionnement intrinsèque du furrow. Dans des configurations ambiguës, bien que le réseau de neurones prédisse les paramètres permettant de bien paramétré la détection du furrow, celle-ci ne réussit quand même pas à s'adapter à tout type de situation et fournir une bonne détection des rangées de plantes. Cela peut mener à une situation où les points d'intérêt détectés peuvent provoquer une divergence dans le suivi de trajectoire par l'algorithme du furrow following (figure 33).

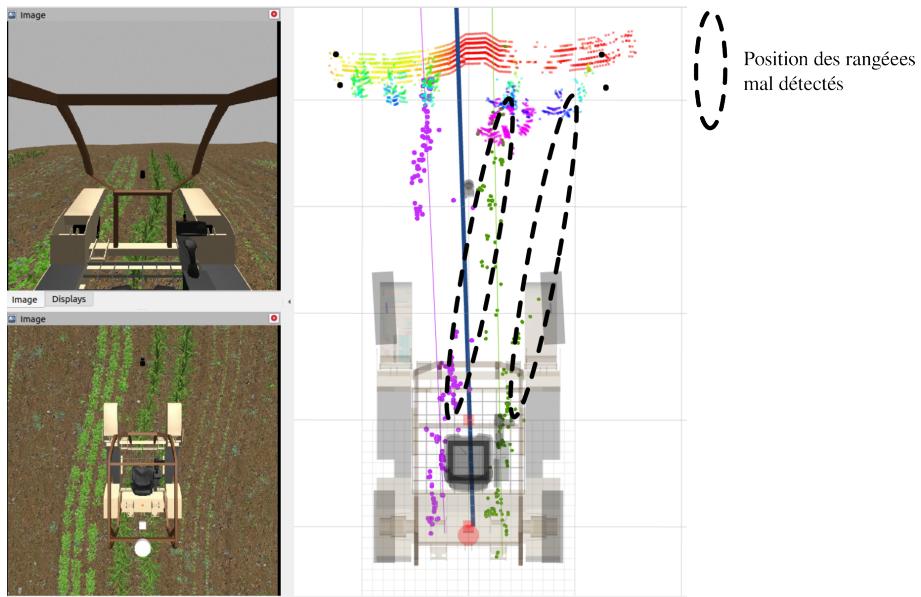


Figure 33: Sorti de route du furrow, malgré les bonnes détections précédentes (points enregistrés en rose et en violet au niveau du châssis), l'algorithme va faire des erreurs de détections qui vont l'amener à changer de rangées (rangées à gauche de la planche originale)

Malgré l'utilisation d'un réseau de neurones, cette architecture reste limitée en termes de détection des points d'intérêt par corrélation. La détection des plantes sera la problématique centrale de l'architecture 2, nous essayerons de trouver directement les points d'intérêts à partir du nuage de points à l'aide d'un réseau de neurones.

5 Architecture 2 : Détection de plantes

Le but de l'architecture 2 est de trouver les point d'intérêts par l'utilisation d'un algorithme d'intelligence artificielle, à la différence de la méthode de corrélation utilisée dans l'algorithme natif. Cette solution nous permettra d'être plus précis dans la détection des plantes, d'obtenir de manière plus robuste le centre des rangées, indépendamment des mauvaise herbes, du sol irrégulier... En effet, la méthode initiale de détection est peu performante, elle présente des difficultés pour différencier les plantes d'un sol irrégulier, au risque de ne rien détecter, ou bien détecter du sol irrégulier en tant que plantes. La détection rencontre des difficultés notamment dans le cadre de :

- Crevasses et bosses
- Plantes asymétriques
- Mauvaises herbes

Pour pallier à ce problème, nous modifierons donc l'architecture originelle par le remplacement de la partie de détection de points d'intérêt par un réseau de neurones, ce qui permet également de ne plus avoir besoin de la détection de paramètres de l'architecture 1 (figure 34).

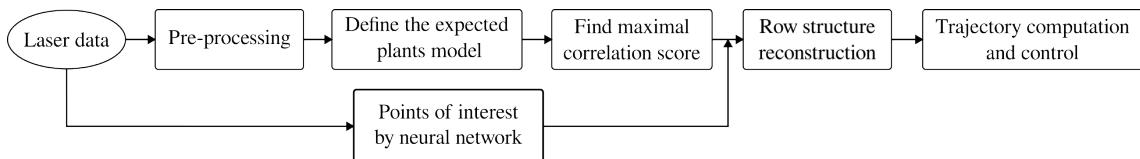


Figure 34: Présentation de l'architecture 2, les points d'intérêts seraient directement détectés par un réseau de neurones

Plutôt que d'utiliser un seul réseau de neurones pour faire l'entièreté de cette opération, nous avons décidé d'accomplir cette tâche en deux étapes (figure 35). La première étape serait un *ground removal*, une segmentation du nuage de points en sol et non-sol, c'est à dire que pour chaque points reçu par le LiDAR, nous déterminons si il s'agit de sol ou bien de plantes. Puis, dans un second temps, à partir du nuage de points segmenté correspondant aux plantes, repérer les amas de points pour déterminer le placement des rangées.

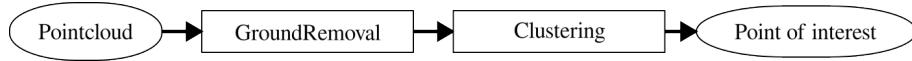


Figure 35: Processus général d'extraction de point d'intérêt de l'architecture 2

5.1 Segmentation du nuages de points

La segmentation d'un nuage de points vise à attribuer à chaque point spécifique une classe (figure 36). Dans le domaine du véhicule autonome, il s'agit souvent de reconnaître la route, les bâtiments, les autres véhicules, les piétons... Dans notre cas, face à un nuage de points 3D issu de l'acquisition dans le domaine agricole, notre but sera de discriminer les points appartenant aux plantes de ceux appartenant au sol. Cette tâche nécessite à la fois une compréhension global et local du nuage de point, c'est un exercice qui est particulièrement approprié pour un réseau de neurones.

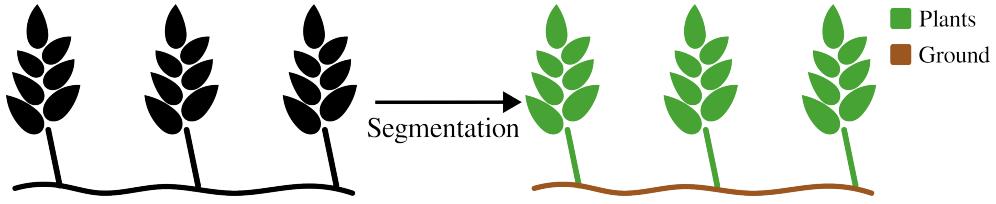


Figure 36: Segmentation du nuages de points en plantes et sol à partir de la géométrie reçu par le LiDAR

Malheureusement pour nous, malgré les bons résultats du réseau de neurones sur sa base de validation, face à une situation nouvelle, il nous donnera un mauvais résultat. Ce défaut compromet l'utilisation de l'intelligence artificielle dans le cas réel, il y a trop de risques que le réseau de neurones prennent des décisions qui mène notre algorithme à ne pas correctement suivre les rangées de plantes (figure 37).

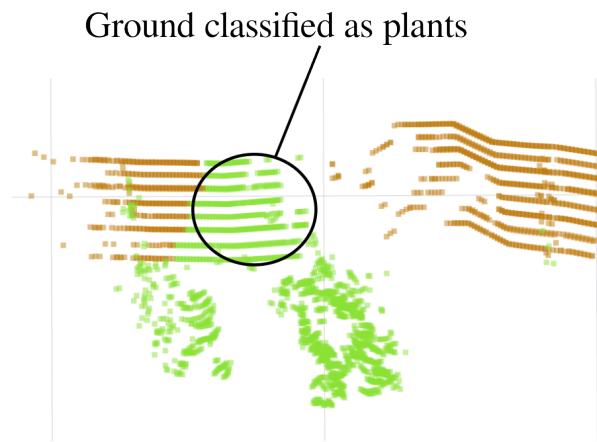


Figure 37: Ambiguïté de détection, le sol devrait être entièrement classifié en marron (les lignes correspondant au LiDAR impactant le sol), contrairement aux plantes qui sont elles correctement classifiées en vert, le réseau de neurones a été trop gourmand en classification des plantes

Nous avons également observé un résultat similaire quand nous avons essayé de segmenter notre nuage de points afin de reconnaître individuellement les rangées de plantes. Le but était qu'en plus de différencier le sol du non-sol, permettre de différencier les points appartenant à la rangée gauche et ceux à la rangée droite. Comme pour la segmentation précédente (figure 37), le réseau de neurones échoue à s'adapter à une nouvelle situation, plutôt que de comprendre réellement le nuage de point, il y reproduit les biais appris lors de son entraînement, effectuer une alternance ABAB des rangées de plantes devant lui, au mépris de leurs dispositions réels (figure 38).

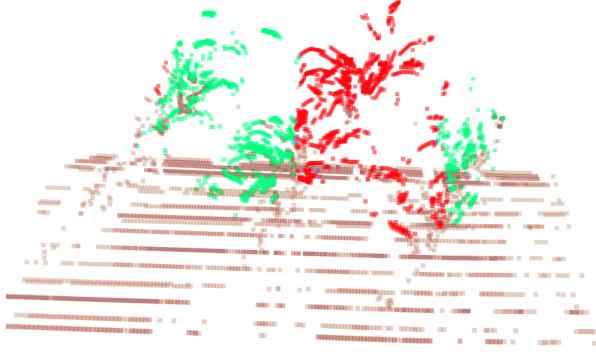


Figure 38: Segmentation de rangées, on cherche à obtenir le sol, et différencier les différentes rangées de plantes

Pour appuyer ces expérimentations, nos recherches bibliographiques nous ont également amené à la conclusion qu'il n'est pas possible de segmenter efficacement dans le cas réel des plantes pour une détection robuste. Les plantes sont difficiles à segmenter car elles se ressemblent, elles se touchent, et pour finir, elles sont dans un environnement non-structuré difficile à reproduire en simulation (voire annexe 8.3).

Afin d'effectuer une segmentation sol/non-sol performante en réel et d'assurer une robustesse dans le suivi de trajectoire, il est préférable d'adopter une approche géométrique plutôt que de s'appuyer sur un réseau de neurones pour la segmentation des points.

Nous nous sommes finalement tournés vers une méthode que nous appellerons la méthode des « toiles de tentes », son but est de détecter les maximums ou minimums locaux dans un nuage de points.

Le nom de la méthode vient de l'idée de placer une tente sur chacun des points, et tous les points non-englobés par la tente des autres points seront considérés comme de maximums locaux (ou minimums selon le sens de la tente). Nous effectuerons cette opération en recherche de minimums locaux sur la hauteur de notre nuage de points, les points englobants seront donc considérés comme le sol, et les points englobés seront considérés comme du non-sol, des plantes dans notre cas (figure 39).

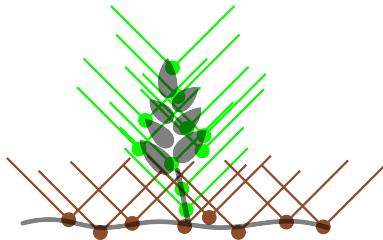


Figure 39: Segmentation sol-plantes par détection de minimums locaux, les points classifiés en minimums locaux englobants sont signalés en marron en tant que sol, les points englobés par d'autres points sont signalés en vert en tant que plantes

Pour chaque points, on regarde si les autres points sont contenus dans la tente formé par celui ci. Notre détection de maximums locaux peut être effectuer sur n'importe quel nombre de dimensions égale ou supérieur à 2. Pour N points, on aura absolument besoin de :

- \underbrace{H}_{N} : La dimension des points où l'on recherche les maximums locaux (dans notre cas, Z, la hauteur)
- $\underbrace{D}_{N,N}$: La distance points à points dans l'espace multi-dimensionnelle (dans notre cas, la distance entre les points sur le plan XY)

Le point P_i est considéré comme un maximum local si tous les autres points sont à l'intérieur du cône formé par celui-ci, défini par la fonction affine $a * x + b$ avec x pour distance entre les 2 points, b la hauteur du sommet étudié, et a pour la raideur du cône :

$$isMax(P_i) = \bigwedge_{j=0}^n H_j \leq a * D_{i,j} + H_i \quad (6)$$

Ce qui nous donne géométriquement une comparaison entre le point P_j et sa projection sur le cône formé par le point étudié P_i (figure 40).

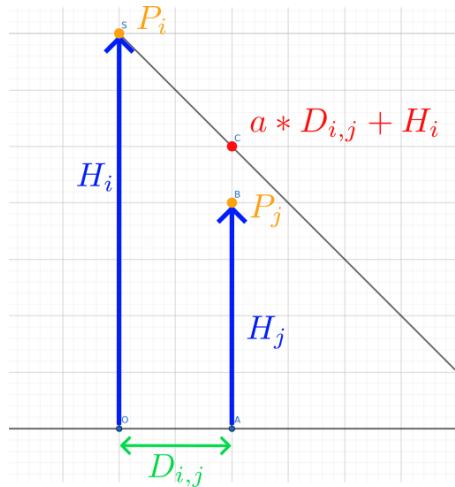


Figure 40: Mise en lumière des variables de la technique des toiles de tente, dans ce cas ci, P_i est un maximum local et P_j un point englobé par P_i

Cette méthode de segmentation s'avérera performante en conditions réelles (figure 41). Nous avons réussi à trouver une méthode basée sur la géométrie pour remplacer le réseau de neurones dans la segmentation du nuage de points 3D, ce qui nous permet d'avoir une segmentation indépendante de l'apprentissage en simulation et robuste en conditions réelles.

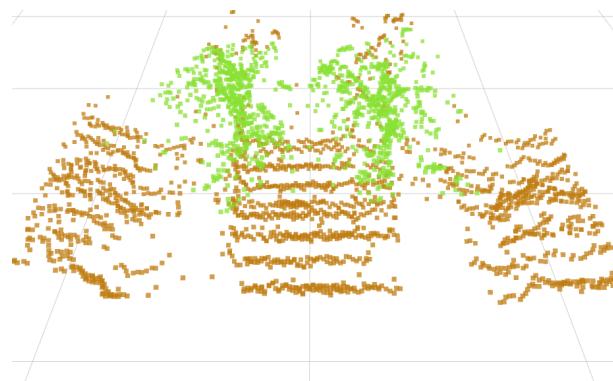


Figure 41: maximums locaux de densité sur une acquisition LiDAR de plantes

Le choix de ne pas utiliser une segmentation basée réseau de neurones s'est également confirmé par une plus large recherche bibliographique détaillée en Annexe A 8.3.

L'explication plus complète ainsi que la formulation mathématique de la détection d'extrema locaux se trouve en Annexe B 8.3.

5.2 Détection de points d'intérêt

Une fois la segmentation effectuée, nous garderons uniquement le nuage de points correspondant aux plantes, notre but final est de fournir les points d'intérêts décrivant le centre de chacune des plantes. Par l'aspect déterministe de cette opération, nous avons décidé d'opter pour des opérations géométrique plutôt que l'utilisation d'un réseau de neurones. En effet, il nous apparaît plus efficace et plus robuste, face à nuage de points composés d'amas, d'utiliser un algorithme classique pour détecter le centre de ces amas de points. Par nos différentes expérimentations, ainsi que la recherche scientifique dans le domaine, pour décrire le centre d'une plante, nous nous tournerons vers la détection du centre des plantes par maximums locaux de densité (figure 42).

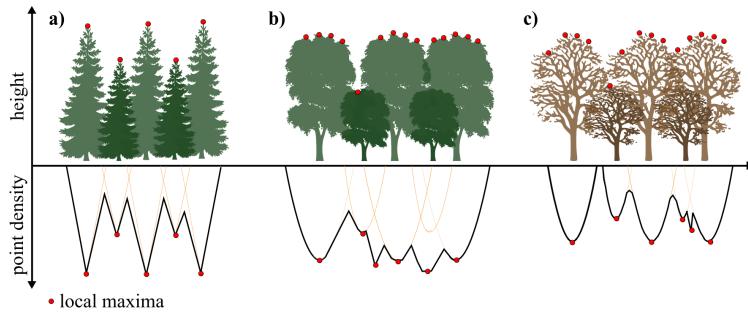


Figure 42: "Comparison between the performance of the local maximums (upper row) and the density-based (lower row) approaches when applied to (a) coniferous stands, and to deciduous tree stands in (b) leaf-on and (c) leaf-off conditions." [11]

Cette méthode nous permettra une grande robustesse de détection, la densité nous permet d'éliminer les fausses détections, ainsi que les feuilles et les branches pour finalement garder uniquement le centre de la plante, centre qui présente la plus grande densité par rapport au reste de la plante.

La densité est également un critère pertinent pour obtenir un résultat robuste s'adaptant à tout type de plantes, qu'il importe leurs tailles ou leurs géométries (figure 43).

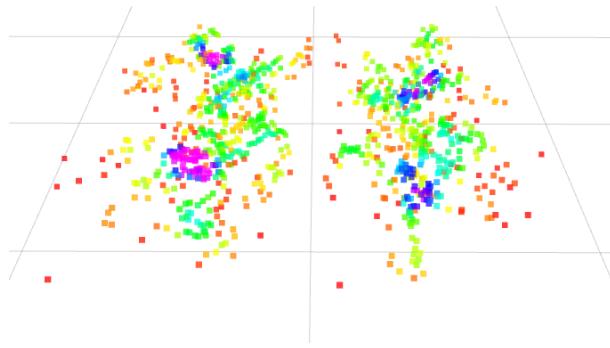


Figure 43: Carte de densité d'un nuage de points 3D résultant d'un scan agricole, après suppression du sol par analyse géométrique

Une fois la densité obtenue par la nombre de plus proche voisin de chaque points, nous ré-utiliserons la technique des maximums locaux par toiles de tente afin d'obtenir les maximums locaux de densité en tant que point d'intérêts de nos plantes (figure 44).

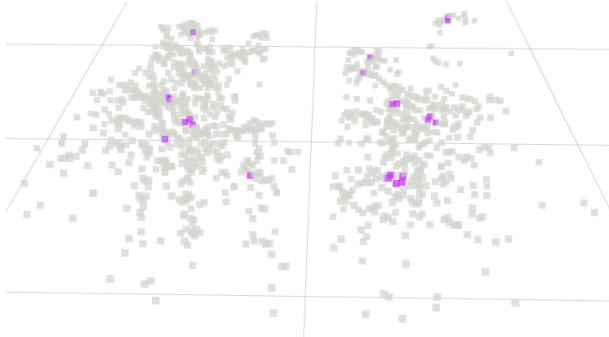


Figure 44: Maximums locaux de densité sur une acquisition LiDAR de plantes

Cette méthode permet à la fois de trouver le point le plus significatif dans une zone de forte densité, mais contrairement au maximum de densité globale, nous pouvons aussi prendre en compte les zones avec une plus faible densités. Une rangée densément peuplée ne prendra pas toute l'attention de notre algorithme, nous serons toujours capable de détecter les rangées les moins denses malgré cela.

Finalement, notre architecture 2 se déroulera en trois étapes, on aura la suppression du sol afin de garder uniquement les plantes, on calcul la densité de notre nuage de points de plantes, et finalement, on détecte les maximums locaux de densité en tant que points d'intérêts des plantes (figure 45).



Figure 45: Processus général d'extraction de point d'intérêt de l'architecture 2

5.3 Expérimentations

Pour évaluer l'amélioration apportée par l'architecture 2, nous confrontons nos algorithmes à des enregistrements de données réelles. Dans un premier temps, nous allons comparer le résultat de l'architecture 1 à l'architecture 2 en terme d'écart latéral et angulaire à la trajectoire détectée. Les écarts détectés viennent de l'expérimentation imparfaite réalisée, par la plantation des plantes et le contrôle du tracteur, la trajectoire formée par les plantes dans le repère du robot varie de plus ou moins 5 centimètres en moyenne. Nous espérons obtenir le même résultat dans le cas d'enregistrement de données réel quand l'algorithme du furrow s'est bien déroulé, l'architecture 2 est censée trouver une trajectoire similaire que l'architecture 1 (figure 46).

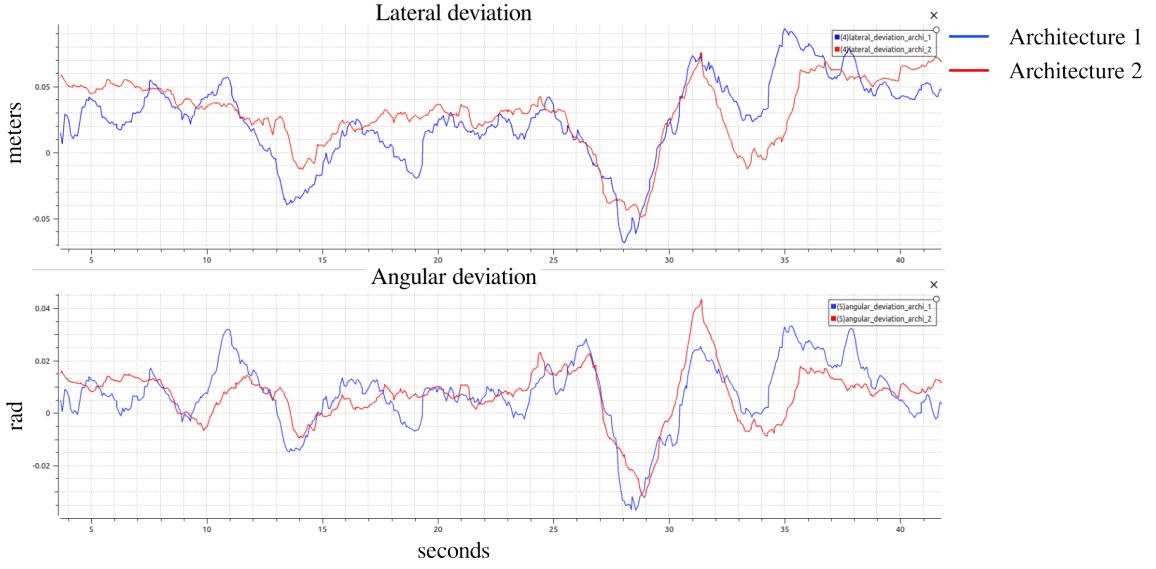


Figure 46: Comparaison de l'écart angulaire et latérale calculé entre l'architecture 1 et 2, dans un fonctionnement nominal

Comme indiqué précédemment, nous ne pouvons pas identifier une nette amélioration dans le fonctionnement classique de l'algorithme, les points détectés sont similaires, la prédition de trajectoire à suivre est similaire elle aussi. Il nous faut alors mettre à l'épreuve cette architecture dans un fonctionnement dégradé, là où la détection des plantes est une tâche difficile pour l'architecture 1, qu'il ait des difficultés à trouver la trajectoire permettant le suivi des plantes (figure 47).

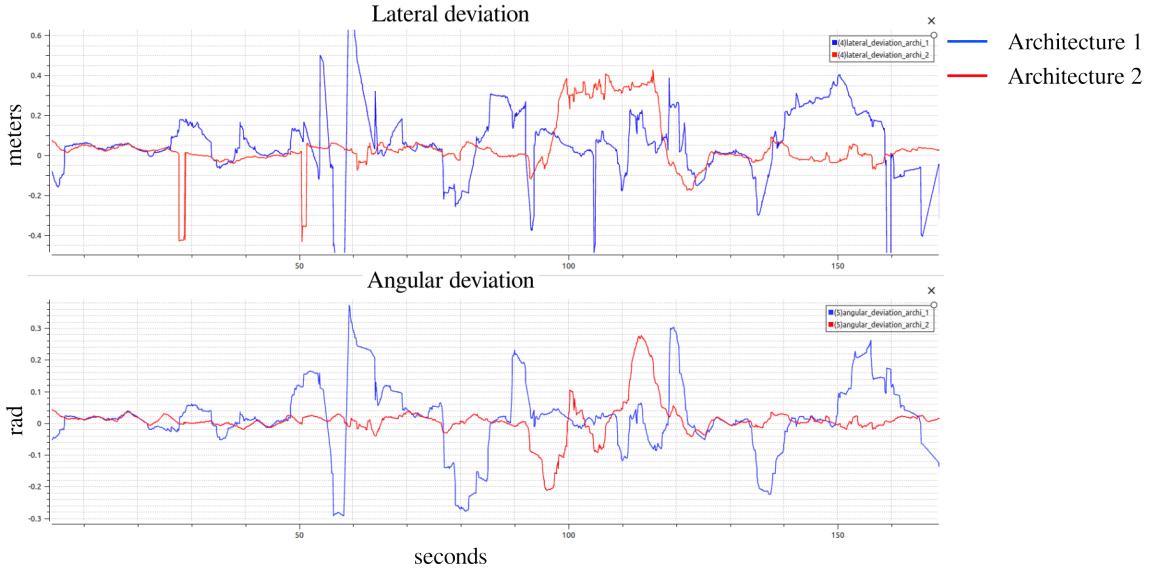


Figure 47: Comparaison de l'écart angulaire et latérale par rapport à une trajectoire fourni par l'architecture 1 et 2, dans un fonctionnement dégradé

Dans le cas d'une expérimentation où les plantes ont été plantées de manière éparsé, avec des petites plantes et la présence de mauvaises herbes, on se doit d'avoir une détection performante dans ce type d'environnement. Nous pouvons observer que l'architecture 1 à beaucoup de difficultés et va de souvent diverger sur les rangées voisines (les discontinuités et plateau au niveau de l'écart latéral), alors que

l'architecture 2 quant à elle, réussi à détecter les bonnes rangées de plantes.

Nous évaluons la robustesse de cette nouvelle méthode de détection, nous mesurons la proportion de rangées de plantes correctement suivies par le tracteur (figure 4).

Table 4: Comparaison entre les différentes architectures

	Validation	Test
Natif	75%	45%
Archi 1	75%	54%
Archi 2	100%	82%

Table 5: Tableau récapitulatif du taux de succès résolution des mondes de test, taux de succès exprimé en proportion de rangées de plantes correctement suivi, architecture actuelle surlignée

Par le changement ciblé de l'algorithme de perception, ces améliorations peuvent être attribuées au changement de paradigme de détection, nous ne nous baserons plus sur la corrélation du LiDAR à un modèle attendu, mais sur la géométrie du nuage de points. Cela nous permet d'éviter les fausses détections, et d'améliorer grandement la robustesse de notre algorithme. C'est notamment le cas quand il y a des rangées avec des petites plantes et des traces de roues, la perception basée par corrélation avec une fonction carrée ne permet pas de discriminer correctement les plantes du sol, c'est le cas du monde de test, où on passe de 54 à 82% de résolution pour la perception basée géométrie.

5.4 Conclusion

Dans cette partie, nous avons exploré l'intelligence artificielle dans la compréhension locale dans notre nuage de points, et nous nous sommes malheureusement heurtés au plus grand problème de l'utilisation à des réseaux de neurones, sa performance en situations réelles est lourdement dégradé par rapport à sa performance en simulation. En effet, dans le processus d'apprentissage, il est absolument nécessaire d'avoir une base de données d'apprentissage proche de la tâche que l'on souhaite réaliser. Dans le cadre d'une segmentation sol-plantes, l'acquisition et l'annotation d'une base de données assez conséquentes en réel aurait été une tâche coûteuse, chronophage et peu pertinent quand d'autres méthodes de segmentation sont possibles.

L'approche géométrique nous permet dorénavant d'appréhender des situations agricoles complexes comme des rangées de plantes asymétriques en forme et en type, des petites plantes aussi grandes que les bosses et sillons de terres perturbant la perception par nuage de points. Nous avons permis une plus grande robustesse, autonomie et précision de notre algorithme, nous percevons mieux, dans plus de situations différentes, et sans nécessité de réglage manuel.

Malgré l'amélioration significative du furrow, dans un fonctionnement dégradé, il est aussi possible que l'architecture 2 se retrouve en situation ambiguë et en vienne à changer de rangées (figure 48). Malgré une meilleure détection des plantes qui réduit largement les erreurs, on peut toujours être victime des aléas de l'environnement agricole.

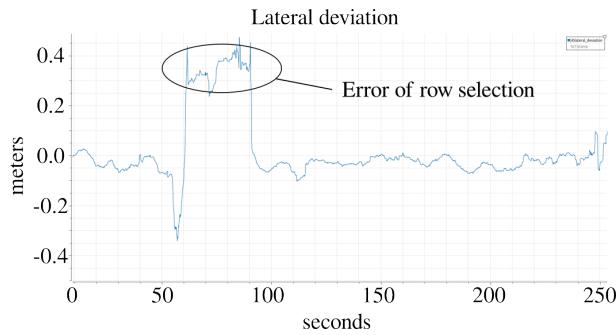


Figure 48: Expression de l'écart angulaire et latérale par rapport à une trajectoire dans un fonctionnement dégradé

La réponse à cette problématique serait d'avoir une meilleure compréhension des points d'intérêt correspondant aux plantes, permettant une meilleure robustesse et adaptabilité aux situations ambiguës. En effet, ces problèmes techniques de l'architecture 2 proviennent du fonctionnement intrinsèque du furrow, la sélection de points d'intérêt par approche itérative ne permet pas la prise de décision adéquate à tout type de situations. Nous allons explorer l'intelligence artificielle comme une solution pertinente à cette problématique. En effet, par le calcul d'écart angulaire et latéral à la trajectoire par un réseau de neurones, nous espérons obtenir un résultat plus robuste malgré les situations ambiguës rencontrées.

6 Architecture 3 : Suivi de trajectoire par réseau de neurones

Le but de l'architecture 3 sera d'avoir un fonctionnement du réseau de neurones dit en "End-to-End", de bout en bout (figure 49). Cela veut dire que notre réseau de neurones prédira, directement à partir du nuage de points, le comportement à adopter par notre robot. Cette méthode permet d'englober tout le processus dans un simple réseau de neurones, la difficulté et l'avantage de cette méthode réside dans la qualité de l'entraînement du réseau de neurones. Nous nous devons donc de proposer un entraînement varié et approfondi, aussi bien pour le comportement du tracteur que pour la configuration des plantes.

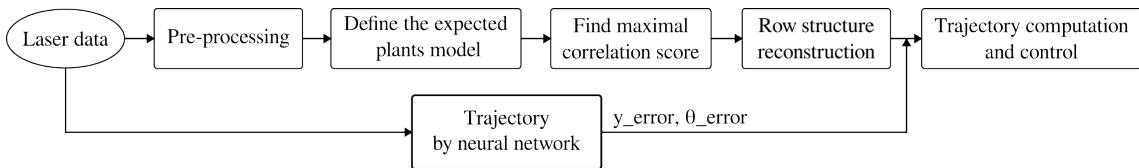


Figure 49: Présentation de l'architecture 3, le réseau de neurones prédit à partir de la donnée LiDAR une trajectoire à suivre sous la forme d'un écart latéral et angulaire (y_error et θ_error)

Pour que le réseau de neurones prédise le comportement du robot, nous lui demanderons de nous donner l'écart latérale et angulaire à la trajectoire, trajectoire définie par rapport à l'alignement des rangées de plantes (figure 50).

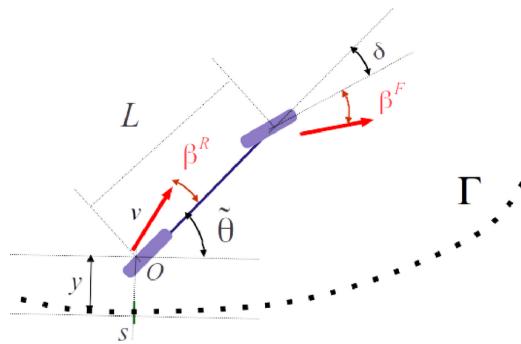


Figure 50: Expression de l'écart angulaire et latérale par rapport à une trajectoire, modèle bicyclette

Durant l'entraînement, nous allons créer une base d'entraînement avec diverses configurations de plantes/comportement du robot, et comme pour l'architecture 1, nous demanderons au même réseau de neurones de nous prédire par régression les variables d'écart angulaire et latéral. Nous pouvons obtenir la réalité terrain grâce à la position du robot et des plantes accessibles via la simulation, à partir de cela, nous pouvons calculer et entraîner notre réseau de réseau de neurones avec l'écart à la trajectoire du robot dans une situation donnée.

Avec cette architecture, nous cherchons à explorer l'idée novatrice d'avoir une intelligence artificielle générale qui nous permettrait d'obtenir une prise de décision en tout temps robuste et adapté à la situation. La rupture avec nos algorithmes actuels viendrait du fait que l'algorithme basé apprentissage donne de meilleurs résultats que la programmation d'un algorithme classique. C'est notamment le cas dans le traitement d'image ou le traitement du langage naturel (NLP), où l'intelligence artificielle a bouleversé les performances des algorithmes déterministes. Pour la perception et le contrôle d'un véhicule dans le milieu agricole, l'utilisation de l'intelligence artificielle est encore peu répandue, nous préfèrons des algorithmes plus classique, performant et déterministe.

Il est difficile dans le domaine du contrôle d'obtenir de meilleurs résultats en laissant la prise de décision

finale au réseau de neurones, quelques résultats ont vu récemment le jour, mais se concentrant plutôt sur le réglage des paramètres de l'algorithme plutôt que le fonctionnement en "End-to-End" [7].

6.1 Expérimentations

Pour tester la prédiction d'écart à la trajectoire de notre réseau de neurones, nous allons dans un premier temps, évaluer la prédiction de trajectoire par le suivies de rangées de plantes en simulation, dans les mondes de validation et de test de nos algorithmes :

Table 6: Comparaison entre les différentes architectures

	Validation	Test
Natif	75%	45%
Archi 1	75%	54%
Archi 2	100%	82%
Archi 3	83%	45%

Table 7: Tableau récapitulatif du taux de succès résolution des mondes de test, taux de succès exprimé en proportion de rangées de plantes correctement suivi, architecture actuelle surlignée

Dans l'évaluation de cette architecture, nous avons été agréablement surpris par la résolution du monde de validation à 83%, la prédiction de trajectoire par le réseau de neurones permet d'avoir un suivi globalement correct. Cependant, dès que l'on va mettre notre réseau de neurones face à une nouvelle situation, malgré qu'elle soit similaire à la base d'entraînement, le résultat chute dramatiquement à 45% pour le monde de test 2. Cela pourrait être dû à la difficulté pour notre réseau de neurones à s'adapter à de nouvelles situations.

Pour aller plus loin et confirmer l'idée que notre réseau de neurones rencontre des difficultés face à une donnée plus complexe que sa base d'entraînement, nous avons effectué la comparaison entre les différentes architectures sur l'expression de l'écart angulaire et latérale sur de la donnée réelle, afin de comparer la prédiction du réseau de neurones par rapport à ce qu'aurait prédit nos algorithmes déterministe (figure 51).

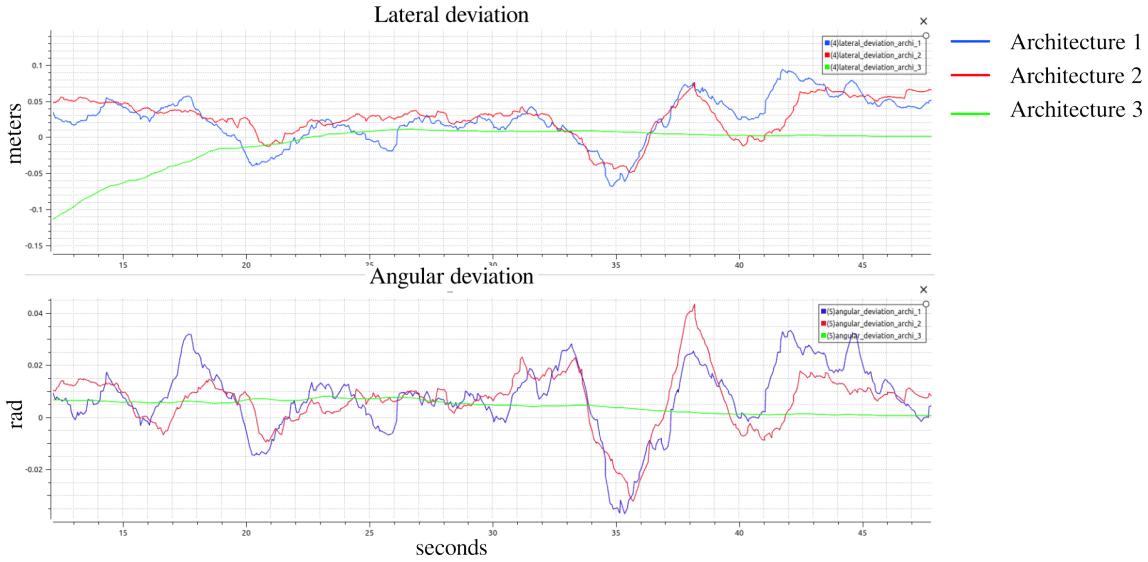


Figure 51: Comparaison de l'écart angulaire et latéral dans le cas général, jeu de données réels, cas nominal

Nous observons que le réseau de neurones est complètement perdu face à une données réel, il ne réussit pas du tout à comprendre son environnement et à nous donner une prédition de trajectoire proche de la réalité.

6.2 Conclusion

En simulation, la prédition du réseau de neurones semble relativement proche du réel et robuste aux difficultés du suivi de rangées, par son succès en monde de validation, il a l'air d'être le candidat idéal pour assurer un suivi de trajectoire dans le contexte agricole. Malheureusement, une problématique apparaît à l'expérimentation sur de la donnée réelle : le réseau de neurones est incapable de prédire la trajectoire à suivre. Ce phénomène est dû à notre base de données simulé trop éloigné des situations réels, malgré sa diversité, la base de données d'entraînement ne permet pas de donner la compréhension nécessaire à notre réseau de neurones pour lui permettre de s'adapter au réel.

Nous aurions pu tester d'autres architectures neuronales dans cette tâche, mais par la nature de notre problématique, nous nous sommes contenté de notre PointNet simplifié [2]. D'après notre intuition, notre réseau de neurones est parfaitement adapté à la mission qui lui est assignée. Il parvient à saisir efficacement son ensemble d'apprentissage, ce qui se manifeste par sa capacité à suivre avec succès les plantes le monde de validation, sa capacité de compréhension des données est suffisante. Les difficultés apparaissent uniquement lorsque le réseau de neurones est confronté à des scénarios qui diffèrent de ceux présents dans sa base d'apprentissage. Il est tout de fois possible que le réseau de neurones lui-même, ou bien son entraînement soit en partie responsable du problème de généralisation.

Conformément à nos attentes, cette architecture ne sera finalement pas une solution pertinente à la perception et au contrôle de robot en milieu agricole. En effet, ce n'est pas un paradigme qui, à l'heure actuelle, serait propice aux améliorations apportées par un réseau de neurones de bout en bout. Le fonctionnement de type boite noire du réseau de neurones est grandement problématique dans son application pour le contrôle robotique, nous ne savons jamais comment va réagir un réseau de neurones face à une situation inédite, et c'est encore pire quand la situation diffère de sa base d'entraînement.

L'utilisation d'une boite noire dans le cadre de la robotique agricole est très problématique, elle ne nous permet pas d'assurer la sécurité des biens et des personnes à chaque instant. Ces différentes raisons rendent difficile la certification et l'utilisation d'intelligence artificielle en contrôle robotique.

Néanmoins, il nous faut trouver une solution permettant d'apporter flexibilité et adaptabilité à nos algorithmes, sans passer par l'utilisation de réseau de neurones. Nous devrons faire en sorte que de

notre algorithme puisse analyser et s'adapter à tout type de situations, malgré les potentielles mauvaises détections et situations ambigu.

7 Architecture 4 : Suivi de trajectoire par modèle probabiliste

Suite à nos résultats basés réseau de neurones, nous avons vu que l'intelligence artificielle ne permet pas nécessairement de répondre à notre problématiques d'autonomie et de robustesse pour le suivies de plantes. Afin d'améliorer les performances du furrow, nous proposons d'approcher ces problématiques par un nouveau paradigme.

Revenons sur la création de trajectoire à partir de la détection des plantes, le furrow n'est pas seulement un algorithme de perception purement déterministe, il joue aussi un rôle de prise de décision dans le calcul de trajectoire (figure 52).

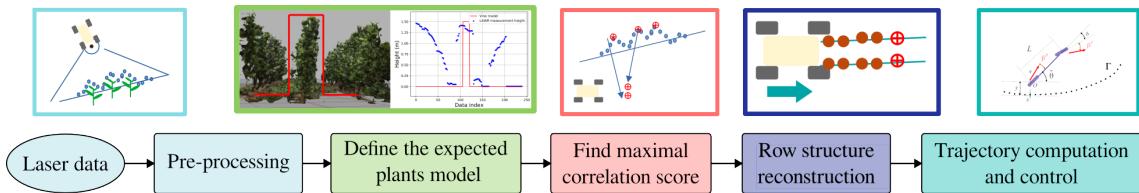


Figure 52: Schéma fonctionnement furrow

Pour permettre la création d'une trajectoire à suivre à partir des points d'intérêt correspondant aux plantes détectées, le furrow procède en 2 étapes :

- La sélection des points d'intérêt : on sélectionne les points appartenant à la rangée de gauche et de droite, en sélectionnant les points courant les plus proches des points sélectionnés précédemment ("Select closest maximum")
- La régression des points d'intérêt classifiés : avec une régression du premier ordre, on définit, pour les deux rangées, les deux droites qui les représentent le mieux, puis par les moyennes des écarts obtenus, on obtient l'écart angulaire et latéral final à la trajectoire ("Trajectory computation and control")

Cet algorithme fonctionne largement dans le cas nominal, si on a assez de détection de bonnes qualités, avec un robot stable n'effectuant pas de grandes manœuvres, l'algorithme réussit à garder une sélection correcte des points d'intérêt, ses mesures sont proches de la réalité.

Par contre, si jamais l'algorithme se retrouve dans une situation dégradé, il peut être amené à prendre de mauvaises décisions (figure 53). C'est-à-dire que dans l'analyse du dernier point d'intérêt reçus, la sélection mènera l'algorithme à changer de rangées de plantes suivies.

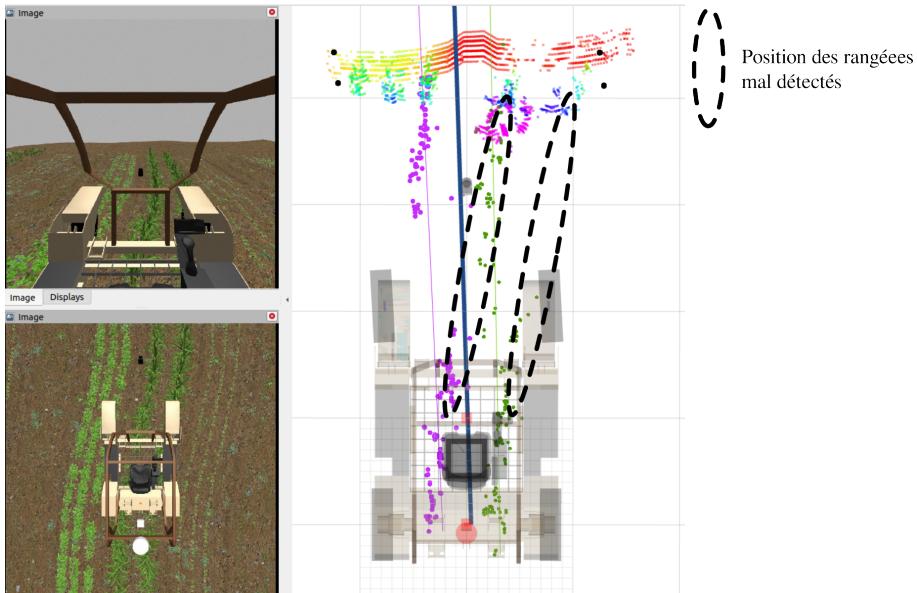


Figure 53: Mauvaise sélection de points d'intérêt à cause des détections précédentes, malgré la bonne détection des plantes au niveau de l'essieu arrière, l'incertitude au niveau des roues avant et les choix de points d'intérêt précédents pousse le véhicule à aller suivre la rangée voisine de celle originale

La raison qui pousse l'algorithme à adopter ce comportement est sa manière de créer sa trajectoire à partir des détections de plantes, il va sélectionner les points d'intérêt de manière déterministe (les plus proches voisins). Lorsque l'algorithme va effectuer une mauvaise classification des plantes détectées, celle-ci va à son tour impacté les sélections suivantes. Ce fonctionnement est peu robuste aux erreurs, une accumulation de fausses détections fait facilement diverger l'algorithme de suivi, ce qui le pousse à sortir de sa trajectoire.

C'est pour cela que dans cette partie, nous allons explorer l'approche probabiliste de la sélection de points d'intérêt et de la création de trajectoire. On va définir un modèle d'état correspondant à la répartition attendue des points d'intérêts, puis, par la correspondance du modèle aux données, nous allons calculer la trajectoire à suivre.

Cette méthode devra avoir deux grands objectifs :

- Comprendre l'ensemble des points d'intérêt sous la forme de deux rangées distinctes
- Reprendre une détection correcte en cas d'absence de points d'intérêt ou de mauvaises décisions (minimums locaux dans la sélection des points)

Afin de répondre à ces différents objectifs, nous allons utiliser le fonctionnement d'une mesure impactant un état afin de pouvoir prendre en compte l'incertitude de cette mesure (figure 54). Le système traitera en entrée le nuage 2D de points d'intérêts V avec N points, pour en sortir l'écart latéral et angulaire à la trajectoire : y, θ . Pour modéliser la répartition des points d'intérêts, nous utiliserons le vecteur d'état X en S dimensions, qui sera déterminé à partir de la mesure M de même représentation et dimension que l'état lui-même. Avant de prendre en compte la mise à jour apportée par la nouvelle mesure M , nous ferons évoluer notre état X afin de prendre en compte le déplacement du robot u , ainsi que l'incertitude d'évolution d'état à état. Pour finir, la mesure des points d'intérêts devra être initialisée à partir de la fusion entre l'état courant du système et un état par défaut, afin de permettre une bonne adaptabilité de la mesure tout en gardant une plage de fonctionnement nominale, ce qui permet adaptabilité et robustesse de la création de trajectoire. Notre mesure des points d'intérêts se fera par une mixture de gaussiennes contraintes, nous détaillerons ce choix par la suite.

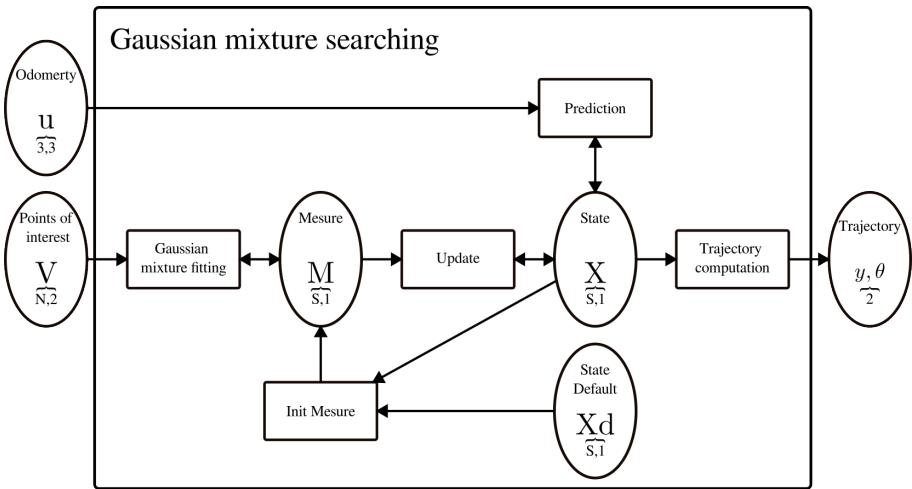


Figure 54: Fonctionnement générale de l'architecture 4

La détection de points d'intérêt sera celle de l'architecture 2 (maximums locaux de densité des plantes), c'est elle qui nous a montré jusque ici la plus grande fiabilité. Nous aurons donc pour architecture globale l'architecture 2, avec la partie de sélection de points d'intérêt remplacée par la recherche d'une mixture de gaussiennes (figure 55).

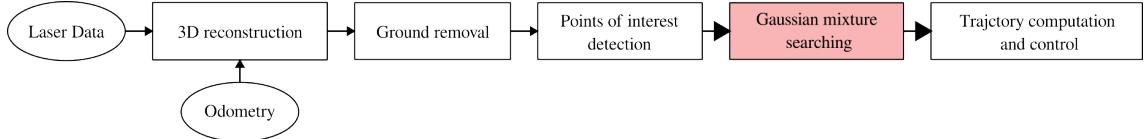


Figure 55: Fonctionnement général du furrow avec la perception de l'architecture 2 (maximums locaux de densité), et la construction de trajectoire de l'architecture 4 (recherche de mixture de gaussiennes), apport de l'architecture 4 mise en valeur

7.1 Recherche des points d'intérêt par modèle gaussien

Pour permettre notre modèle d'état de correspondre aux données, de manière fiable et contrôlé, nous avons utilisé la fonction gaussienne en 2 dimensions pour représenter la répartition des points d'intérêt correspondant aux plantes (figure 56). Nous faisons l'hypothèse que les points d'intérêt d'une rangée de plantes peuvent être répartis selon une gaussienne allongée. L'utilisation de gaussiennes a pour objectif de nous permettre une mesure proche des rangées de plantes, avec la possibilité de restreindre l'impact des outliers, les détections éloignées de la rangée de plantes.

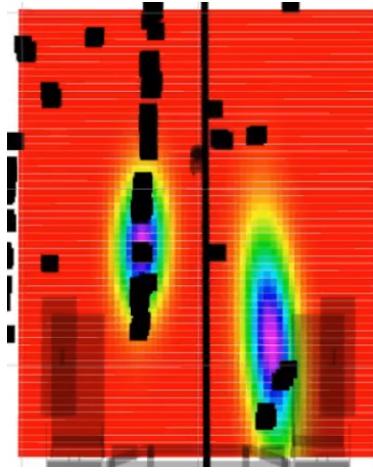


Figure 56: Répartitions des points d'intérêt par modèle gaussien, face une situation réelle comportant des absences de plantes et des mauvaises herbes

7.1.1 Définition du modèle gaussien

Dans un premier temps, nous définissons la manière qu'aura le modèle gaussiens de représenter au mieux les points d'intérêts. Nous utiliserons des fonctions gaussiennes avec des degrés de liberté, ce seront eux qui composeront notre état, et qui permettront à nos gaussiennes de venir se superposer aux points d'intérêt. Nous avons exploré plusieurs niveaux de degrés de liberté pour notre algorithme afin d'obtenir le résultat le plus satisfaisant pour détecter nos rangées de plantes (figure 57).

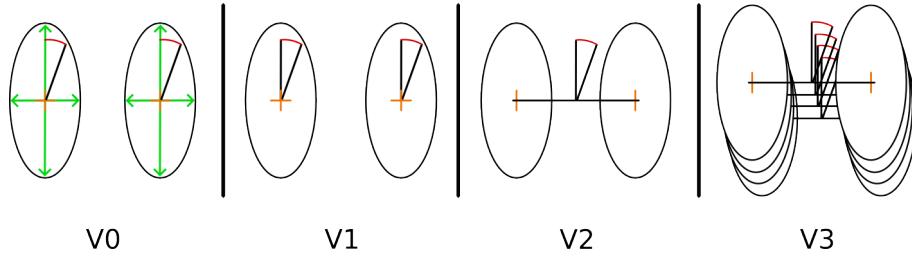


Figure 57: Schéma des différents scénarios de l'architecture 4, les degrés de liberté sont représentés par les grandeurs colorées

Les degrés de liberté seront la forme de la gaussienne (sa matrice de covariance), ainsi que son placement : soit totalement libre, soit contraint sur un des axes. Les différentes versions se présentent comme ceci :

- V0 : Deux Gaussiennes totalement libres de leurs formes et de leurs placements ($\underbrace{\mu_1}_2, \underbrace{\mu_2}_2, \underbrace{\Sigma_1}_{2,2}, \underbrace{\Sigma_2}_{2,2}$)
- V1 : Deux Gaussiennes contraintes de leurs formes et totalement libres de leurs placements ($\underbrace{\mu_1}_2, \underbrace{\mu_2}_2, \theta_1, \theta_2$)
- V2 : Deux Gaussiennes contraintes de leurs formes et contraintes de leurs placements dans le sens d'avancement du robot (y_1, y_2, θ_1)
- V3 : Mixture de pairs de Gaussiennes contraintes de leurs formes et contraintes de leurs placements dans le sens d'avancement du robot ($((y_{1,1}, y_{2,1}, \theta_{1,1}), \dots, (y_{1,n}, y_{2,n}, \theta_{1,n}))$)

Nous décrivons les degrés de liberté et caractéristiques des différentes versions sous la forme d'un tableau :

Table 8: Comparaison entre les différentes architectures

	V0	V1	V2	V3
Covariance Σ	oui	non	non	non
Rotation indépendante	oui	oui	non	non
Position en x	oui	oui	non	non
Position en y	oui	oui	oui	oui
Mixture de gaussiennes	non	non	non	oui
Degrés de liberté	2*5	2*3	3	$N_{gauss} * 3$

Table 9: Tableau récapitulatif des degrés de liberté des différentes versions de modèle d'état

Le défi à relever sur le choix du niveau de degrés de liberté, est d'obtenir le modèle le plus libre possible pour lui permettre de s'adapter à la donnée, mais avec la contrainte de son but initial : décrire les rangées de plantes afin de fournir une erreur angulaire et latérale à la trajectoire formée par celle ci, en garantissant robustesse et précision de suivie.

Nous nous sommes finalement dirigé vers une mixture de pairs de gaussiennes contraintes. La version V3 offre de nombreuses qualités de détection. Notre choix s'est tourné vers elle pour différentes raisons tels que :

- La stabilité de l'état par la contrainte sur le placement sur l'axe X (placement à un certain niveau devant l'essieu arrière du robot, liberté sur l'écart latéral et angulaire)
- La robustesse de détection offerte par la même rotation entre les 2 gaussiennes (on présume que les rangées de plantes sont toujours parallèles)
- Le lissage et la précision de la commande par la prise en compte d'une multitude de gaussiennes

Cette modélisation de nos points d'intérêts nous permettra d'obtenir une représentation adaptée aux courbures (figure 58).

Gaussian mixture
Curve fitting Long gaussian
Curve fitting

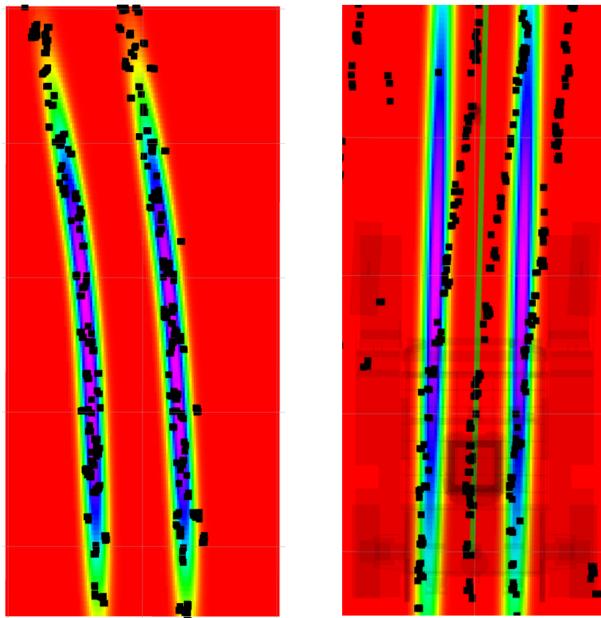


Figure 58: Comparaison entre une seule grande gaussienne contre une mixture de gaussiennes par rapport à une courbure

La première idée était d'utiliser une gaussienne totalement libre en position et en covariance. Elle peut se placer sur les données qui lui sont les plus représentatives, avec la répartition qui correspond le mieux. On définit dans un premier temps notre matrice correspondant aux coordonnées du nuage de points 2D des points d'intérêt. On aura n points sur les dimensions x et y :

$$V = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \dots & \dots \\ x_n & y_n \end{bmatrix} \quad (7)$$

Nous cherchons à mesurer notre état à partir des points d'intérêts correspondant à la détection des plantes. Notre mesure étant de même nature que notre état, dans le fonctionnement parfait, sans bruit, sans variations entre la mesure et l'état, nous pouvons approximer l'état par la mesure des points d'intérêts :

$$\underbrace{X}_{S,1} \simeq \underbrace{M}_{S,1} = \text{Mesure}(\underbrace{V}_{N,2}) \quad (8)$$

Pour définir et permettre la correspondance entre la mesure et les points d'intérêts, nous utiliserons la méthode de la descente de gradient. Pour cela, nous définirons un critère de correspondance entre le modèle et la donnée, puis, par la minimisation de ce critère par la modification des degrés de liberté du modèle, nous obtiendrons un modèle représentatif de la répartition des points d'intérêts. Les degrés de liberté déterminés seront considérés comme la mesure de notre état (respectivement M et X).

Le choix du critère à maximiser² se tourne vers la somme des projections des points d'intérêt sur la gaussienne, afin qu'elle se place sur l'amas de point le plus proche d'elle. En représentation sur une gaussienne 1D :

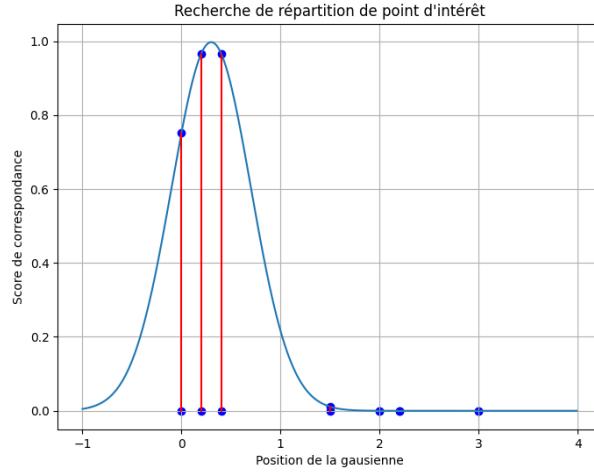


Figure 59: Schéma représentant la recherche de points d'intérêt par placement de gaussienne, le but étant de maximiser la somme des projections de chacun des points (distance en rouge)

Nous définissons notre Gaussienne 2D par :

$$N(\underbrace{x}_2, \underbrace{\mu}_2, \underbrace{\Sigma}_{2,2}) = Ae^B \quad (9)$$

Avec A et B :

$$A = \frac{1}{2\pi\sqrt{|\Sigma|}}, \quad B = -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \quad (10)$$

Ce qui nous donne :

$$N(x, \mu, \Sigma) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)} \quad (11)$$

Nous définissons également une gaussienne à amplitude limitée par :

$$N_{limited}(x, \mu, \Sigma) = e^B \quad (12)$$

Nous définissons la métrique $\psi(X, \mu, \Sigma)$ qui, en fonction de la donnée et du modèle gaussien, nous renvoie un score de correspondance :

$$\psi(V, \mu, \Sigma) = \sum_{i=0}^n N(V_i, \mu, \Sigma) \quad (13)$$

Nous cherchons à obtenir une correspondance maximale. Pour cela, nous cherchons à maximiser ce score de correspondance en fonction des degrés de liberté (μ, Σ) :

$$\mu^*, \Sigma^* = \arg \max_{\mu, \Sigma} \psi(X, \mu, \Sigma) \quad (14)$$

Pour effectuer cette résolution, nous nous appuyerons sur la célèbre méthode de descente de gradient, avec pour fonction de perte à minimiser $-\psi(X, \mu, \Sigma)$.

²minimiser x revient à maximiser -x, on traitera dans ce projet les minimisations et maximisations de variables de la même manière

La fonction de gaussienne utilisée sera légèrement modifiée pour éviter le phénomène de sur-convergence (si on utilise la gaussienne telle quelle, elle peut converger avec une variance nulle et une amplitude infinie sur un seul point d'intérêt). Résoudre ce problème n'est pas simple, il n'y a d'ailleurs pas de solution mathématique. La littérature scientifique conseille d'ajuster intelligemment nos métriques, ou bien de relancer notre algorithme avec une nouvelle initialisation si le phénomène se reproduit (Christopher M.Bishop [13]). Il n'est pas possible non-plus de définir une amplitude constante, la gaussienne se mettrait à grandir à l'infini pour prendre en compte le plus de points possibles. La solution choisie est de redéfinir l'amplitude en fonction du déterminant de la covariance (précédemment, une relation inverse entre l'amplitude et la covariance) :

$$N_{modified}(\underbrace{x}_2, \underbrace{\mu}_2, \underbrace{\Sigma}_{2,2}) = N(\underbrace{|\Sigma|}_1, 0, \underbrace{v}_1) N_{limited}(\underbrace{x}_2, \underbrace{\mu}_2, \underbrace{\Sigma}_{2,2}) \quad (15)$$

v défini la plage de largeur que peut atteindre le déterminant de Σ de la gaussienne. Plutôt que de tendre vers 0 pour obtenir une amplitude infinie, Σ tendra vers un compromis entre largeur et amplitude lui permettant d'obtenir le plus de points possibles, sans pour autant trop s'écartez. Avant d'arriver à cette solution, nous avions exploré l'utilisation de la gaussienne à amplitude limitée pour résoudre le problème de l'amplitude infinie, mais malheureusement, elle a tendance à grossir à l'infini afin d'obtenir le plus de points possibles dans sa largeur.

7.1.2 Fonctions de pertes

La simple recherche de gaussiennes libres ne nous permettra pas d'obtenir un résultat exploitable. En effet, par la liberté offerte par la descente de gradient, chaque variable d'état est modifiée librement et indépendamment dans le but de minimiser la fonction de perte finale. Ce fonctionnement peut amener notre état dans des situations sans sens physique, comme décrit précédemment, une covariance nulle par exemple. Pour permettre de garder un état dans une configuration nominale, il faut rajouter des contraintes de variables.

On ajoute à la fonction de perte de mesure des critères permettant de garder notre mesure dans une plage de valeur proche du nominal, permettant de contraindre notre état à ne pas prendre des valeurs aberrantes. Nous définissons une multitude de fonction de pertes qui nous permettrons d'obtenir le comportement souhaité par nos gaussiennes :

- $Loss_{gauss_long}$: une gaussienne plus longue que large
- $Loss_{gauss_corr}$: une corrélation x et y la plus faible possible dans la covariance
- $Loss_{bed_spacing}$: un espacement minimum entre les gaussiennes (arbitrairement défini à 20 cm)

La descente de gradient finale se fera par une seule fonction de perte à minimiser :

$$Loss_{general} = Loss_{mesure} + Loss_{etat_mesure} \quad (16)$$

Avec pour la partie mesure la maximisation de la correspondance entre les données et le modèle :

$$Loss_{mesure} = -\psi(X, \mu, \Sigma) \quad (17)$$

Avec pour la partie de l'état de la mesure le fait de garder les gaussiennes au bon espacement avec une forme adéquate :

$$Loss_{etat_mesure} = a * Loss_{bed_spacing} + b * Loss_{gauss_corr} + c * Loss_{gauss_long} \quad (18)$$

Les coefficients (a, b, c) seront à déterminer empiriquement afin de régler le comportement des gaussiennes face à tous ces critères à minimiser.

Détaillons mathématiquement les fonctions de perte permettant à notre état de garder une configuration de fonctionnement nominal :

Loss_{gauss_long} : cette fonction de perte est plutôt simple à comprendre et à implémenter. On prend le ratio entre σ_x et σ_y , et on maintient cette valeur dans une plage nominale via sa projection dans une gaussienne négative, nous maintenons cette valeur dans le creux de la gaussienne. Plus la mesure actuelle sera différente de la mesure par défaut, plus la valeur de projection sera élevée, et par conséquent, la minimisation de ce critère viendra "tirer" le ratio vers sa valeur nominale, le "creux" de la gaussienne négative :

$$R = \frac{\sigma_x}{\sigma_y}; \quad Loss_{gauss_long} = N(R, R_{default}, \sigma_{loss_ratio}) \quad (19)$$

Loss_{gauss_corr} : fonction de perte similaire à *Loss_{gauss_long}*. Nous cherchons à placer la corrélation autour de zéro à l'aide de la méthode précédente :

$$\Sigma = \begin{bmatrix} C_{x,x} & C_{x,y} \\ C_{y,x} & C_{y,y} \end{bmatrix} = \begin{bmatrix} \sigma_x^2 & \sigma_x * \sigma_y * corr \\ \sigma_y * \sigma_x * corr & \sigma_y^2 \end{bmatrix} \Rightarrow corr = \frac{C_{x,y}}{\sqrt{C_{x,x}C_{y,y}}} \quad (20)$$

On aura finalement pour la fonction de perte qui maintient la corrélation proche de zéro :

$$Loss_{gauss_corr} = -N(corr, 0.0, \sigma_{loss_gauss_corr}) \quad (21)$$

Loss_{bed_spacing} : cette fonction de perte peut être vue comme une fonction de perte, qui définit entre 20cm et l'infini une fonction décroissante pour l'écart entre les 2 gaussiennes (noté Δ_y). Cette fonction tend vers l'infini en 20 cm. Nous utiliserons donc la fonction inverse, ce qui nous donnera :

$$Loss_{bed_spacing} = \frac{1}{(\Delta_y - 0.20)}, \quad \text{with } \Delta_y \in [0.20, +\infty[\quad (22)$$

On aura le risque que la valeur d'écartement de lit végétale passe en dessous de 20 cm, on aura également un écrêtage qui permet de garder cette valeur toujours dans une plage de fonctionnement nominal. On aura également un écrêtage au niveau de la covariance de la gaussienne, ce qui nous permettra de garder une covariance positive.

Ces différentes contraintes nous permettront par la suite d'obtenir un calcul de trajectoire correct à partir du modèle d'état. L'implémentation de cet algorithme d'optimisation doit satisfaire trois attentes :

- Une correspondance entre les données reçues et le modèle de mesure
- Un évolution de l'état du modèle restant dans une plage de fonctionnement nominal
- Un état du modèle permettant de calculer explicitement une trajectoire à suivre

Ces différents objectifs sont absolument nécessaires et doivent par conséquent intelligemment cohabiter, un objectif ne doit pas (trop) prendre le pas sur les deux autres. Pour se faire, on définit manuellement et empiriquement les pondérations entre les différents critères d'optimisation afin d'obtenir le comportement souhaité pour notre modèle d'état.

Les expérimentations ont montré une grande flexibilité de notre algorithme, que ce soit au niveau de la prise en compte de la donnée reçue, ou bien de l'impact des paramètres au niveau du comportement de l'algorithme. Pour permettre une adaptabilité plus concise et une implémentation plus proche du réel et plus proche de valeurs physiques explicites, on s'est peu à peu détourné d'un modèle de données totalement libre pour se diriger vers un modèle plus contraint, avec une plus grande prise en compte

de l'état *a priori* attendu. Ce choix a notamment été motivé par la difficulté de régler les différentes pondérations de critères pour la fonction de perte générale.

Une fois la descente de gradient effectuée, on obtient une mesure de notre état, qui est du même ordre de grandeur. La certitude de cette mesure est corrélée aux nombre de points présents dans chacune des gaussiennes. Nous avons choisi de faire évoluer un état en fonction d'une mesure dans le but de rendre plus fiable notre algorithme. Dans cette continuité, nous opterons pour un filtre de Kalman étendu dans son modèle d'évolution, et linéaire dans sa mesure (respectivement pour la prise en compte du déplacement du véhicule, par le même ordre de grandeur entre la mesure et l'état).

7.2 Filtre de Kalman

Afin de prendre en compte l'incertitude de notre mesure pour faire évoluer notre modèle d'état, nous nous va appuyerons sur la formulation et l'implémentation du filtre de Kalman. Nous allons l'expliquer pour la version V0 de nos gaussiennes (les gaussiennes totalement libres), ce qui nous permettra d'expliquer le cas général le plus complexe et de mettre en lumière le choix de ne pas utiliser cette représentation pour notre répartition de points d'intérêt. Nous nous concentrerons sur la recherche d'une seule gaussienne. Pour en prendre en compte deux, il suffit de soit étendre notre état et notre mesure, soit d'exécuter l'algorithme en deux fois (dans le cas où les gaussiennes de la rangée de gauche et de droite sont indépendantes l'une de l'autre).

Avec pour variables d'état :

$$X = \begin{bmatrix} \text{Position } x \\ \text{Position } y \\ \text{Covariance } xx \\ \text{Covariance } yy \\ \text{Covariance } xy \end{bmatrix} \quad (23)$$

On aura pour nos variables :

- $\underbrace{X}_{5,1}$: État représentant la répartition du nuage de points correspondant à une rangée de plante
- $\underbrace{P}_{5,5}$: Incertitude d'état
- $\underbrace{z}_{5,1}$: La mesure reçue
- $\underbrace{Q}_{5,5}$: Covariance d'évolution entre l'état k et k+1
- $\underbrace{R}_{5,5}$: Covariance de la mesure
- $\underbrace{H}_{5,5}$: Lien entre l'état et la mesure, dans notre cas, l'identité car l'état est du même espace que la mesure
- $\underbrace{F}_{5,5}$: La jacobienne de transition entre l'état k et k+1
- $\underbrace{u}_{3,3}$: La commande de déplacement du robot, odométrie filtrée issue de la fusion GPS, IMU et rotation des roues, déplacement considéré comme un mouvement plan en 2D sur le plan XY (vue du ciel)

La prédiction de notre nouvel état à partir du déplacement du robot sera formulée de la façon suivante:

$$F_k = \frac{\partial \text{Deplacement_robot}(X_{k|k}, u_k)}{\partial X} |_{X_{k|k}, u_k} \quad (24)$$

$$X_{k+1|k} = \text{Deplacement_robot}(X_{k|k}, u_k) \quad (25)$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q \quad (26)$$

La fonction $\text{Deplacement_robot}(X_{k|k}, u_k)$ transforme les positions de l'état afin de les déplacer dans le repère robot à partir du déplacement du robot dans le repère monde.

La mise à jour de notre état à partir de la mesure reçue sera formulée de la façon suivante :

$$y_k = z_k - H X_{k+1|k} \quad (27)$$

$$S_k = H P_{k+1|k} H^T + R \quad (28)$$

$$K_k = P_{k+1|k} H^T S^{-1} \quad (29)$$

$$P_{k+1|k+1} = (I - K_k H) P_{k+1|k} \quad (30)$$

$$X_{k+1|k+1} = X_{k+1|k} + K_k y_k \quad (31)$$

Le filtre de Kalman nous permettra d'utiliser l'état de notre modèle comme une représentation de la répartition des points d'intérêt correspondant aux plantes détectés par le furrow. La prise en compte de l'incertitude permet de ne pas prendre en compte les mesures erronées, ou bien celle avec peu de points (la covariance d'incertitude de mesure R évolue en fonction du nombre de points d'intérêt présente dans la gaussienne) :

$$R_{k+1} = \frac{R_{init}}{1 + \sum_{i=0}^n N(V_i, M_{k+1})} \quad (32)$$

L'incertitude de mesure est définie par l'incertitude d'initialisation divisé par les projections des points sur la gaussienne définie par la mesure actuelle. Plus on aura de points à l'intérieur de celle ci, plus la mesure délivrée par la gaussienne aura une incertitude faible. Ce processus nous permettra d'obtenir un état fiable sur lequel baser notre création de trajectoires. Finalement, l'utilisation d'une mesure par maximisation entre un modèle et une donnée, ainsi que l'évolution de notre état avec la prise en compte des différentes incertitudes de nos variables grâce au filtre de Kalman nous permet d'obtenir un fonctionnement robuste et précis de notre localisation.

7.3 Calcul de trajectoire

Une multitude de critères peuvent être employés pour le calcul de trajectoire visant à suivre un élément. Nous en retiendrons trois :

- La robustesse de suivi : garder à portée de perception l'élément à suivre
- La qualité de suivi : minimiser l'écart latéral et angulaire au niveau de l'essieu arrière
- La faisabilité du suivi : faire en sorte que la trajectoire proposée soit physiquement réalisable pour les actionneurs

Nous devons faire en sorte de calculer une trajectoire pour tout type de mixture de gaussiennes.

Une approche efficace pour la fusion des données consiste à utiliser la moyenne pondérée des gaussiennes. Dans un premier temps, nous définissons une zone de focus arbitraire le long de la trajectoire, puis nous pondérons les différentes gaussiennes en fonction de leur position par rapport à cette zone de focus (figure 60). Nous avons expérimenté l'utilisation d'une seule paire de gaussiennes pour définir la trajectoire dans les versions V0, V1 et V2. Cependant, cela a conduit à des trajectoires instables et oscillantes, et

l'augmentation de la taille des gaussiennes a affecté la prise en compte de la courbure (une gaussienne trop longue n'arrive pas à rester sur une seule rangée lors de courbures importantes). C'est pourquoi nous avons opté pour la solution de la V3, qui prend en compte une multitude de gaussiennes par moyenne pondérée pour obtenir des résultats plus satisfaisants.

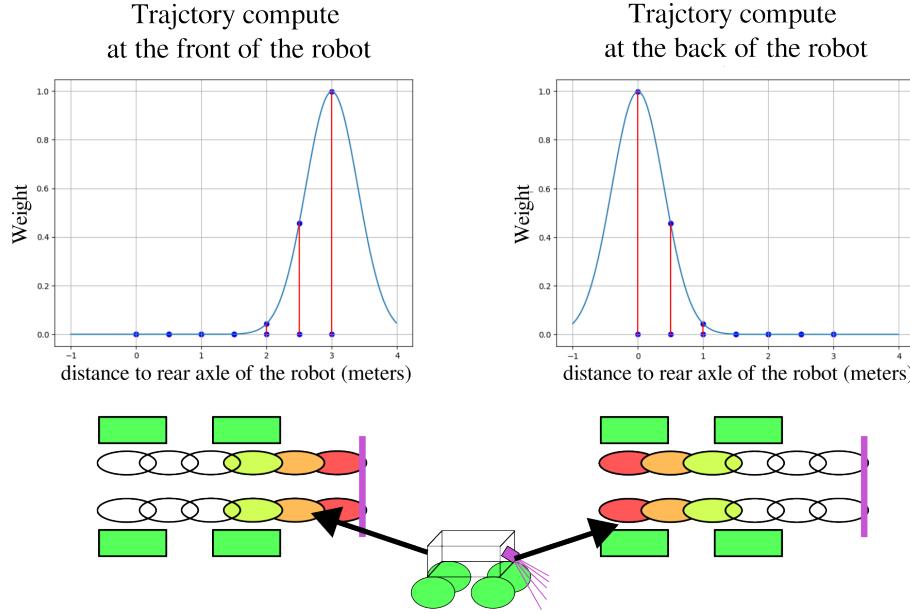


Figure 60: Placement de la zone de focus de calcul de trajectoire, à gauche, proche de l'essieu arrière à 0.0 mètres, à droite, proche de la perception LiDAR devant le robot à 3.0 mètres

Une fois notre zone de focus définie, on pondère les coefficients afin d'obtenir une moyenne des trajectoires en fonction de la zone choisie. Cette méthode s'avérera efficace et stable, mais le projet étant amené à évoluer vers une prise en compte de trajectoire courbée, on va donc évoluer vers une méthode plus explicite de calcul de trajectoire. On va approximer la mixture de gaussiennes par un polynôme de degrés 2 via une régression quadratique faisant intervenir les moindres carrés (figure 61). Puis, une fois la courbe définie, on linéarise en un point pour obtenir l'écart angulaire et latérale.

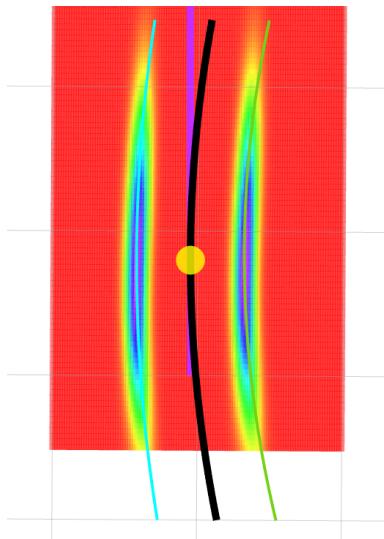


Figure 61: Schéma de création de trajectoire à partir d'une courbe d'ordre 2, les courbes bleus et vertes représente les deux gaussiennes, la courbe centrale est l'approximation générale de la courbure, la droite violette correspond à la trajectoire linéarisée tangent au point jaune

Face à cette nouvelle manière de calculer les écarts angulaires et latéraux à notre trajectoire, on devra évaluer en terme de précision ces améliorations et l'impact des différents paramètres. Le but sera de déterminer l'algorithme ainsi que ses paramètres permettant précision et robustesse dans le suivi de rangées de plantes en contexte agricole.

7.4 Expérimentations

Lors de nos expérimentations, nous avons comparé deux points en particulier :

- la comparaison entre la création de trajectoire par pondération de gaussiennes ou linéarisations de polynôme obtenu par régression d'ordre 2
- le placement de calcul de trajectoire par rapport à l'essieu arrière

Afin de comprendre les différences entre chaque algorithme pour chaque paramètre étudié, nous évaluons les écarts à la trajectoire des différentes configurations sur une même courbure à suivre (figure 62).

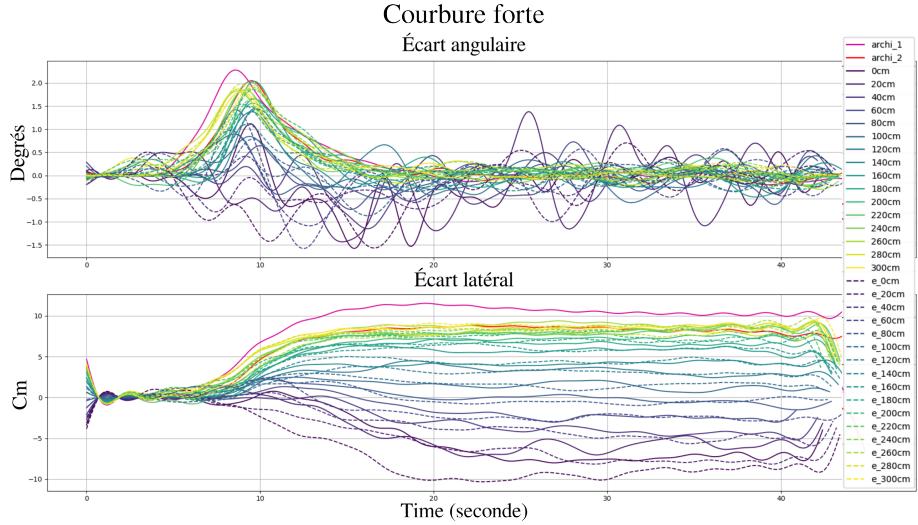


Figure 62: Visualisation des trajectoires selon le placement et l'architecture de création de trajectoire, écart latéral et angulaire au niveau de l'essieu arrière, archi_1 = architecture 1, archi_2 = architecture 2

Nous pouvons constater plusieurs choses :

- Un résultat similaire entre les 2 méthodes de création de trajectoire
- L'écrasement de la trajectoire si on regarde trop loin, l'écartement à la courbure si on regarde trop proche
- L'architecture 1 et 2, basées sur le calcul de trajectoire par régression linéaire des points d'intérêts classifiés, correspondant à une perception en avant de l'essieu arrière

Par la visualisation du résultat des différentes perceptions, il est difficile de discriminer quel réglage est le meilleur pour obtenir un suivi optimal dans le cadre agricole. Nous décidons donc d'évaluer l'accumulation d'erreur et de comparer les résultats sous la forme d'histogrammes (figure 63).

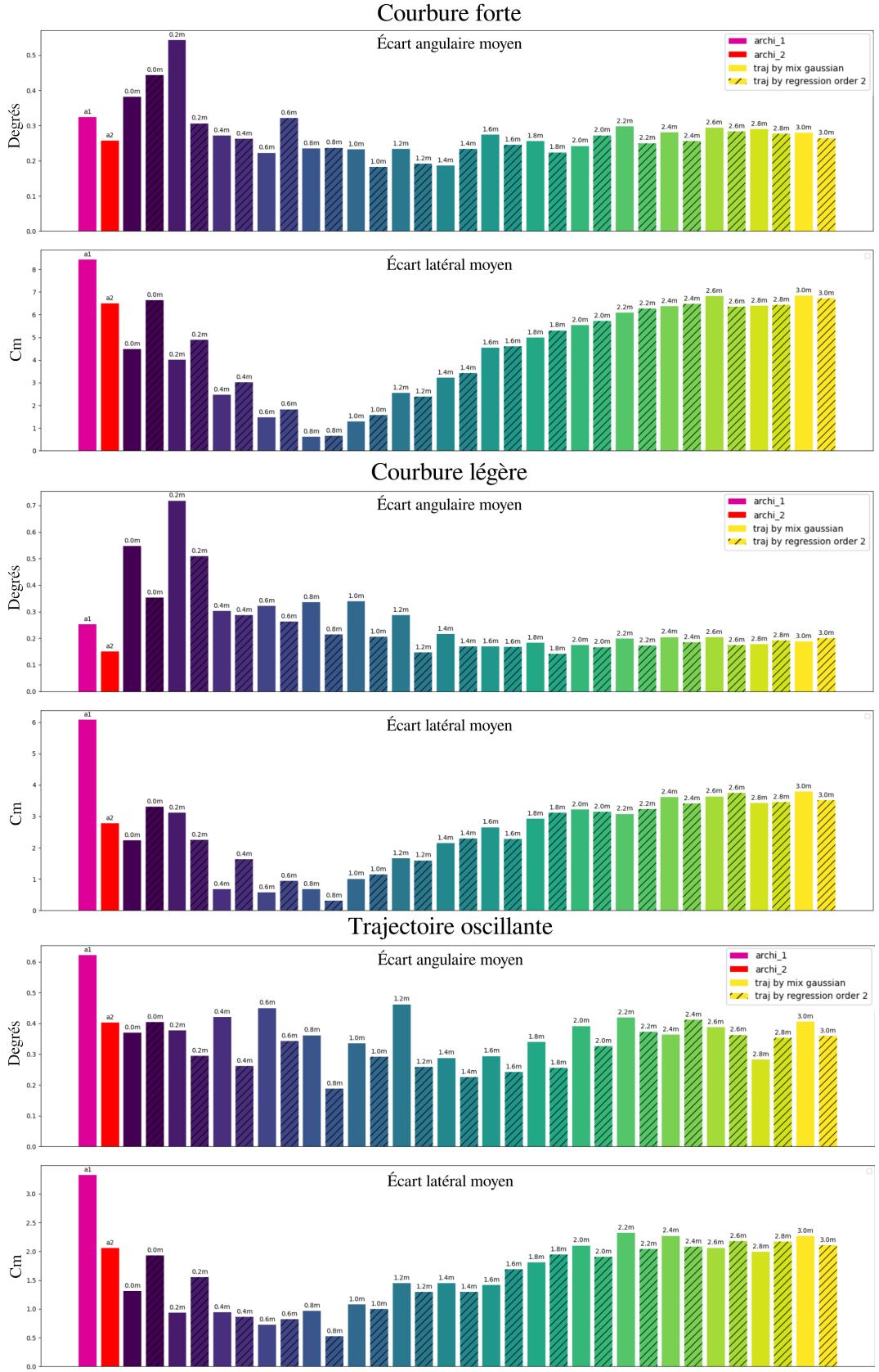


Figure 63: Visualisation de l'accumulation de l'erreur à la trajectoire selon le placement et l'architecture de création de trajectoire, écart latéral et angulaire au niveau de l'essieu arrière, a1 = architecture 1, a2 = architecture 2

Nous constatons beaucoup plus explicitement les différences entre les différentes configurations, nous pouvons affirmer que la création de trajectoire est similaire, bien qu'elle ne soit pas totalement la même et que l'on constate des différences résiduelles. Nous arrivons alors à la conclusion qu'une perception basée sur la linéarisation à 80 cm de l'essieu arrière du polynôme formé par la régression quadratique du placement des gaussiennes est la meilleure pour assurer un suivi précis au niveau de l'essieu arrière.

Dans un second temps, nous évaluons le gain de robustesse de l'algorithme :

Table 10: Comparaison entre les différentes architectures

	Validation	Test
Natif	75%	45%
Archi 1	75%	54%
Archi 2	100%	82%
Archi 3	83%	45%
Archi 4	100%	91%

Table 11: Tableau récapitulatif du taux de succès résolution des mondes de test, taux de succès exprimé en proportion de rangées de plantes correctement suivies, architecture actuelle surlignée

À travers nos expérimentations, nous avons observé que, compte tenu des importantes courbures présentes dans nos environnements de test, il était nécessaire d'adopter une perception prédictive positionnée à une distance de 200 cm de l'essieu arrière, afin de maintenir la trajectoire souhaitée, au détriment de la précision à l'essieu arrière.

Par l'analyse précédente, Nous avons pu observer que la création d'une trajectoire native du furrow avait le comportement d'une création de trajectoire placée en avant du tracteur. Ceci permet d'expliquer, malgré ses divers défauts, la relative robustesse de suivi du furrow originel, au prix d'une perte de précision au niveau de l'essieu arrière.

Les expérimentations sur le placement de création de trajectoire mettent en évidence la plus grande faiblesse de notre algorithme. Nous devons faire un compromis entre la précision et la robustesse. Le placement de la création de trajectoire à 80 cm offre la plus grande précision, mais en cas de forte courbure, elle risque de manquer de robustesse. Ce défaut n'est pas dommageable dans le cas réel : les courbures sont bien moins forte dans le monde agricole que dans nos bases de test. Une création de trajectoire à 80cm nous permettra d'assurer une précision sans pour autant compromettre la robustesse de notre algorithme de suivi.

7.5 Conclusion

Dans cette architecture, nous avons pu, par une approche probabiliste, mieux appréhender la problématique de robustesse de perception. Nous avons pu explorer et se rendre compte qu'une approche purement déterministe n'était peut être pas la meilleure solution, que l'algorithme était susceptible de faire des erreurs. Plutôt que de chercher à réduire un maximum les erreurs de notre algorithme, nous nous sommes questionnés sur la possibilité d'utiliser une méthode pouvant se corriger indépendamment des décisions passées.

Cette architecture nous aura permis d'obtenir un algorithme robuste par son approche probabiliste, ainsi qu'un algorithme précis pour son adaptabilité dans la création de trajectoire. Nous pouvons choisir plus précisément quelle partie de la trajectoire doit se concentrer notre algorithme, avec dorénavant des outils permettant d'évaluer la pertinence de ces choix.

Dans l'attente d'expérimentations réelles, nous espérons que l'architecture 4 constituera la pièce maîtresse de ce projet. Capitalisant sur les avantages de la détection végétale pour façonner la création et le suivi de trajectoire par une approche probabiliste de la localisation, cette solution allie robustesse, précision et pertinence pour notre nouvel architecture général de perception et de suivi de trajectoire.

8 Conclusion de stage

Lors de ce stage, nous avons pu découvrir le domaine de la perception artificielle en explorant en profondeur l'algorithme du furrow. Nous avons pu expérimenter l'apport de l'intelligence artificielle à travers divers architectures, ce qui nous a permis, au fur et à mesure de déterminer les grandes problématiques de l'algorithme original qui nuisent à ses performances finales. Dans un second temps, après l'exploration de l'intelligence artificielle, nous avons décidé de nous tourner vers des algorithmes plus classique telles que l'analyse géométrique, l'optimisation et la fusion de données. On a changé, depuis la perception LiDAR, jusqu'au calcul de la trajectoire finale, toute la chaîne de traitement afin d'obtenir un algorithme plus robuste (voire figure 64).

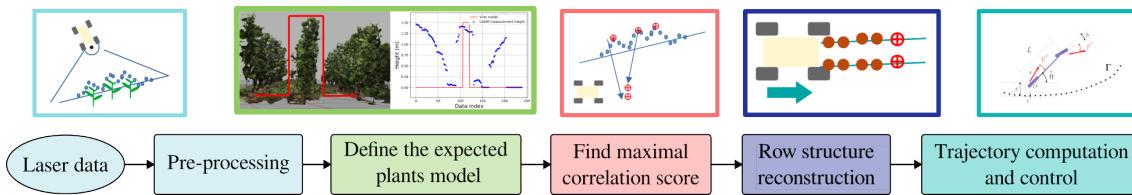


Figure 64: Architecture du furrow V1, version initiale de début de stage

Nous aurons donc pour version finale, cette architecture. Chaque étape aura des opérations adaptées à la perception et à la création de trajectoires en milieu agricole (voir figure 65).

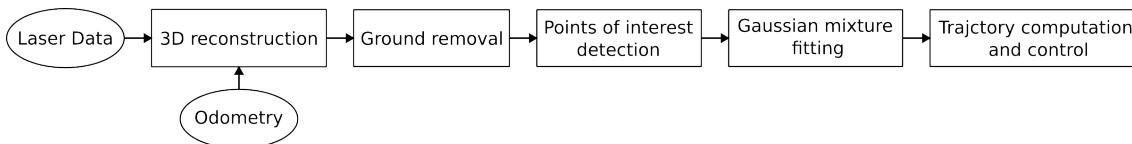


Figure 65: Architecture du furrow V2, version final de fin de stage

8.1 Apport de l'intelligence artificielle dans la robotique agricole

Concernant l'utilisation de l'intelligence artificielle dans le domaine de la robotique agricole, nous nous sommes malheureusement heurtés aux diverses limitations technique et fonctionnelle de l'intelligence artificielle. Nous n'avons pas réussi à mettre en lumière un impact positif de l'intelligence artificielle. La généralisation d'un réseau de neurones de sa base de données vers la réalité est une tâche difficile, surtout dans le cas de capteur difficile à simuler réaliste (le LiDAR se base sur la profondeur de ce qu'il capte, il est difficile de créer une base de données d'objets avec une structure géométrique fortement détaillé et réaliste).

Ces limitations résultent principalement des décisions techniques prises et de l'orientation adoptée par ce projet. Il est fortement probable que l'intelligence artificielle puisse contribuer à résoudre certaines problématiques, potentiellement en proposant des approches que nous n'avons pas encore expérimentées ou envisagées.

8.2 Analyse Forces-Faiblesses et Risques-Opportunités du projet mené

Forces

Une avancé théorique majeure durant ce projet est l'approche perceptive probabiliste. Elle permet de modéliser plus précisément nos problèmes de perception en faisant intervenir un algorithme d'optimisation. Cette méthode permet de prendre en compte toute la spécificité des tâches agricoles, prendre en compte les outliers, les fausses détections, évitant les minima locaux, les aberrations physiques. C'est une direction

qui était déjà largement utilisée dans le cadre de l'automatique, de la traversabilité, mais son introduction dans la partie perception nous permet une plus grande robustesse dans le suivi de trajectoire.

Faiblesses

L'utilisation de l'intelligence artificielle représente une faiblesse de ce projet. Nous n'avons pas pu trouver une méthode nous permettant d'obtenir de bons résultats à partir de celui-ci. Nous aurions pu adopter d'autres manières d'utiliser l'intelligence artificielles pour répondre à nos problématiques, soit par l'utilisation de réseau de neurones plus avancés ou bien des paradigmes de fonctionnement totalement différents, soit par le changement complet de notre architecture de perception et de contrôle de nos robots.

Une grande faiblesse de ce projet est également l'élaboration de base de données proches du réel. Nous n'avons pas réussi à trouver une méthode nous permettant de générer de la donnée réelle annotée. C'est un des points essentiels de l'utilisation d'algorithmes d'apprentissage en robotique agricole, il nous faut une base d'entraînement proche de la tâche à accomplir. Cette problématique pourrait potentiellement être également résolue par l'utilisation de l'intelligence elle-même, nous permettant d'avoir une base d'entraînement solide pour nos futures algorithmes.

Risques

Une particularité de notre projet est que l'algorithme sera passé d'une version totalement en C++, à une version totalement en Python. Ce choix peut paraître inadapté dans le cadre d'une exécution en temps réel, mais par les besoins de traitement massif de données sous forme d'opérations matricielles, et la facilité qu'offre python avec la bibliothèque Pytorch, on obtient un programme plus simple à maintenir pour la suite, ainsi qu'une vitesse d'exécution correcte pour de l'exécution en temps réel (il est tout à fait possible de faire des opérations matricielles sur carte graphique en C++, mais c'est plus complexe et chronophage d'implémentation). Finalement, les plus grands dangers auxquels peut être confronté ce projet sont des problématiques futurs qui nécessiteraient de remettre en question toute l'architecture de perception et de contrôle, comme ça a été le cas pour la version précédente du furrow.

Opportunités

Malgré diverses améliorations, nous avons pu constater au cours de ce projet diverses problématiques qui mériteraient d'être approfondies lors de futurs travaux au sein de l'équipe Roméa.

Une opportunité mise en lumière par ce projet serait une approche multi-points de contrôle de la trajectoire, permettant ainsi de définir une commande basée sur la perception permettant de satisfaire au mieux le placement de chaque partie du robot. En effet, dans ce projet, et plus largement dans la recherche agricole actuelle, nous partons du principe qu'il faut placer l'essieu arrière sur la trajectoire, que l'essieu avant est libre d'effectuer tout mouvement pour permettre cet objectif, et que l'outil arrière quant à lui, sera correctement positionné si on assure un bon suivi de trajectoire avec peu d'oscillations. Ce paradigme fonctionne largement dans le cas nominal, mais lors des manœuvres, il est possible que l'essieu avant écrase les rangées voisines afin de bien positionner l'essieu arrière, sans même que l'outil soit lui correctement positionné.

8.3 Perspectives

Ce projet nous aura permis de nous plonger dans le milieu de la recherche à travers la manipulation et l'évaluation de différents domaines dans le but de résoudre une problématique donnée. Ce projet aura été riche d'enseignements par l'application de compétence purement théorique, dans le cadre expérimental réel.

De nouvelles voies sont également ouvertes par la clôture de ce projet : Nous sommes désormais beaucoup mieux informé sur l'apport de l'intelligence artificielle dans le domaine de la perception et du contrôle en robotique agricole. Notre expertise développée nous permettra d'être d'autant plus efficace à

l'avenir face aux problématiques de demain, nous permettant rigueur académique et universitaire face à des projets de plus grandes envergures.

Citations projet

- [1] Pierre C., Lenain R., Laneurit J., and Rousseau V. A multi-control strategy to achieve autonomous field operation, 2022.
- [2] Ruizhongtai Qi C., Su H., and J. Guibas L. Mo K. Pointnet: Deep learning on point sets for 3d classification and segmentation. <<https://dblp.org/rec/journals/corr/QiSMG16.bib>>, 2016.
- [3] Nived Chebrolu1, Philipp Lottes1, Alexander Schaefer, Wera Winterhalter, Wolfram Burgard, , and Cyrill Stachniss. Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields, 2017.
- [4] Iberraken D., Gaurier F., Roux J.C., Chaballier C., and Lenain R. Autonomous vineyard tracking using a four-wheel-steering mobile robot and a 2d lidar. <<https://doi.org/10.3390/agriengineering4040053>>, 2022.
- [5] Mathieu Deremetz, Roland Lenain, and Benoit Thuilot. Path tracking of a two-wheel steering mobile robot : An accurate and robust multi-model off-road steering strategy, 2018.
- [6] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.
- [7] Ashley Hill, Jean Laneurit, Roland Lenain, and Eric Lucet. Online gain tuning using neural networks: A comparative study.
- [8] Jawad Iqbal, Rui Xu, Shangpeng Sun, and Changying Li. Simulation of an autonomous mobile robot for lidar-based in-field phenotyping and navigation, 2020.
- [9] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator, 2004.
- [10] Kamel Lahssini. Fusion de données lidar aéroporté et sentinel-2 par apprentissage pour la caractérisation de la ressource forestière du pnr des bauges.
- [11] Melissa Latella, Fabio Sola, and Carlo Camporeale. A density-based algorithm for the detection of individual trees from lidar data.
- [12] Altieri M.A. Agroecology : The science of sustainable agriculture.
- [13] Christopher M.Bishop. Pattern recognition and machine learning, page 434, 2006.
- [14] Jean-Matthieu Monnet, Émilie Chirouze, and Éric Mermim. Estimation de paramètres forestiers par données lidar aéroporté et imagerie satellitaire rapideye : Étude de sensibilité.
- [15] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. <<https://doi.org/10.48550/arXiv.1706.02413>>, 2017.
- [16] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. <<https://doi.org/10.48550/arXiv.1711.06396>>, 2017.

Annexe : état de l'art sur la segmentation du nuage de points dans le domaine de la robotique agricole

Une solution au traitements de nuage de points qui a fortement été étudiée lors de ce projet est d'attribuer les points de notre nuage 3D à des groupes distinct tel que le sol, les plantes, les obstacles. C'est ce qu'on appelle la segmentation sémantique, on attribue une classe à chacun des points dans notre nuage, on forme des groupes représentant un type d'élément. Pour permettre la création de trajectoires pour le suivi de rangées de plantes, il est également intéressant de séparer les plantes les unes des autres, c'est ce qu'on appelle la segmentation d'instances, nous cherchons à différencier plusieurs instances d'une même classe d'objets (figure 66).

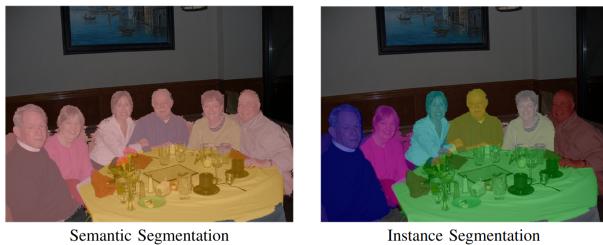


Figure 66: Illustration de la différence entre la segmentation sémantique et la segmentation d'instances [17]

Le mixe des deux méthodes s'appelle la segmentation panoptique, on voudrait à la fois séparer les points par instance, mais aussi attribuer à ces instances une classe. Cette segmentation permet d'avoir une compréhension complète de notre environnement, afin de prendre les décisions de contrôle adéquate de notre engin agricole (c'est également le cas dans le domaine du véhicule autonome).

Divers méthodes de segmentations existent, notamment pour le traitements d'image. Nous allons les étudié et estimé leurs forces et faiblesses afin de les transposer au cas du traitement de nuage de oints 3D pour la segmentation de rangées de plantes.

Tout d'abord, nous avons la segmentation d'image par grille qui consiste à diviser une image en plusieurs cases ou grilles de taille égale, puis à appliquer un algorithme d'intelligence artificielle pour déterminer les zones d'intérêt dans chaque case. Cette méthode permet de segmenter l'image de manière précise et efficace. L'algorithme d'intelligence artificielle peut utiliser différentes techniques pour segmenter chaque case, telles que la détection de contours, la reconnaissance de formes ou l'apprentissage profond (figure 67). Une fois que chaque case est segmentée, les résultats peuvent être combinés pour obtenir la segmentation de l'image complète. C'est un processus itératif, on est obligé de passer plusieurs fois sur nos objets pour pouvoir être capable de distinguer les différentes classes. De plus, cette technique est bien adapté pour le traitement d'images, mais si on l'applique aux nuages de points 3D, ça devient très lourd de discriminer le nuage de cette façon, on doit former des voxels assez petit pour pouvoir avoir une frontière entre les différents groupes de points.

Face aux similitudes entre différentes instances et à leur possible chevauchement dans un nuage de points 3D, il est difficile pour un réseau de neurones de différencier avec précision des rangées de plantes dans un contexte agricole.

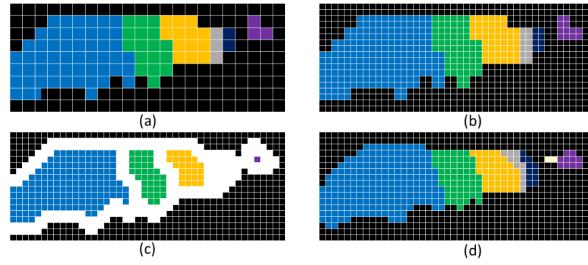


Figure 67: Segmentation par délimitations successives de la classe, processus itératif, [20]

Nous avons également la segmentation par silhouette, qui est une méthode de segmentation d'image qui consiste à détecter et à isoler des objets dans une image en utilisant leurs silhouettes (figure 68). Cette méthode est souvent utilisée pour segmenter des objets dans des images complexes où les contours ne sont pas clairement définis. Une fois que les silhouettes des objets sont isolées, elles peuvent être utilisées pour segmenter l'image en différents objets ou régions. Cette méthode est difficilement applicable au cas du nuage de points 3D éparse.



Figure 68: Ségmentation d'instance par traçage d'une silhouette de chaque objet, [21]

Deux approches courantes de segmentation d'images pourraient nous être utile dans notre tâche. :

- la segmentation basée sur une proposition (ou segmentation guidée)
- la segmentation sans proposition (ou segmentation non guidée)

La segmentation basée sur une proposition consiste à utiliser des informations externes pour guider le processus de segmentation. Ces informations peuvent être fournies sous forme de propositions de région ou de contour fournis par un utilisateur ou un autre algorithme. L'algorithme de segmentation utilise ensuite ces propositions pour diviser l'image en régions ou segments.

La segmentation sans proposition, en revanche, ne nécessite pas d'informations externes pour guider le processus de segmentation. L'algorithme de segmentation utilise des caractéristiques intrinsèques de l'image, telles que la couleur, la texture ou la forme, pour diviser l'image en régions ou segments.

La principale différence entre ces deux approches est que la segmentation basée sur une proposition est plus contrôlée et peut fournir des résultats plus précis, mais elle nécessite des informations externes. La segmentation sans proposition est plus autonome et peut fonctionner avec des images sans aucune information préalable, mais les résultats peuvent être moins précis.

Malheureusement, dans le cas de la segmentation de nuages de points 3D dans un contexte agricole, les nuages de points sont trop ressemblant pour formuler des propositions différentes pour chaque groupes, et trop proche pour pouvoir élargir correctement le groupe sans venir croisé avec les autres groupes (figure 69).

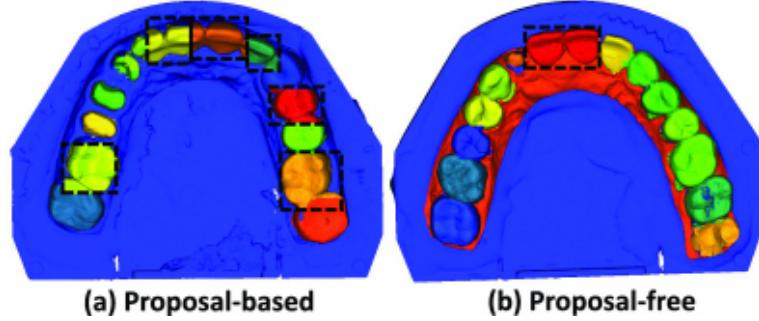


Figure 69: Mise en lumière de la difficulté de segmenter des instances qui se ressemblent, se touchent et se confondent, [19]

L’élargissement d’instances pour les nuages de points se ferait par des méthodes locales de proche voisins pour finaliser la segmentation. Cette méthode consiste à trouver les k points les plus proches de chaque point dans le nuage de points et à les regrouper en fonction de certaines propriétés, telles que la distance ou la couleur.

La segmentation basé proposition présente la qualité de pouvoir se baser sur des informations précédente, permettant de garder en mémoire pour notre réseau de neurones le placement des éléments de la scène, et ne pas repartir de zéro à chaque segmentation effectuée.

Malheureusement, cette méthode ne peut pas être utilisée dans le cadre agricole, les plantes formant souvent des ponts les unes avec les autres, la discrimination par les plus proche voisin serait impossible. Cette méthode marche bien quand il s’agit d’objet ou de personne dans un environnement structuré (figure 70).



Figure 70: Segmentation basée sur une proposition, on attribue un classe au centre de chacun des objets, puis on propage la classification par clusterisation par plus proche voisin, [18]

Malgré ces difficultés, nous avons tout de même réussi à implémenter les 2 segmentations (figure 71), mais comme soulevé précédemment dans ce projet, ce ne sera pas la solution privilégiée pour le contrôle robotique en milieu agricole.

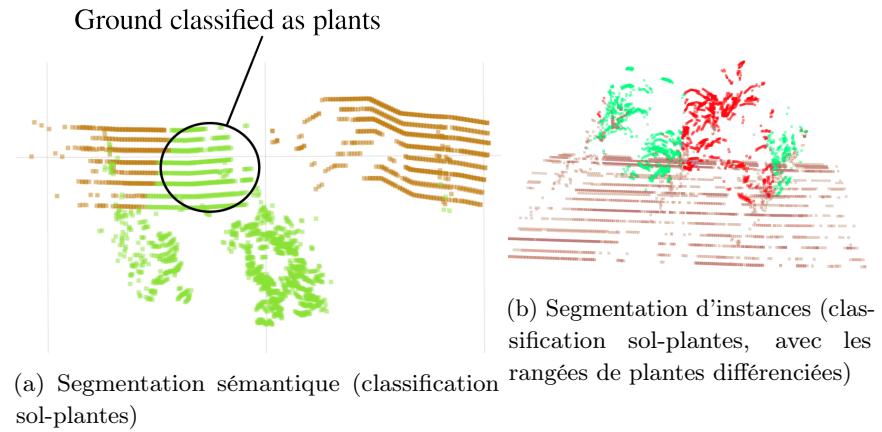


Figure 71: présentation et erreurs des ségmentation dans ce projet

Quand les données présentées à notre réseau de neurones diffère de sa base d'entraînement, la segmentation comporte divers erreurs et il alors impossible d'utiliser un réseau de neurones dans la robotique agricole et d'espérer obtenir un résultat robuste et précis.

Annexe : Segmentation sol-plantes

Suite à nos recherches bibliographique, afin d'utiliser l'intelligence artificielle dans le cadre le plus robuste et exploitable possible, nous avons décidé de ne faire qu'une segmentation entre le sol et les plantes.

Afin d'augmenter la robustesse de notre réseau de neurones, on s'est notamment intéressé à la modification de la fonction de perte, et l'analyse de caractéristiques locales de notre nuage de points, afin d'améliorer le résultat final de nos classifieurs.

Réseau de neurones pour la segmentation

Afin d'effectuer la classification sol plante du nuages de points 3D nous allons avoir besoin d'un réseau de neurones prenant en compte le nuage de point 3D, puis attribuant un label à chacun des points. Il prendra en entrée un tableau de N points sur 3 dimensions, et renverra en sortie un tableau de N points sur 2 probabilités d'appartenances (pour chacun des points : la probabilité d'appartenir au sol, la probabilité d'appartenir aux plantes) :

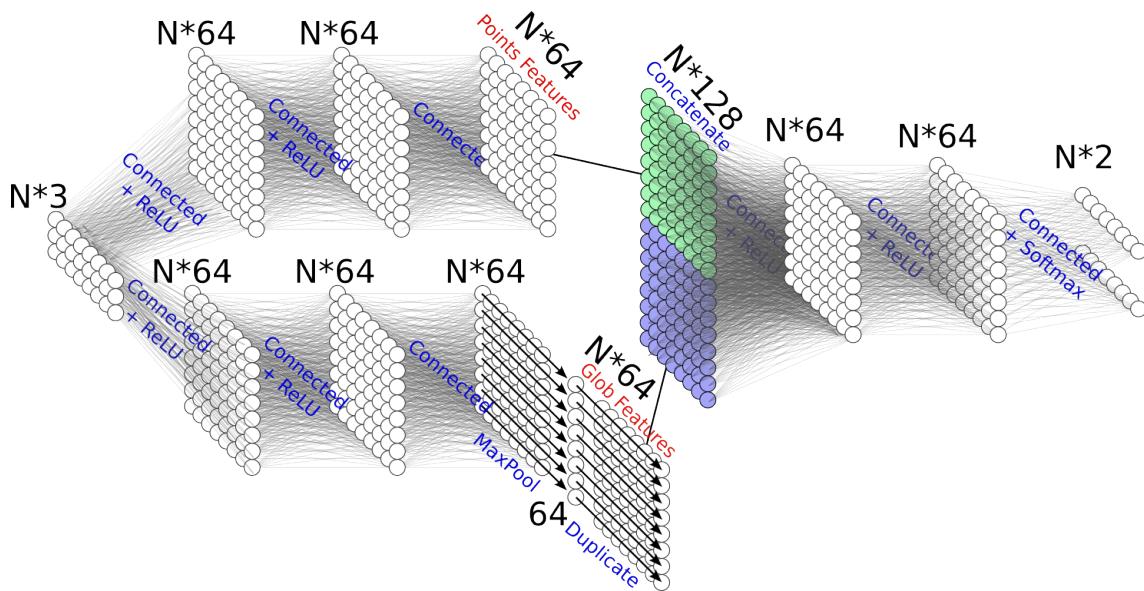


Figure 72: Schéma du réseau de neurones de segmentation, basé sur le pointnet

Plus précisément, on peut voir le réseau de neurones en 3 opérations distinctes :

- L'extraction de caractéristiques locales
- L'extraction de caractéristiques globales
- La segmentation à partir des caractéristiques global et local

Ces opérations se détailleront de la façon suivante :

- Extraction des caractéristiques local du nuage de points

On traite indépendamment les N points afin de les projeter en F_l caractéristiques, on obtient un tableau de caractéristiques en (N, F_l) ,

- Extraction des caractéristiques global du nuage de points

On traite indépendamment les N points afin de les projeter en F_g caractéristiques, on obtient un tableau de caractéristiques en (N, F_g) ,

On prend le maximum des F_g caractéristiques à travers les N points, on obtient un vecteur de caractéristiques en (F_g)

Pour permettre aux caractéristiques locales et globales de fusionne, on duplique le vecteur de F_g caractéristiques en N pointes, on obtient un tableau de caractéristiques en (N,F_g))

- Segmentation à partir des caractéristiques global et local en 2 classes

Pour chaque point, on concatène les F_l caractéristiques locales et les F_g caractéristiques globales, on obtient un tableau en (N,F_l+F_g)

On traite les caractéristiques des points indépendamment les uns des autres (points par point), on obtient alors un tableau de $(N,2)$ décrivant pour chaque point les probabilités d'appartenir au sol ou aux plantes

Une fois le réseau de neurones implémenté, on l'entraîne sur une base de données composé de nuages de points où on connaît la classe de chacun des points. Puis enfin, pour valider les fonctionnements du classifieur, on teste la segmentation sur un nuage de point présent dans l'entraînement, et on visualise également les points ayant été mal classifiés (figure 73).

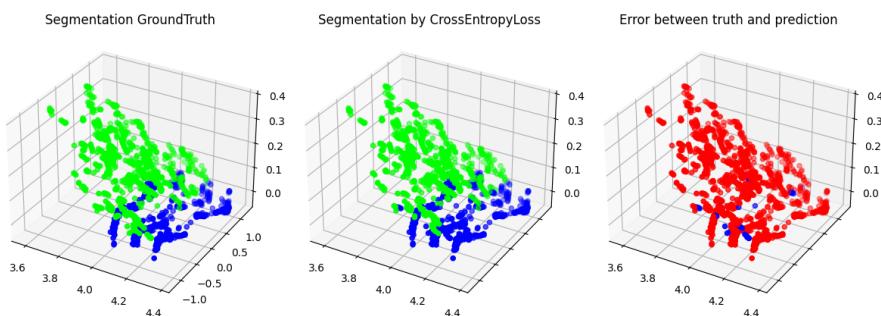


Figure 73: Nuage de points segmenté par le SegPointNet avec une CrossEntropy, réalité terrain à gauche, classification par réseau de neurones au milieu, erreurs de classification signalées en bleu sur rouge à droite

Fonctions de perte pour la segmentation

On obtient un résultat satisfaisant dans sa globalité, mais suite à diverses expérimentations, on s'est rendu compte que le réseau de neurones avait tendances à écraser les petites plantes et à tout classifier en tant que sol. Pour nous permettre de résoudre cette problématique, nous avons reproduit cette erreur sur un nuage de point créer et annoté manuellement (figure 74).

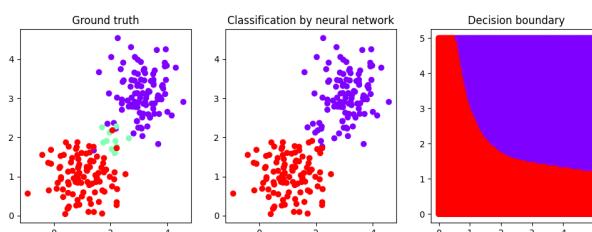


Figure 74: Segmentation ratée d'un nuage de points en 3 classes, les classes les plus présentent viennent écraser celle la moins présente

Dans le cas où une classe est beaucoup moins présentes qu'une ou plusieurs autres, elle se fera facilement éliminer par le réseau de neurones. C'est notamment le cas dans le traitement LiDAR agricole, la globalité

des faisceaux LiDAR impacte le sol, les plantes sont quant à elles plus compliquées à discerner, c'est une tâche complexe et peu rentable pour le réseau de neurones de classifier correctement les plantes. Le réseau de neurones préférera prédire que tous le nuage est du sol pour se tromper le moins possible, il sera juste à 90%, il va difficilement converger à segmenter correctement les plantes, au risque de classifier du sol en tant que plantes.

On peut le vérifier dans la fonction de perte, on fait juste la somme pour tous les points de la perte de classification, les points d'une classe peu présente impacte proportionnellement la fonction de perte globale (la SegCrossEntropy).

Notre fonction de SegCrossEntropy compare la prédiction (Y_p) avec la réalité encodée (Y_e) pour tous les points (la CrossEntropy ne s'occupe que d'une seule prédiction) :

$$\text{SegCrossEntropy}(Y_p, Y_e) = - \sum_{i=0}^{N_p} \sum_{c=0}^{N_c} Y_{e,i,c} * \log(Y_{p,i,c}) \quad (33)$$

Avec N_c classes :

$$C = [sol, plantes] = [0, 1] \quad (34)$$

Pour notre réalité terrain, nos classes connues, on aura le vecteur Y avec le numéro de la classe, ainsi que le tableau Y_e avec la version encodé de Y :

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \dots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} ; \quad Y_e = \begin{bmatrix} Y_1 == 0 & Y_1 == 1 \\ Y_2 == 0 & Y_1 == 1 \\ \dots \\ Y_n == 0 & Y_1 == 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ \dots \\ 1 & 0 \end{bmatrix} \quad (35)$$

Notre classifieur fera des prédictions d'appartenance de classe appelées Y_p :

$$Y_p = \begin{bmatrix} P_{X_1 \in C_0} & P_{X_1 \in C_1} \\ P_{X_2 \in C_0} & P_{X_2 \in C_1} \\ \dots \\ P_{X_n \in C_0} & P_{X_n \in C_1} \end{bmatrix} \quad (36)$$

Les prédictions se feront à partir des coordonnées des points. On aura pour points la représentation sur la matrice X en N_p points (chaque point X_i est un vecteur) :

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_{N_p} \end{bmatrix} \quad (37)$$

Pour éviter un écrasement dans la fonction de perte des plantes peu présentes, nous opérons à une régularisation sur la fonction de coût, les classes les moins présentes pèsent beaucoup plus lourd, son score n'est plus basé sur le nombre de points correctement classifié, mais sur le nombre de classes correctement classifiées. Classifier correctement les classes les plus présentes minimisera de façon équivalente que la classification des classes les moins présentes, le classifieur se concentrera sur être performant dans chacune des classes.

Mis en équation, on aura avec les facteurs de régularisation $R_{plantes}$ et R_{sol} :

$$\text{Loss} = \text{Loss}_{plantes} * R_{plantes} + \text{Loss}_{sol} * R_{sol} \quad (38)$$

Ce qui se traduira dans la fonction de perte par :

$$\text{SegmentationCrossEntropyCorrected}(Yp, Ye) = - \sum_{i=0}^{Np} \sum_{c=0}^{Nc} Ye_{i,c} * \log(Yp_{i,c}) * \text{Regul}_c \quad (39)$$

La régularisation est quant à elle défini par la fonction inverse du nombre de classe présent dans le nuage de points :

$$\text{Regul}_c = \frac{1}{\sum_{i=0}^{Np} Ye_{i,c}} \quad (40)$$

Si on applique cette régularisation à la somme de tous les points d'une même classe, les classes sont équivalentes les unes aux autres

$$\sum_{i=0}^{Np} Ye_{i,0} * \text{Regul}_0 = \sum_{i=0}^{Np} Ye_{i,1} * \text{Regul}_1 \quad (41)$$

En développant le terme de régularisation, on retombe exactement sur l'égalité des sommes de classes régularisées :

$$\sum_{i=0}^{Np} Ye_{i,0} * \frac{1}{\sum_{i=0}^{Np} Ye_{i,0}} = \sum_{i=0}^{Np} Ye_{i,1} * \frac{1}{\sum_{i=0}^{Np} Ye_{i,1}} = 1 \quad (42)$$

D'un point de vue graphique, nous pouvons placer les proportions en fonctions du poids accordés à leurs individus, moins une classe sera présente, plus ses individus seront importants dans la fonction de perte (figure 75).

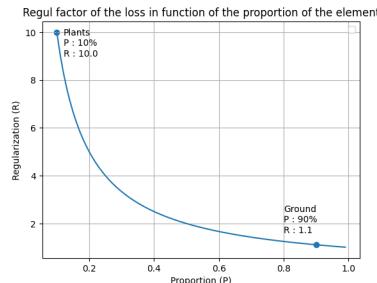


Figure 75: Correction proportionnelle de la fonction de perte de la classification, dorénavant, les classes se valent toutes, peu importe le nombre d'individus présents parmi elles

En voici le résultat de la classification :

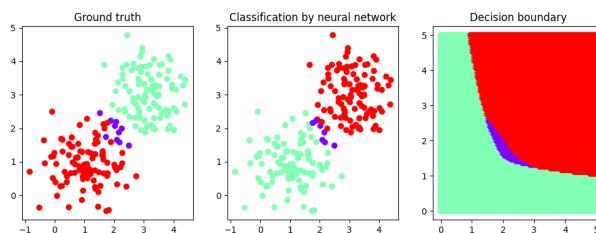


Figure 76: Classification en 3 classes avec une CrossEntropy corrigé

Cette méthode va permettre de focus l'attention du réseau de neurones sur les petites plantes, d'essayer d'un maximum les repérés pour ne pas les confondre avec le sol (figure 77).

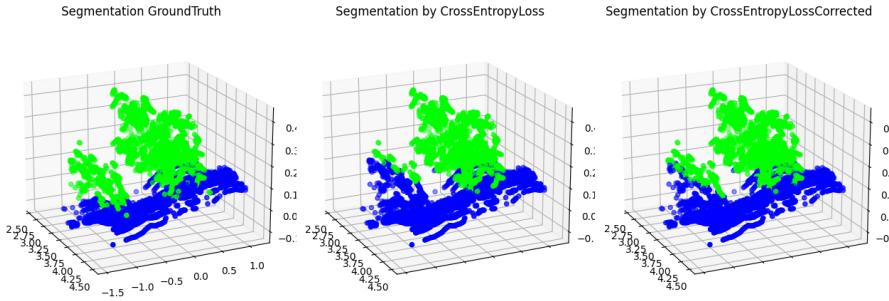


Figure 77: Comparaison des fonctions de perte de segmentation, meilleurs résultats parmi la fonction de perte corrigée au niveau de la discrimination sol-plantes

On a permis à notre classifieur de mieux se focus sur les plantes à détecter, mais malheureusement, on a pu observer en expérimentation un effet contraire, dorénavant, le réseau de neurones à tendance à faire beaucoup de fausses détections de sol en tant que plantes. Cette solution a malheureusement un contre-coût, si jamais il y a très peu de plantes, elles vaudront énormément par rapport au sol, et par conséquent, dès qu'il y aura une zone d'ambiguïté, le réseau de neurones préférera de faire des fausses détections en tant que plantes plutôt que de classifier en tant que sol et de "perdre" beaucoup plus de points.

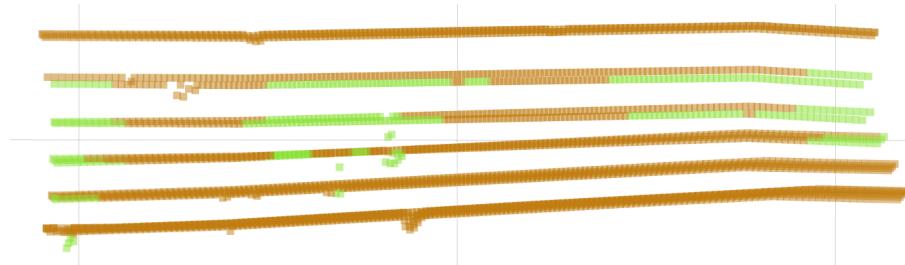


Figure 78: Erreur de segmentation, sol classifié en tant que plantes à la moindre anomalie de celui-ci

Pour remédier à cela, nous opterons pour un régularisation linéaire, les plantes vaudront toujours plus que le sol, mais ça n'explosera pas en terme de proportion. De plus, par cette méthode, si jamais les plantes se retrouvent en majorité, la correction sera toujours effective mais pour l'autre classe, ce qui permet de garder une cohérence dans les différentes configurations de plantes.

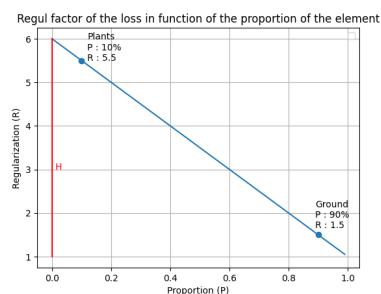


Figure 79: Correction linéaire de la fonction de perte de la classification,

Cette solution sera la meilleure, alliant équité de classification, robustesse et cohérence. Malheureuse-

ment, malgré cette amélioration, le réseau de neurones reste sensible aux zones ambigu, quand il y a un peu de bruit, d'occlusion, de vibration au niveau du robot... On peut en conclure que, une fonction de perte adaptée peut permettre de guider notre réseau de neurones à privilégier un comportement en particulier, mais ne peut malheureusement pas améliorer la compréhension des données, on peut changer uniquement le type de décision par rapport celles-ci.

Outil de visualisation et d'évaluation de la classification

Lors de ce projet, nous avons peu parlé des outils mis en place pour permettre d'évaluer les performances de nos algorithmes, nous nous sommes appuyé sur la visualisation d'un seul nuage de point pour exprimer les défauts de nos algorithme. Le test nuage par nuage permet de visuellement repérer les erreurs de nos classifieurs, mais c'est en réalité une technique peu représentative du résultat de classification, il faut pouvoir avoir une visualisation de l'ensemble des nuages de points formant la base de données. Pour se faire, nous avons utilisé les matrices de confusion, permettant de savoir à la fois la proportion d'erreur, mais aussi ce qui cause ces erreurs (faux-positif ou bien vrai-négatif).

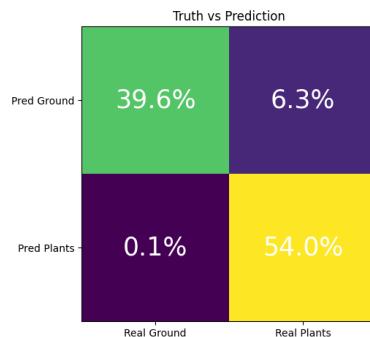


Figure 80: Matrice de confusion pour la classification d'un seul nuage de point

Comme décris précédemment, la visualisation d'un seul nuage de point pour visualiser le résultat de notre classification, on en fait la moyenne et comparons les différents classifieurs sur la même base de donné :

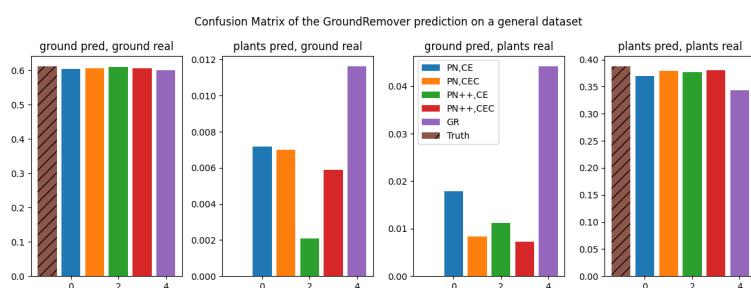


Figure 81: Comparaison des moyennes de classifieur

Cette métrique n'est malheureusement encore pas suffisante, il ne suffit pas qu'un classifieur soit meilleur qu'un autre en moyenne pour affirmer qu'il est meilleur. Il se peut, par exemple, que le classifieur A soit juste à 80% dans 100% des situations, le classifieur B est juste à 100% dans 90% des situations. Si on regarde uniquement la moyenne, le classifieur B sera alors considéré comme meilleur que le A alors qu'il se trompe complètement dans 10% des situations, alors que le A est lui globalement moins précis, mais aussi plus robuste, il se trompe un peu mais tous le temps pareil.

De plus, on ne peut évaluer les classificateurs uniquement dans des réalités simulés, on a évalué la technique basée géométrie du *ground removal* noté GR pour estimé son résultat sur les base de données, il apparaît moins performant que les autres classificateurs sur la base de données simulé, mais il a l'avantage de pouvoir s'adapter aux situations réelles. Pour permettre d'y voire plus claire, projettons le résultat de tous les nuages de points sur une matrice de confusion exprimée sous la forme d'un histogrammes, ce qui permet d'avoir une vue d'ensemble pour chacun des classificateurs :

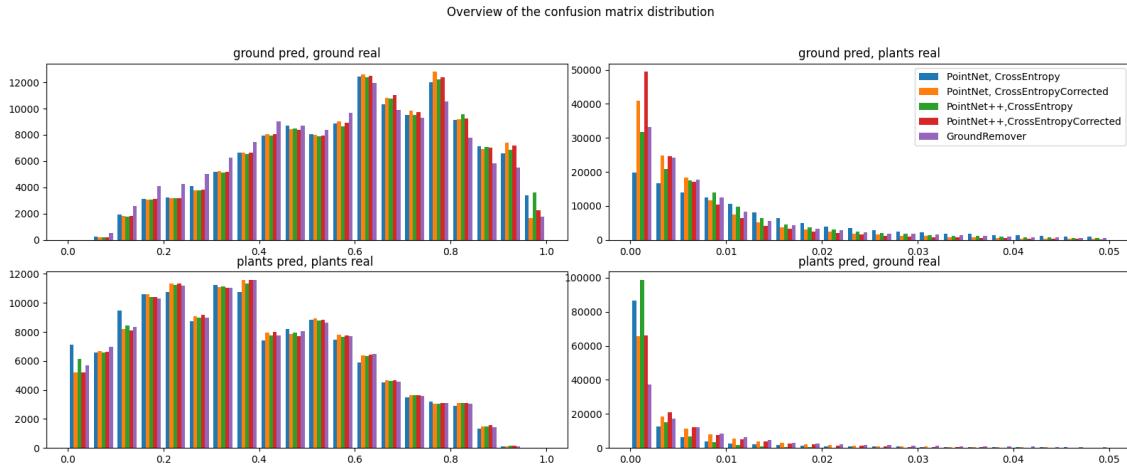


Figure 82: Résultat générale des différentes architectures sur un dataset général et un dataset regroupant le cas des petites plantes

On comprend mieux alors le choix de montrer les performances de chaque classifieur au moyen d'un unique nuage de points. L'analyse exhaustive et approfondie des résultats propres à chaque classifieur représente en soi une branche du domaine de l'intelligence artificielle à part entière. Acquérir et représenter visuellement les variables pertinentes pour discerner entre les divers classificateurs de manière précise qui est le meilleur peut se révéler une entreprise complexe.

Citations annexe

- [17] Anurag A. and Shuai Z. Conditional random fields meet deep neural networks for semantic segmentation. <<https://doi.org/10.1109/MSP.2017.2762355>>, 2018.
- [18] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Niebner. 3d-mpa: Multi proposal aggregation for 3d semantic instance segmentation. <<https://doi.org/10.48550/arXiv.2003.13867>>, 2020.
- [19] Yan Tian and Yujie Zhang. 3d tooth instance segmentation learning objectness and affinity in point cloud. <<https://dl.acm.org/doi/10.1145/3504033>>, 2022.
- [20] Xingqian X., Mang Tik C., Thomas S. H., and Honghui S. Deep affinity net: Instance segmentation via affinity. <<https://doi.org/10.48550/arXiv.2003.06849>>, 2020.
- [21] Minh Ôn Vu Ngoc and Yizi Chen. Introducing the boundary-aware loss for deep image segmentation. <<https://www.bmvc2021-virtualconference.com/assets/papers/1546.pdf>>, 2021.