

UNIVERSITÉ DE RENNES 1

MASTER 2 CALCUL SCIENTIFIQUE ET MODÉLISATION
RAPPORT DE STAGE

Etude et développement d'outils mathématiques pour estimer, en temps réel, le tassage et le volume d'un silo de maïs à partir de capteurs embarqués

Auteur :
Sébastien HERVIEU

Tuteur de Stage :
Geoffroy ETAIX

Tuteur Universitaire :
Fabrice MAHÉ

27 août 2018



Todo list

■ COMPLETER Compléter les remerciements	1
■ MISE EN FORME : Aérer	1
■ AMELIORER : Ce § peut servir à l'intro sur Tellus.	5
■ VERIFIER est-ce le bon terme ?	5
■ REDIGER Expertise Tellus Traitement des données	7
■ REDIGER Activité R&D	7
■ REDIGER Context Projet	7
■ REDIGER Symeter V2 : objectifs	8
Figure : Architecture capteurs → flux de données.	9
■ REDIGER Décrire les caractéristiques exactes	11
■ REDIGER données générées par un IMU	13
■ OBJECTIF : Introduire les concepts, indiquer qu'ils seront approfondis dans la suite du document.	15
■ REDIGER introduire les filtres de Kalman	16
Figure : chaîne de traitement de fusion de données	16
■ AJOUTER REF Filter de Kalman pour la fusion de données.	16
Figure : Architecture de traitement des nuages de points	16
Figure : Pose 3D d'un objet	22
■ AJOUTER REF Quaternion et quaternions unité	22
■ REDIGER précision d'évolution d'une plateforme robotique dépend de la prise en compte de la position relative de ses différents capteurs et actionneurs	22
■ REDIGER un petit laïus sur les capteurs supportés	22
■ TROUVER REF EXTERNE : SDF	23
■ REDIGER URDF et SDF, world, contrôleurs	23
■ AJOUTER REF le tuto de petit robot	24
Figure : Le petit robot	24
■ Ecrire type de tracteur	25
Figure : Image du tracteur modélisé	26
Figure : Tableau avec la correspondance entre vitesse roue et topic, angle direction et roue directionnelle	26
■ AJOUTER REF tableau avec les topics	29
■ COMPLETER Calcul des vitesses de rotation	29
■ Description de l'implémentation sous ROS	29
Figure : Schéma des flux topic qui permet de transformer une commande twist en 4 commandes de vitesses.	29
Figure : Représentation du vecteur d'état par rapport au chantier	31
■ REDIGER navsat transformation node	37
Figure : Architecture des nodes	37
■ REDIGER Les variables d'état influencées par IMU	37

■ REDIGER les variables d'état influencées par le GPS	37
■ AJOUTER REF figure	38
Figure : Chaîne de traitement LIDAR → Nuage de Point	38
Figure : Transformer un ensemble de lignes en un nuage de points cohérent	38
■ REDIGER LIDAR : ensemble d'angles, temps de vol, puissance reçue	39
■ REDIGER Nuage de points : positionnement x,y,z dans la scène	39
■ COMPLETER principe de fonctionnement des Voxels	39
■ REDIGER Il est nécessaire de connaître précisément la pose du LIDAR pour générer le nuage de points.	40
Figure : Illustration nature donnée LIDAR	40
Figure : Illustration nature donnée Nuage de Point	40
Figure : Diagramme fonctionnel LIDAR → Nuage de Point	40
■ REDIGER les structures de données de stockage de l'information 3D	41
■ TROUVER REF EXTERNE : les structures de données 3D	41
■ REDIGER Les b-trees, principes	41
■ REDIGER Point forts : calculs performants	41
■ REDIGER points faible : arbre équilibrés → difficile d'ajouter de nouveaux points.	41
■ REDIGER Octree, principes	41
■ REDIGER points forts : structure déséquilibrées sans problème, possibilité d'ajouter de nouveau points avec une bonne performance	41
■ REDIGER point faible : beaucoup d'overhead de memoire si pas implémenté correctement.	41
■ REDIGER Utilisation d'octomap car composant sur étagère	41
■ REDIGER permet d'implémenter rapidement la chaîne de traitement pour vé- rifier la validité de la faisabilité	41
Figure : Capture d'écran représentant le mur sous Gazebo.	42
Figure : Capture d'écran modèle d'ensilage sous blender + capture d'écran mur et ensilage dans Gazebo	42
Figure : Capture Gazebo du chantier avec tracteur et mur	42
■ REDIGER Une fois l'acquisition effectuer, on invoque le node de sauvegarde de nuage, qui le sauve dans un fichier	42
■ REDIGER Outil de visualisation utilisé : Paraview	43
■ REDIGER besoin de convertir pcd en vtk à l'aide de l'outil approprié	43
■ REDIGER permet de naviguer de manière efficace dans le nuage de point, d'effectuer des projections, etc, etc	43
■ REDIGER Le nuage est épais	43
■ REDIGER octomap génère un nuage de point basé sur le centre des voxel occupés → problème de quantification volumique implique perte de précision de la mesure	43
■ REDIGER Soit réduire la taille du voxel → augmentation de la taille	43
■ REDIGER Soit retourne le point moyen de chaque voxel → non supporté par octomap → à implémenter nous même.	43
■ AJOUTER REF octomap	43
■ REDIGER TellusCar avec Instrument montés sur l'arrière	44
■ REDIGER données GPS, données IMU et données LIDAR, toutes acquises en même temps	44
Figure : text	44

■ REDIGER Capture Bagfile alors que la TellusCar évolue dans le parking de adjacent à TellusEnv	44
■ REDIGER La partie logiciel de Symeter V2 peut exploiter des rejeux des captures pour estimer l'évolution de la TellusCar, et tenter de reconstituer le parking.	44
Figure : trajectoire sous Googlemap	45
Figure : graphe des accélérations et de l'assiette issues de l'IMU	45
Figure : données LIDAR brutes	45
■ REDIGER Il a été compliqué de faire fonctionner la partie localisation de Symeter V2 avec les données réelle	45
■ REDIGER grande sensibilité des données IMU si la pose de l'IMU n'est pas rigoureusement décrite dans la configuration de symeter V2	45
■ REDIGER Obligé de traiter les données IMU pour rétablir la bonne orientation	45
■ REDIGER L'imu mesure le cap avec 0 au nord alors que robot_localization s'attend à un 0 pour un cap à l'Est	45
■ REDIGER une fois le système de localisation OK, pb avec Octomap qui n'a pas l'air de fonctionner au dela d'une certaine distante	46
■ REDIGER Illustre les problématiques du passage de la simulation à un integration en grandeur	46
■ REDIGER IMU simulé n'est pas fiable : il faudra en trouver un autre	46
■ REDIGER A l'heure actuelle pas encore terminé	46
■ PEUT ETRE caméra video pour odométrie visuelle?	47
■ REDIGER Decrire le cycle de simulation	47
Figure : Diagramme d'état simulation de l'acte de tassage.	47

Remerciements

Je tiens à remercier les personnes qui m'ont permis de près ou de loin à accomplir ce stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements à Messieurs **Fabrice Mahé et Eric Darri-grand** professeurs de l'Université de Rennes 1, qui m'ont permis de suivre cette formation et qui m'ont accompagné lors de la recherche de stage.

Je tiens à remercier Monsieur **Geoffroy ETAIX**, qui m'a accordé sa confiance pour le stage et pour etc

Remercier Fongecif

Remercier IDApps, Yann, Stéphane, et surtout Laila !

Enfin je remercie mon épouse Céline et mes deux filles pour leur encouragements, leurs soutient et leur patience.

COMPLETER Completer les remerciements

MISE EN FORME : Aérer

Table des matières

Remerciements	1
1 Introduction	5
1.1 Tellus Environment - Missions Principales	5
1.1.1 Géophysique et cartographie haute-définition	5
1.1.2 Collecte des données géophysique	5
1.1.3 Traitement des données	7
1.2 Activité R&D	7
1.3 Projet Symeter V2	7
1.3.1 Symeter V1	7
1.3.2 Symeter V2 : Objectifs	8
2 Capteurs, Modélisation, Contraintes	9
2.1 Capteurs	9
2.1.1 LIDAR	9
2.1.1.1 Détecteur Actif :	9
2.1.1.2 Scan de l'environnement	10
2.1.1.3 LIDAR Hokuyo UTM-30LX-EW	10
2.1.2 IMU	12
2.1.2.1 Les grandeurs mesurées	12
2.1.2.2 IMU XSense MTi 28A53G35	13
2.1.3 GPS en mode RTK	13
2.1.3.1 GPS simple	13
2.1.3.2 Principe du mode RTK	14
2.2 Environnement de programmation : ROS	15
2.3 Modélisation	15
2.3.1 Traitement des nuages de points	16
2.4 Contraintes de développement	16
2.4.1 Capacités de tests en grandeur limitées	17
2.4.2 Plateformes de test disponibles	17
2.4.2.1 Environnement de simulation Gazebo	17
2.4.2.2 Prototype monté sur Camionnette	17
2.5 Les grandes phases du stage	18
3 Simulation d'un tracteur évoluant sur un chantier d'ensilage à l'aide de ROS/Gazebo	20
3.1 Présentation de ROS	20
3.1.1 Gestion des transformations	21

3.1.1.1	Poses	21
3.1.2	Gestions des Capteurs	22
3.2	Présentation de Gazebo	23
3.2.1	Construction d'un robot virtuel	23
3.2.2	Vérification de disponibilité des capteurs	24
3.3	Contraintes de mise en oeuvre	24
3.3.1	Pas d'adhérence au démarrage de la simulation	24
3.3.2	Simulation mécanique, frottements, adhérence	25
3.3.3	Conclusions sur les contraintes	25
3.4	Mise en Oeuvre : simulation d'un environnement de tassage de silo	25
3.4.1	Montage d'un tracteur simulé	25
3.4.1.1	Description Physique	25
3.4.1.1.1	Chassis	25
3.4.1.1.2	Actuateurs et Contrôleurs	26
3.4.1.2	Propulsion et Guidage	26
3.4.1.2.1	Algorithme	26
3.4.1.2.2	Implémentation sous ROS	28
4	Mise en place du processus de localisation	30
4.1	Présentation du problème	30
4.1.1	Repères	31
4.1.1.1	Repère Inertiel	32
4.1.1.2	Repère ECEF	32
4.1.1.3	Repère de navigation local	32
4.1.1.4	Repère du véhicule	32
4.1.2	Système de Navigation Inertielle	32
4.1.3	Transformation des coordonnées géocentriques en coordonnées locale	32
4.2	Filtres de Kalman	32
4.2.1	Filtres de Kalman Linéaires	32
4.2.1.1	Discrétisation	33
4.2.1.2	Boucle de Kalman	34
4.2.1.3	Fonctionnement conceptuel du Filtre de Kalman	35
4.2.2	Estimation de Pose 3D : Filtres de Kalman Etendus	35
4.3	Mise en oeuvre sous ROS.	36
4.3.1	Module robot_localisation	36
4.3.2	navsat_transformation_node	37
4.3.3	Configuration des Capteurs	37
4.3.4	Quelques tests	37
5	Exploitation des données LIDAR	38
5.1	Présentation de la chaine de traitement des données LIDAR	38
5.2	Acquisition et mise en forme des données LIDAR	39
5.2.1	Transformation trame LIDAR en un nuage de points	39
5.2.2	Filtrage de la ligne de point par downsampling	39
5.3	Accumulation des nuages de points	40
5.3.1	Spécifications pour le point_cloud_aggregator	40
5.3.2	Principe de stockage des données 3D	41
5.3.2.1	Les B-Trees	41

5.3.2.2	Les Octrees	41
5.3.2.3	Le choix : octree correspond à notre besoin.	41
5.3.3	Mise en oeuvre : octomap	41
5.4	Mise en oeuvre sous Gazebo	41
5.4.1	Modélisation d'un chantier d'ensilage	42
5.4.2	Test sous gazebo	42
5.4.3	Analyse du nuage de point généré	43
5.4.3.1	Outil pour l'analyse de nuage de points : Paraview . . .	43
5.4.3.2	Points à améliorer sur le nuage de points	43
6	Mise en oeuvre à partir de mesures réelles	44
6.1	Protocole de test	44
6.1.1	Instruments d'acquisition)	44
6.1.2	Exploitation	44
6.2	Données générée - visualisation sous google maps	45
6.3	Exploitation des données	45
6.3.1	Configuration : description des poses des instruments par rapport au repère du véhicule	45
6.3.2	Intégration des données GPS et IMU pour localisation	45
6.3.3	Reconstitution du parking	46
6.4	Récapitulation	46
7	Reste à faire et Axes Améliorations	47
7.1	Difficulté de Simulation de l'action de tassage	47
8	Conclusion	48
A	Représentation de l'Attitude et des Rotations	49
A.1	Angles d'Euler	49
A.2	Quaternions Unitaires	50
	Bibliographie	52

Chapitre 1

Introduction

J'effectue depuis le 3 avril 2018 mon stage de fin d'étude de Master 2 CSM au sein de Tellus Environnement, afin de m'occuper de la partie étude et développement du projet appelé en interne "Symeter V2".

Cette introduction présente les activités de Tellus Environnement et introduit le contexte du projet Symeter V2.

1.1 Tellus Environment - Missions Principales

1.1.1 Géophysique et cartographie haute-définition

Tellus Environment a été créée en 2012 et s'est initialement spécialisée dans le traitement de données géophysiques, la cartographie haute définition des sous-sols et des fond-marins. Elle propose à ses clients une offre bout en bout d'acquisition et de traitement des données pour permettre à ses clients d'agir en fonction de ces conclusions.

1.1.2 Collecte des données géophysique

Tellus Environment met en oeuvre ses propres équipements géoradar et magnétomètre pour l'acquisition de données sur le sous-sol sur des surfaces de quelques mètres-carré à quelque hectares. Elle est aussi en mesure de planifier et coordonner la mise en oeuvre d'équipements plus lourds - LIDAR, géoradar, magnétomètres aéroporté - pour obtenir des données sur des surfaces beaucoup plus importantes, de l'ordre du kilomètre carré.

Tellus Environment peut aussi coordonner la mise en oeuvre d'équipements d'acquisition marins - sonar, multibeam, percuteur - pour permettre la cartographie des fonds marins et des sous-sols aquatiques, que ce soit en environnement eau douce - rivières, lacs, étangs - ou marins.

Tellus Environment a accès à de nombreuses bases cartographiques pour compléter les données acquises sous sa supervision pour aider à la mise en oeuvre des équipements ainsi qu'à compléter les données acquises en vue de leur traitement.

AMELIORER
Ce § peut s'améliorer
l'intro sur 7

VERIFIER
le bon terme



FIGURE 1.1 – Géoradar ZOND



FIGURE 1.2 – Sonar (en cours d'utilisation)

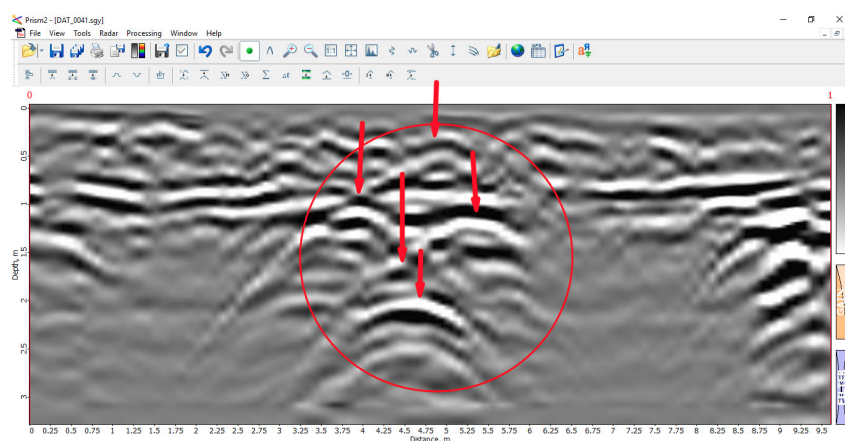


FIGURE 1.3 – Exemple d'un radargramme d'une détection de réseaux enterrés. (Image Tellus Environment)

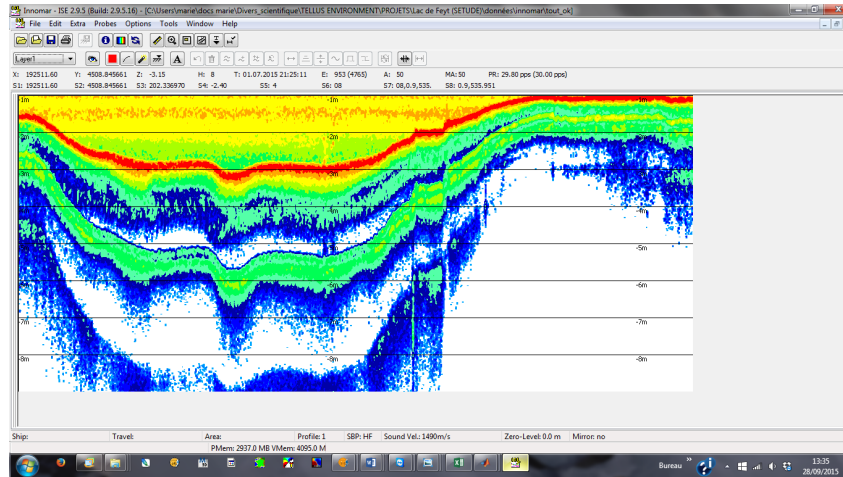


FIGURE 1.4 – Exemple d’un sonagramme généré par un sonar sub-bottom (Image Tellus Environment)

1.1.3 Traitement des données

REDIGER Expertise Tellus Traitement des données

1.2 Activité R&D

REDIGER Activité R&D

Tellus Environment met aussi en oeuvre une activité R&D qui lui permet de développer des composants mettant en oeuvre les expertises de Tellus Environment pour créer des produits innovants. Les objectifs sont d’augmenter la productivité de ses activités services, ou alors répondre à des besoins exprimés par des partenaires sur des projets spécifiques.

1.3 Projet Symeter V2

Je prends part depuis le début de mon stage au projet Symeter V2, qui vise à améliorer un produit existant, Symeter V1.

REDIGER Contexte Projet

1.3.1 Symeter V1

Symeter V1 est un équipement qui permet de mesurer en temps réel l’état de tassage d’un silo de fourrage (maïs, herbe, ...) en vue d’augmenter la qualité des silos générés. Il est basé sur un équipement de mesure Laser, le LIDAR, qui permet de collecter très rapidement de nombreux points de mesure de position d’objets.

Ce LIDAR est placé à l’aplomb du silo au sommet d’une perche de 7m de hauteur, stabilisée par un trepied.

Ce produit avait été mis en oeuvre dans le courant de l'année 2017, avait démontré que la mesure de tassage était possible et précise, mais avait mis en évidence des points d'amélioration.

1.3.2 Symeter V2 : Objectifs

REDIGER Symeter V2 : objectifs

Symeter V2 : Objectifs Symeter V2 : Présentation du plan de projet - test du simulateur Gazebo pour évaluer son utilité dans le projet Symeter2 - Simulation couverture lidar - Montage des outils nécessaires au développement simulé du projet symeter - Mise en place de la localisation : installation et tests - Mise en place de l'acquisition des relevés.

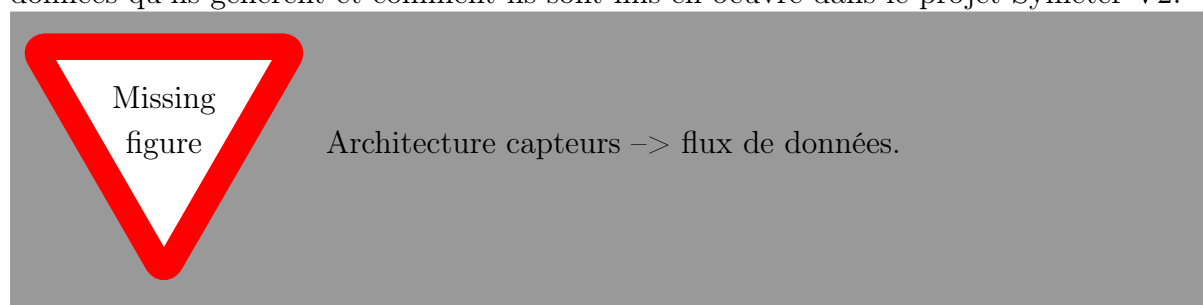
Chapitre 2

Capteurs, Modélisation, Contraintes

Ce chapitre présente dans leurs grandes lignes les capteurs cibles du projet Symeter V2, les outils de modélisation qui seront utilisés pour les mettre en oeuvre et les contraintes qui en découlent. Les grandes étapes du projet sont enfin élaborées à la lumière de ces informations.

2.1 Capteurs

Les capteurs à mettre en oeuvre seront un ou plusieurs LIDAR, un ou plusieurs IMU et un ou plusieurs GPS. Cette section décrit les fonctionnalités de ces capteurs et les données qu'ils génèrent et comment ils sont mis en oeuvre dans le projet Symeter V2.



2.1.1 LIDAR

Un LIDAR est un équipement qui permet de prendre de nombreuses mesures de l'environnement à partir d'un laser. En exploitant les informations d'angle de pointage et en observant les caractéristiques du signal réfléchi tant du point de vue temporel (temps de vol, phase) que du point de vue de la forme du signal (puissance réfléchie, déphasage, distorsion, effet Doppler, ...), il est possible de mesurer de nombreuses caractéristiques de l'environnement.

Le LIDAR est utilisé par Symeter V2 pour détecter la forme précise du chantier d'ensilage.

2.1.1.1 Détecteur Actif :

Le LIDAR est essentiellement un détecteur actif dont le principe est identique à celui du radar : une pulse électromagnétique est émise dans une direction privilégiée par

une partie émettrice, constituant un signal incident.

Ce signal incident est réfléchi par un objet de l'environnement, générant un signal réfléchi qui retourne en direction du LIDAR. Ce signal réfléchi est reçu par un détecteur.

En comparant le signal émis et le signal reçu nous pouvons en déduire certaines caractéristiques de l'environnement.

Différents types de mesure : Il existe deux grands types de LIDAR :

LIDAR à "temps de vol" : les lidars de ce type déterminent la distance de la cible en mesurant le temps écoulé entre le moment de l'émission du pulse et la réception du signal réfléchi par la cible.

LIDAR "full waveform" : les LIDAR de ce type émettent un signal périodique continu et mesurent la distance de l'objet détecté essentiellement par mesure de déphasage du signal réfléchi par rapport au signal émis. Les modifications de la forme du signal réfléchi et de son intensité permettent d'extraire d'autres informations à propos de la cible.

2.1.1.2 Scan de l'environnement

En faisant varier la direction du bloc émetteur/récepteur du LIDAR, il est possible de faire plusieurs mesures de l'environnement. Le LIDAR mettant en oeuvre des procédés optiques, il est possible de faire varier très rapidement son pointage et ainsi d'obtenir de très nombreux relevés de l'environnement.

Deux grands modes de scan de l'environnement : Il existe deux grands modes de scan de l'environnement pour un LIDAR

Mode 2D ou "planar" : Dans ce mode le LIDAR mesure l'environnement dans un plan unique, généralement en faisant tourner le faisceau de manière circulaire autour d'un axe, comme illustré figure 2.1,

Mode 3D : Dans ce mode le LIDAR mesure des portions d'espace, généralement en combinant deux rotations simulées du faisceau, soit en faisant tourner sur un même axe de révolution plusieurs faisceaux ayant des inclinaisons différentes (voir).

2.1.1.3 LIDAR Hokuyo UTM-30LX-EW

Le projet Symeter V2 met en oeuvre un LIDAR de type Hokuyo UTM-30-LX-EW (figure), le même que celui utilisé par Symeter V1. C'est un équipement robuste et précis qui est de plus bien connu par Tellus Environment.

Cet équipement est composé d'un laser infrarouge (longueur d'onde de 905nm) pour scanner un champ semi-circulaire de 270°. C'est un LIDAR de type "planar, temps de vol". Il mesure les distances des objets à sa portée par pas de 0,25°. La distance de détection maximale est de 30m.

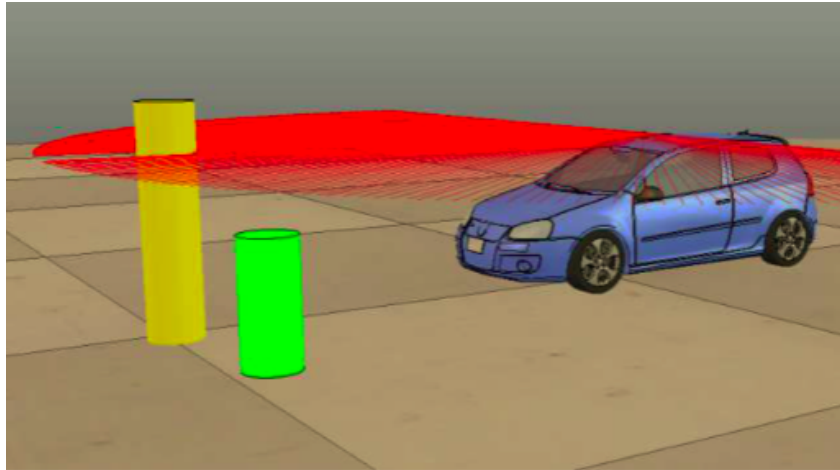


FIGURE 2.1 – Plan de détection d'un lidar 2D ([YSA⁺13])

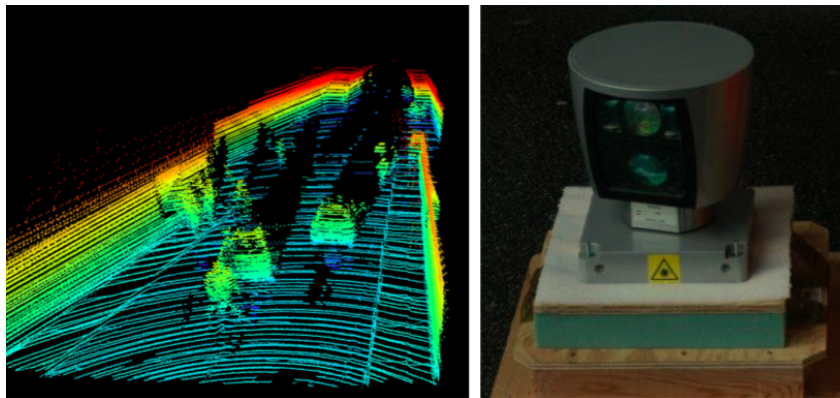


FIGURE 2.2 – Scan laser 3D (gauche) acquis au moyen d'un LIDAR HDL-64E de Velodyne (droite) - Image [Ste14]

Angle de Balayage	270°
Résolution Angulaire	env. 0,25°
Temps de Balayage	25ms/balayage (40 balayages par seconde)
Distance de détection	Portée garantie : 0,1 à 30m
Résolution de la mesure	1mm σ



FIGURE 2.3 – LIDAR Hokuyo UTM-30LX-EW (roboshop.com)

2.1.2 IMU

Un IMU - Inertial Measurement Unit - est un équipement comportant plusieurs capteurs inertiels, d'accélération et d'angles qui permettent avec un traitement des données approprié d'établir de suivre la pose du véhicule sur lequel il est monté.

2.1.2.1 Les grandeurs mesurées

Un IMU comporte généralement 3 accéléromètres linéaires et un gyroscope, et souvent un magnétomètre.

Le **gyroscope** permet de mesurer, avec précision et en continu, l'inclinaison du véhicule en terme de **roulis**, **tangage** et **lacet**.

Les **accéléromètres** mesurent en continu les accélérations linéaires de la partie du véhicule sur lequel l'IMU est fixé, dans trois directions différentes. Ces directions sont généralement dénotées x, y, z , par analogie avec le repère cartésien. Simplement, les accélérations sont mesurées dans le repère du véhicule, mobile et variable et non dans un repère absolu immobile.

En combinant les variations d'inclinaison et les mesures d'accélérations linéaires dans le repère mobile du véhicule, il est possible de reconstruire avec une certaine certitude la trajectoire du véhicule dans le repère absolu, en utilisant les équations de dynamique Newtonnienne (fig 2.4).

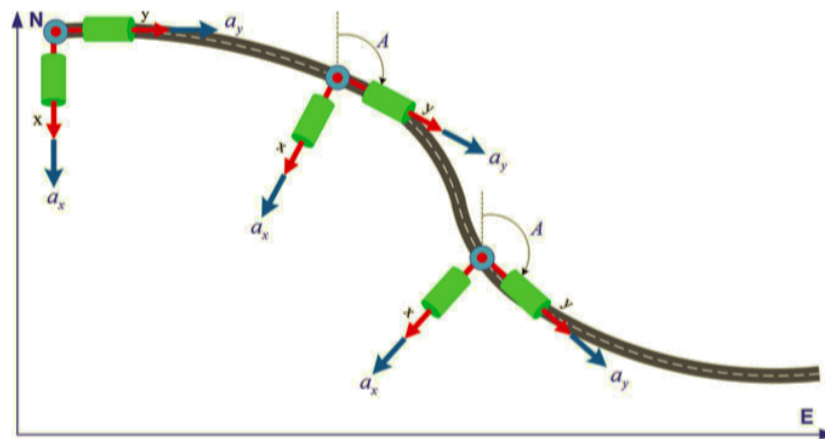


FIGURE 2.4 – Principe de la Navigation Inertielle

Comme indiqué ci-dessus, un IMU possède souvent un **magnétomètre**. Celui-ci permet de mesurer la direction et l'intensité du champ magnétique terrestre par rapport à l'IMU et permet d'en dériver le cap, la direction vers lequel le véhicule pointe sur la surface de la Terre.

Ce permet, en démarrant l'IMU alors que le véhicule est parfaitement immobile, d'obtenir les conditions initiales du vecteur d'état du véhicule, par rapport au repère odométrique.

Par ailleurs, reconstituer une position à partir d'informations d'accélération implique mathématiquement de procéder à une double intégration numérique. Ce procédé est donc sujet à une dérive dans le temps et doit donc être recalé par une mesure de pose issue d'un autre capteur.

Par contre, un IMU peut fournir ses informations avec un très haut taux de rafraîchissement, sans nécessiter d'information extérieure au véhicule. Il peut donc se révéler indispensable lorsqu'aucune autre nouvelle information sur la position du véhicule n'est disponible sur des périodes de temps plus ou moins longue.

2.1.2.2 IMU XSense MTi 28A53G35

REDIGER données générées par un IMU



FIGURE 2.5 – IMU Xsens MTi-28A53G35

2.1.3 GPS en mode RTK

Le système Symeter V2 ayant besoin d'une précision centimétrique pour permettre la mesure du tas d'ensilage avec précision du même ordre de grandeur, il mettra en oeuvre d'un GPS en mode RTK. Ce mode fonctionnement particulier du système GPS est en mesure de fournir position centimétrique du véhicule relativement à une borne GPS (base) situé à proximité du chantier.

2.1.3.1 GPS simple

Pour expliquer le principe du GPS en mode RTK, il est nécessaire de comprendre comment le système GPS "simple" fonctionne.

Le système GPS est un système de positionnement par satellite qui, en se basant sur les signaux émis par une constellation de satellites, est en mesure de calculer la position sur la surface de la Terre d'un récepteur avec une précision de l'ordre de la dizaine de mètres, pour une utilisation courante.

Le principe de base est celui de la triangulation : les satellites GPS émettent en permanence un signal comportant une horloge (temps GPS). En connaissant les éphémérides des satellites et en écoutant les signaux de plusieurs de ces satellites, notamment leur signal d'horloge respectif, un récepteur GPS peut calculer avec une grande précision la position de chacun des satellites émetteurs dans le référentiel géocentrique.

Une fois le positionnement des satellites établi, une simple triangulation permet de calculer avec une grande précision du récepteur sur la surface de la Terre (voir 2.6).

Pour obtenir une précision suffisante, un récepteur doit écouter au moins 4 satellites en même temps.

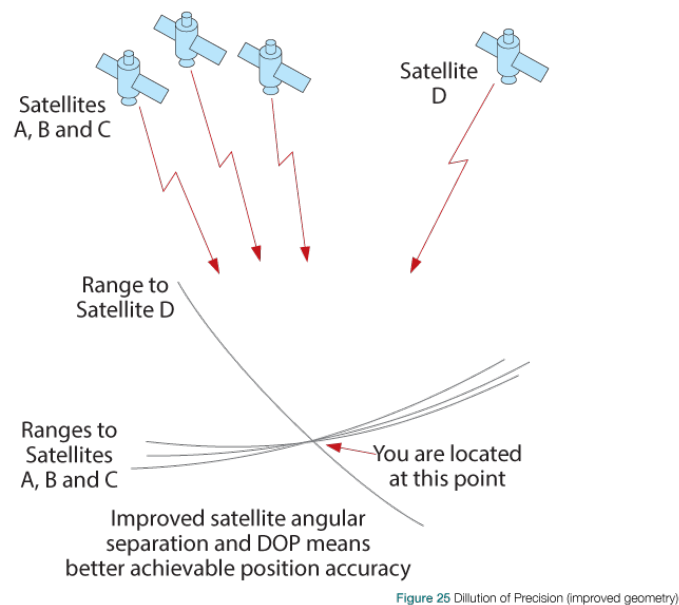


FIGURE 2.6 – Principe de fonctionnement du GPS simple (image [Jef10])

2.1.3.2 Principe du mode RTK

Le mode RTK du système GPS est un mode de fonctionnement qui permet de mesurer avec une très grande précision la position relative de deux récepteurs GPS distincts. L'un, dénommé **la base** est un récepteur fixe, dont la position GPS est connue avec une grande précision. L'autre dénommé **le rover** est en communication constante avec la base au moyen d'une liaison radio adaptée (voir 2.7)

Les deux récepteurs étant présents sur le même site à la surface de la Terre, ils seront à l'écoute tous les deux mêmes satellites. En comparant les signaux reçus d'un même satellite par la base et le rover, le rover est en mesure de calculer avec une très grande précision - de l'ordre du centimètre - la différence de distance entre la base et le satellite d'une part, et le rover et le satellite de l'autre part.

En combinant ces différences de distance à partir de plusieurs satellites, le rover est donc capable de calculer très précisément sa position relativement à la base.

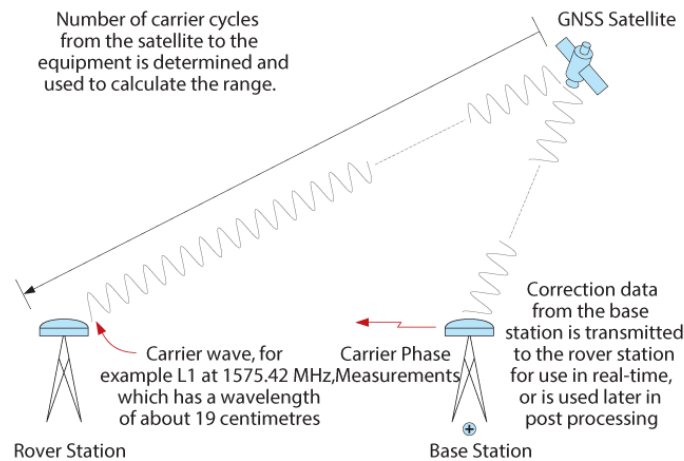


Figure 42 Real-Time Kinematic

FIGURE 2.7 – Principe de fonctionnement GPS RTK (image [Jef10])

2.2 Environnement de programmation : ROS

Du fait de son utilisation dans la première version de Symeter, et du fait de ses qualités en terme de modularité, de disponibilité des drivers pour les capteurs et actionneurs, l'environnement logiciel ROS a été choisi avant même le début du stage pour être la base de la partie logicielle de Symeter V2.

2.3 Modélisation

OBJECTIF : Introduire les concepts, indiquer qu'ils seront approfondis dans la suite du document.

Comme indiqué en introduction, le système Symeter V2 sera une plateforme semi-autonome, qui devra être en mesure de fournir une aide à la décision à partir de mesures effectuées depuis une plateforme mobile, et ceux avec une intervention humaine minimale.

Symeter V2 devra donc estimer en temps réel sa pose, afin de pouvoir intégrer une représentation 3D du chantier afin de pouvoir effectuer son service.

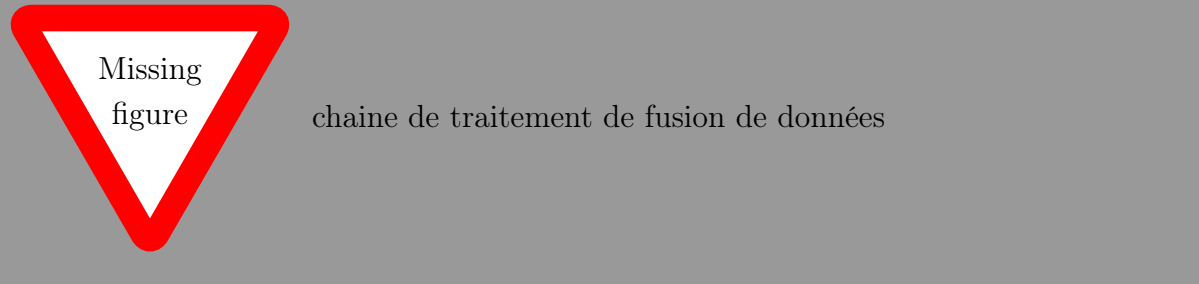
Les outils de modélisation qui devront être mis en oeuvre sont essentiellement ceux mis en oeuvre dans les systèmes robotiques :

- Les *poses* et les *transformation* : La pose des capteurs et des éléments mécaniques du véhicule les uns par rapport aux autres, et celle du véhicule par rapport à son environnement.
- L'estimation de paramètres à partir d'un ensemble de mesures issues de différents capteurs, au moyen de la *fusion de données*

— Reconstruction d’une scène 3D à partir de *nuages de points*.

Ces outils sont décrits plus en détail dans les chapitres suivants.

REDIGER introduire les filtres de Kalman



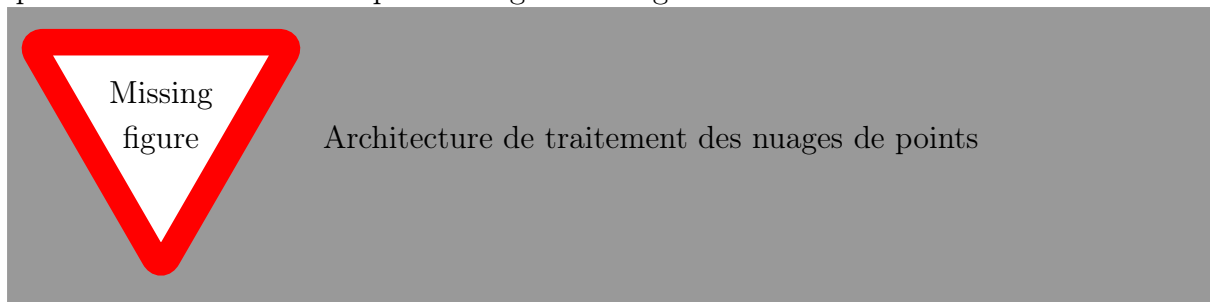
2.3.1 Traitement des nuages de points

Le LIDAR utilisé par le système Symeter V2 étant de type "2D planar", les points retournés lors d’une unique mesure feront tous partie du même plan. L’ensemble des mesures sera donc constitué d’un ensemble de tranches qu’il faudra reconstituer à l’aide d’un traitement adapté.

Le traitement des données LIDAR comportera les étapes suivantes :

1. Filtrage des données pour une obtenir une densité de points de mesure qui soient suffisamment espacés
2. Conversion des données LIDAR (direction / distance) en un nuage de points (liste de point P_n avec chacun des coordonnées x_n, y_n, z_n)
3. Stockage de ces points dans une structure de données capable d’accumuler les nouveaux points mesurés au cours du temps, et constituer une carte 3D d’occupation.

La plupart de ces traitements seront implémentés en utilisant le Point Cloud Library qui offre de nombreux composants logiciels intégrés directement dans ROS.



2.4 Contraintes de développement

Le système Symeter V2 st destiné à évoluer sur un véhicule de type tracteur agricole. Il est de plus destiné à construire une représentation 3D d’un chantier d’ensilage, partir de positions qui vont varier non seulement en x et en y , mais aussi et surtout en z .

AJOUTER
Filter de K
pour la fusi
données.

2.4.1 Capacités de tests en grandeur limitées

Pour développer et tester le système Symeter V2, le développeur ne disposera pas de tracteur en grandeur réelle, ni de chantier d'ensilage, ceux-ci étant des chantiers annuels se déroulant à des moments bien défini dans l'année (printemps et automne).

Par ailleurs, au début du stage il n'est pas encore tout à fait décidé de combien de LIDAR seront nécessaires pour que le système puisse assurer une couverture complète depuis des équipements embarqués sur le tracteur.

Les équipements à mettre en oeuvre sont relativement couteux et leur mise en oeuvre requiert une certaine expertise. Il est donc nécessaire de pouvoir mettre en oeuvre le développement du système par le biais de mises en oeuvre alternatives.

Ces possibilités sont au nombre de 2 : la simulation et la mise en oeuvre en grandeur des équipements sur une camionnette pour effectuer des tests simples de reconstitution du terrain.

2.4.2 Plateformes de test disponibles

2.4.2.1 Environnement de simulation Gazebo

Gazebo est un environnement de simulation très complet, qui permet de simuler un environnement physique dans lequel une plateforme semi-robotique simulée peut évoluer.

Cette plateforme simulée peut mettre en oeuvre un grand nombre de capteurs et actionneurs virtuel et ainsi offrir une grande fidélité dans la mise en oeuvre logicielle de la plateforme robotique.

Gazebo est un logiciel intégré à ROS. Sa mise en oeuvre requiert de programmer un véhicule virtuel similaire à la plateforme cible du système (dans notre cas un tracteur avec les capteurs listés dans les sections précédentes), puis de déployer le logiciel ROS pour Symeter V2 sur ce véhicule simulé.

Il est aussi possible de spécifier le "monde" dans lequel le tracteur simulé évoluera, en y ajoutant des murs, du relief, voire une butte en guise de tas d'ensilage.

Il est donc envisageable d'utiliser Gazebo pour simuler un chantier d'ensilage pour tester la localisation, l'acquisition du terrain, la mesure d'un modèle de Silo.

2.4.2.2 Prototype monté sur Camionnette

Une simulation n'étant jamais parfaite (comme nous le verrons par la suite), il est quand même nécessaire d'effectuer des certains tests avec des instruments réels, lors de tests en grandeur.

Pour la mise en oeuvre en grandeur, un prototype de l'équipement a été monté sur la camionnette de Tellus Environnement, la "Tellus Car".

Le prototype comporte :

- Une unité de calcul
- Un GPS en mode RTK
- Un LIDAR de type Hkuyo UTM-30LX-EW
- Un IMU de type XSense

Ce prototype a été utilisé pour effectuer des captures des trois capteurs alors que la Tellus Car évoluait dans le parking adjacent aux locaux de Tellus Environnement.

2.5 Les grandes phases du stage

Le passage de Symeter V1, où le LIDAR est fixe par rapport au chantier, à Symeter V2 où le LIDAR est monté sur le tracteur et est donc mobile par rapport au chantier, remet tout en question vis-à-vis du travail effectué sur Symeter V1.

Le fait d'embarquer le système sur un tracteur pose une contrainte très forte sur le projet en forçant l'ajout d'IMU et d'un GPS qui devront être utilisés pour la localisation.

Donc d'un sujet V1 où nous avons une reconstitution du chantier par une accumulation simple de scans LIDAR rectangulaires, nous avons en V2 deux problèmes complexes interdépendants :

- Localisation à partir de fusion de données IMU et GPS
- Reconstitution du chantier à partir de scans aléatoires

Un problème supplémentaire étant l'absence d'une plateforme matérielle de test pour pouvoir tester les solutions de ces deux problèmes.

Comme il était apparent que sur tous ces sujets nous partions de zéros, il a été décidé très tôt dans le projet de monter une plateforme de tracteur simulée sous Gazebo pour effectuer le développement initial de Symeter V2. Cette plateforme simulée nous permettrait dans un premier temps d'effectuer des tests qualitatifs des solutions choisies, pour valider un certain niveau de faisabilité ainsi que faire des essais sur le choix du nombre de capteur et leur disposition.

A plus long terme, cette plateforme simulée nous permettra d'effectuer des tests quantitatifs de Symeter V2 lorsque de l'implémentation de la mesure de tassage sera complétée, et donc de valider le bon fonctionnement du système avant même de faire des tests sur des chantiers d'ensilage en grandeur.

Le déroulement du stage a donc été le suivant :

1. Vérification des capacités de Gazebo à simuler notre plateforme (capteurs, véhicule)
2. Montage de la plateforme Simulée tracteur et capteurs sous ROS / Gazebo
3. Montage du processus de localisation par fusion IMU + GPS

4. Montage de la chaine de traitement LIDAR pour reconstitution de la représentation 3D du chantier
5. Tests des éléments développés ci-dessus sur des captures issues du prototype monté sur la Tellus Car.

Les résultats de ces différentes parties du stage sont décrits dans les chapitres suivants de ce rapport.

Chapitre 3

Simulation d'un tracteur évoluant sur un chantier d'ensilage à l'aide de ROS/Gazebo

Comme indiqué dans le chapitre précédent, il a été très tôt dans le projet décidé d'utiliser l'environnement logiciel ROS pour implémenter le système Symeter V2. Il avait aussi été décidé de monter un véhicule simulé à l'aide de Gazebo afin d'effectuer le développement initial.

Ce chapitre présente en détail l'environnement logiciel ROS et le logiciel de simulation Gazebo. Cette présentation est illustrée par la construction d'un robot virtuel simple, pour lequel un retour d'expérience est effectué en fin de chapitre suite à cette première expérience.

L'implémentation du tracteur de test est ensuite présentée en fin de chapitre.

3.1 Présentation de ROS

ROS - Robot OS - est un environnement logiciel destiner au pilotage autonome de systèmes robotiques. Ce logiciel Open Source originellement développé conjointement par Willow Garage et l'université de Stanford, a été lancé pour sa version 1.0 en janvier 2010 (voir www.ros.org).

ROS en est maintenant à sa 11^{ème} distribution, dénommée "ROS Lunar Loggerhead", distribution qui sera utilisée pour Symeter V2.

ROS est un environnement logiciel destiné à la mise en oeuvre de plateforme robotique. Cet environnement très modulaire permet de programmer et de déployer de nombreux modules interdépendants, appelés **packages**, constitués eux de noeuds exécutables appelés **nodes**.

Ces nodes peuvent communiquer entre eux aux moyens de deux mécanismes principaux :

- les services

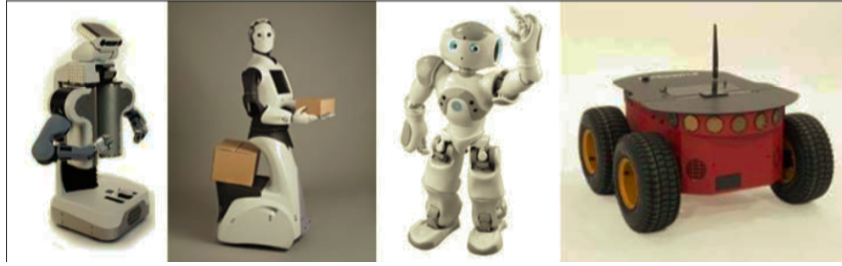


FIGURE 3.1 – Quelques exemples de robots utilisant ROS (image [MF13])

— les topics

Ces deux mécanismes permettent l'échange de données entre nodes selon des messages de la structure peut être spécifiée par le développeur du système.

Les **services** sont mis à disposition chacun par un node. Ils peuvent être invoqués par d'autre node en effectuant une transaction explicite : le node appelant fourni une donnée d'entrée au node mettant à disposition le service. Ce dernier traite la donnée entrante puis génère une donnée de sortie qui est enfin renvoyée au node, terminant ainsi la transaction.

Les **topics** sont des canaux de diffusion d'information similaire à une autoroute : tout node peut diffuser un message vers un topic et tout node peut écouter un topic pour recevoir et traiter les messages qui y sont émis.

Ces topics pourvoient des données que l'on pourrait qualifier de "sensorielle", dont la durée de vie est courte, telle que les données générées par un capteur fournissant des données en continu.

Ils sont aussi le support de diffusion des **transformations**, qui permettent de décrire l'état de **pose** de chacuns des composants mécaniques d'un véhicule autonome, ainsi que la pose du véhicule dans son environnement.

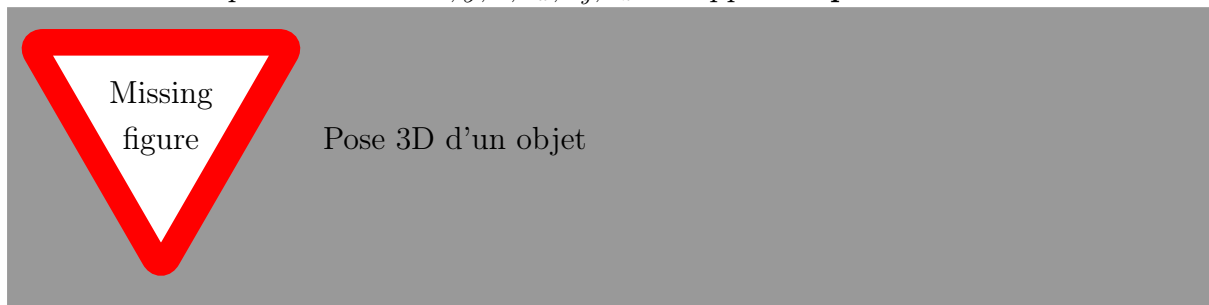
ROS est un environnement fourni sous une licence Open Source et propose des drivers pour de très nombreux capteurs et actionneurs de tous types. Il intègre aussi des composants logiciels tiers spécialisés tels que, entre autre, la Point Cloud Library pour le traitement des nuages de points et la librairie OpenCV pour le traitement de la vision par ordinateur.

3.1.1 Gestion des transformations

3.1.1.1 Poses

La robotique à pour enjeu de permettre la mise en oeuvre de systèmes mécaniques autonomes, mobiles ou non, qui mettent en oeuvre une série de capteurs et d'actionneurs pour agir sur leur environnement avec une interaction humaine très limitées.

La capacité de représenter de manière fine la position des différents éléments d'un tel système les uns par rapport aux autres, que ce soit en position x, y, z , mais aussi en inclinaison - θ_x (tanguage), θ_y (roulis), θ_z (lacet) - est donc primordiale. L'état de l'élément décrit par ce vecteur $x, y, z, \theta_x, \theta_y, \theta_z$ est appelé sa **pose**.



Les outils mathématiques de prédilection pour décrire une pose sont bien sur les transformations dans \mathbb{R}^3 , translations et rotations.

ROS utilise les quaternions de manière native dans la description de ses transformations.

REDIGER précision d'évolution d'un plateforme robotique dépend de la prise en compte de la position relative de ses différents capteurs et actionneurs

AJOUTER
Quaternion
quaternions

3.1.2 Gestions des Capteurs

REDIGER un petit laïus sur les capteurs supportés

3.2 Présentation de Gazebo

Gazebo est un logiciel qui permet de simuler un "monde" virtuel 3D, dans lequel il est possible de plonger des véhicules robotiques qui seront alors soumis aux lois physiques simulées.

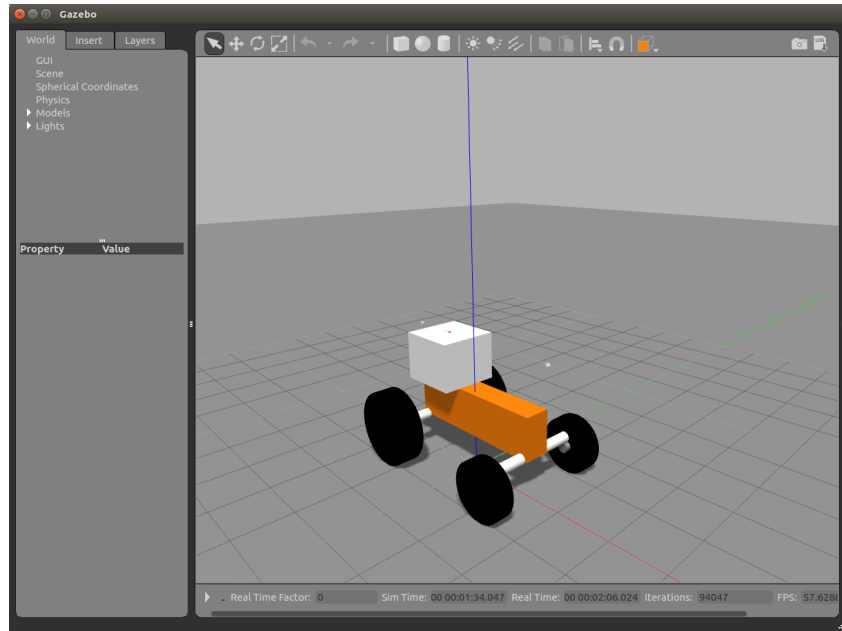


FIGURE 3.2 – Capture d'écran de l'interface graphique de Gazebo

3.2.1 Construction d'un robot virtuel

Un robot virtuel sous Gazebo est décrit par un fichier XML au format URDF. Ce format permet de spécifier un robot en ses éléments constitutifs simplifiés (chassis, roue, partie de bras, etc) de décrire comment ces éléments sont placés les uns par rapport aux autres.

Chaque élément du robot doit comporter 3 parties différentes :

- un tag `<collision>` qui décrit le modèle 3D avec lequel le simulateur doit calculer les collisions de l'élément avec d'autres objets,
- un tag `<visual>` qui décrit le modèle 3D de l'apparence visuelle de l'élément (pour l'affichage dans le simulateur),
- un tag `<inertial>` qui décrit les caractéristiques inertielles de l'élément.

Ce robot doit faire partie d'un monde qui est lui-même décrit par un fichier URDF.

TROUVER REF EXTERNE : SDF

REDIGER URDF et SDF, world, contrôleurs

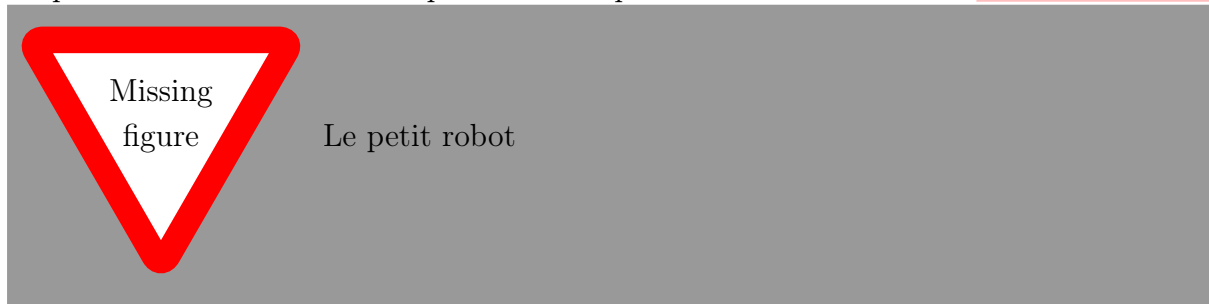
3.2.2 Vérification de disponibilité des capteurs

Gazebo propose tout une série de capteurs simulés, disponible nativement ou par l'intermédiaire de module tiers. Ces capteurs simulés génère une donnée en cohérence avec l'environnement simulé qui peut être rendu disponible aux modules ROS par l'intermédiaire de plugins adaptés.

Pour les besoins du projet Symeter V2, nous avons vérifié et testé la disponibilité des types de capteurs tels que listés dans le tableau suivant.

Capteur	Plugin Gazebo	Utilisabilité
IMU		
LIDAR		
GPS		

Pour vérifier la disponibilité et le bon fonctionnement de ces capteurs, nous avons utilisé un petit robot basé sur une implémentation pré-existante décrite dans .



AJOUTER
le tuto de p
robot

3.3 Contraintes de mise en oeuvre

La mise en oeuvre de ce petit robot au sein de Gazebo a été riche d'enseignements. Elle a permis notamment de détecter quelques problèmes gênant dans l'utilisation de Gazebo.

3.3.1 Pas d'adhérence au démarrage de la simulation

La mise en oeuvre du robot de base a révélé rapidement que Gazebo comporte quelques problèmes lié à la simulation de l'adhérence des roues avec les objets environnants, notamment le sol.

En particulier, les roues d'un robot ROS plongé dans Gazebo n'auront aucune adhérence sur le sol au démarrage de la simulation. Si la propulsion est mise en oeuvre, nous pouvons constater que les roues tournent mais que le robot ne bouge pas, comme si les roues tournaient dans le vide, ou sur de la glace.

Après avoir changé des réglages dans le moteur physique de Gazebo, dans le réglage du coefficient de frottement des roues et de nombreux autres paramètres sans effet sur le problème, un contournement a été mis en place pour le mitiger : un node ROS a été écrit pour forcer la position du robot à des coordonnées spatiale bien précises, en utilisant le topic `/gazebo/set_model_state`.

Après ce forçage, Gazebo prend ensuite bien en compte l'adhérence des roues du robot et le robot bouge quand les roues tournent.

Cela fait que ce script, dénommé **setpose**, doit être invoqué systématiquement après le démarrage de la simulation pour que l'adhérence des roues soit bien prise en compte par le moteur physique de Gazebo.

3.3.2 Simulation mécanique, frottements, adhérence

Simuler un robot avec des points de contact multiples dans Gazebo peut poser parfois problème si les points multiples dérapent. Dans ce cas il est rare que l'adhérence soit récupérée.

Ceci est exacerbé par le fait que les matériaux simulés par Gazebo sont par défaut infiniment rigides et que le moindre choc se répercute dans les points de contact avec le sol.

3.3.3 Conclusions sur les contraintes

Pour conclure avec les limitations et contraintes de Gazebo, la leçon principale est qu'il faut au maximul éviter que le véhicule entre en dérapage : il ne récupère généralement pas, et devient inutilisable : il faut relancer la simulation.

Les raisons pour lesquelles le véhicule peut entrer en dérapage sont nombreuse :

- Mauvaise configuration des paramètres de frottement des éléments en contact avec le sol.
- Vitesses de rotation différentes des roues, soit par une mauvaise consigne de vitesse sur l'une des roues, soit une non prise en compte du différentiel de vitesse entre roues intérieur et extérieur dans un virage, etc
- En virage, les roues intérieures et extérieures sont soumises à des rayons différents. S'ils ne sont pas pris en compte correctement, une ou plusieurs roues sont susceptible de déraiper.

3.4 Mise en Oeuvre : simulation d'un environnement de tassage de silo

3.4.1 Montage d'un tracteur simulé

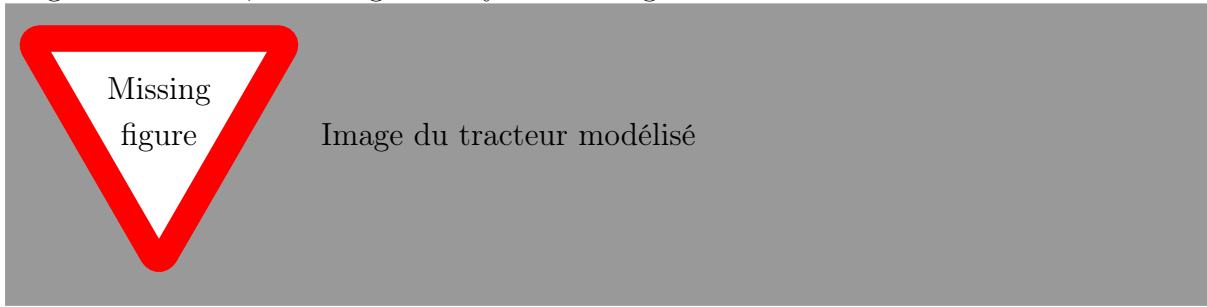
Intérêt : simuler l'implantation physique des capteurs avec des dimensions du même ordre de grandeur que les plateformes cibles.

3.4.1.1 Description Physique

3.4.1.1.1 Chassis Le chassis du tracteur à été monté en se basant sur les dimensions générales d'un tracteur de type **TODO : ICI TYPE DE TRACTEUR** , en utilisant

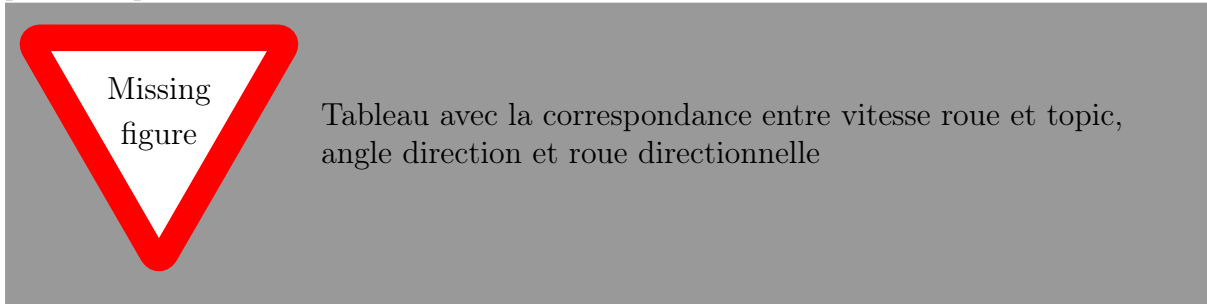
Ecrire type tracteur

Le tracteur est donc composé d'un pavé en guise de châssis, de 4 cylindres allongés en guise d'essieux, et de 4 grands cylindres en guise de roues.



3.4.1.1.2 Actuateurs et Contrôleurs Pour permettre une conduite en terrain accidenté, le tracteur simulé sera muni de 4 roues motrices, avec 2 roues directrices à l'avant.

Chacune des roues motrices est commandée en vitesse par son propre topic, telle que décrit dans le tableau suivant, et l'angle de chacune des roues de direction est commandée par un topic dédié.



3.4.1.2 Propulsion et Guidage

Du fait des contraintes exposées dans le paragraphe 3.3.3, la consigne de vitesse sur chacune des roues doit être cohérente vis-à-vis de la vitesse de consigne, vis-à-vis du rayon de chacune des roues, ainsi qu'avec le rayon de virage imposé. De plus l'angle de chacune des roues directrices doit être réglé de manière différentielle en fonction de l'angle de direction donné en consigne.

3.4.1.2.1 Algorithme

La problématique de la direction est illustrée par la figure 3.3 : si l'on donne une consigne d'angle de direction générale ϕ , quels doivent être les angles ϕ_l , pour la roue avant gauche, et ϕ_r , pour la roue avant droite, pour que ces deux roues restent tangentes à leur trajectoire propre ?.

Les données d'entrée sont les suivantes :

- L l'empattement du tracteur, c'est à dire la distance entre les deux essieux,
- l la distance entre l'axe avant/arrière du tracteur et le centre d'une roue avant (les roues sont supposées disposées de manière symétrique),
- ϕ l'angle de consigne.

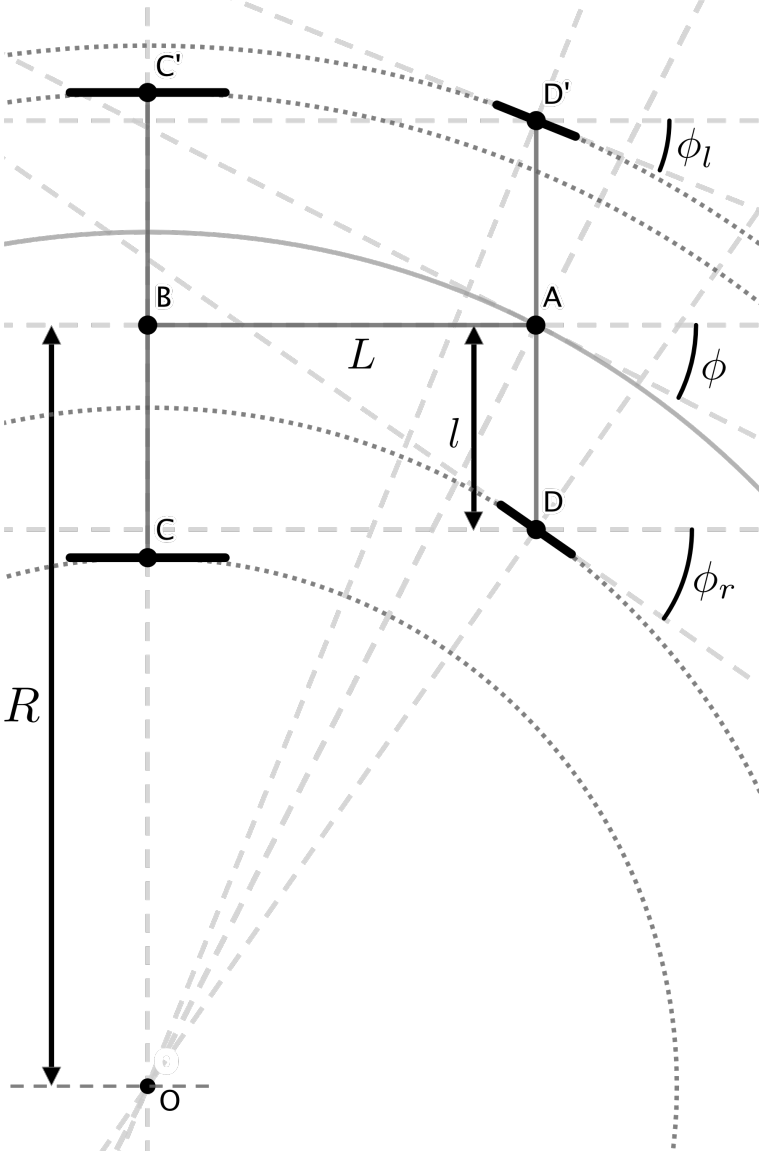


FIGURE 3.3 – Géométrie du différentiel de direction

Pour déterminer tous ces angles, nous d'abord devons déterminer R le rayon de courbure de la trajectoire du point central B de l'essieu arrière. Si nous considérons le triangle rectangle (OAB), nous constatons que l'angle (OA,OB) est le même que ϕ . Nous pouvons donc écrire, en posant $L = AB$ et $R = OB$

$$\tan \phi = \frac{L}{R} \quad (3.1)$$

d'où

$$R = \frac{L}{\tan \phi} \quad (3.2)$$

Par un raisonnement similaire nous pouvons trouver une formule pour ϕ_l et ϕ_r :

$$\tan \phi_l = \frac{L}{R+l} \iff \phi_l = \arctan \frac{L}{R+l} \quad (3.3)$$

$$\tan \phi_r = \frac{L}{R-l} \iff \phi_r = \arctan \frac{L}{R-l} \quad (3.4)$$

En incorporant 3.2 dans 3.3, nous obtenons

$$\phi_l = \arctan \frac{L}{\frac{L}{\tan \phi} + l} = \arctan \frac{L}{\frac{L+l \tan \phi}{\tan \phi}} = \arctan \frac{L \tan \phi}{L + l \tan \phi} \quad (3.5)$$

De manière équivalente,

$$\phi_r = \arctan \frac{L \tan \phi}{L - l \tan \phi} \quad (3.6)$$

Réglage des vitesses de rotation : En plus du différentiel d'angle des roues directionnelles, la vitesse de rotation de toutes les roues doit être précisément réglée lors un virage pour éviter tout dérapage.

Toujours selon la figure 3.3, nous constatons que, lors d'un virage à angle constant, chacune des roues parcourt son propre cercle de centre O.

Si la vitesse de consigne v est donnée pour le milieu de l'essieu arrière (point B), quelle doit être la vitesse de rotation des roues pour que la vitesse de consigne soit suivie par le point B ?

3.4.1.2.2 Implémentation sous ROS L'implémentation sous ROS de cet algorithme est effectué au moyen d'un node dédié, nommé `tracteur_steering.py`, qui fait partie du package `tracteur_control`.

Le node `tracteur_steering` prend en entrée le topic `/tracteur/cmd_vel` qui comporte des messages de type `Twist`. Ce message contient deux informations : l'un comporte la consigne de vitesse du tracteur, et l'autre le taux de rotation à gauche ou à droite du tracteur.

Le node `tracteur_steering` calcule sur la base des consignes données 6 nouveaux paramètres, sur la base des algorithmes exposés en 3.4.1.2.1 :

- les 4 consignes de vitesse pour chacune des roues
- les 2 consignes de direction pour chacune des roues directionnelles.

Ces consignes sont soumises à chacun des contrôleurs par l'intermédiaire des topics qui leur sont dédiés .

COMPLETER Calcul des vitesses de rotation



Schéma des flux topic qui permet de transformer une commande twist en 4 commandes de vitesses.

AJOUTER
tableau avec
topics

Description
l'implément
sous ROS

Chapitre 4

Mise en place du processus de localisation

Ce chapitre présente la problématique de localisation par fusion de données IMU et GPS. Malgré l'objectif de mettre en oeuvre des composants disponibles "sur étagère" pour monter la version initiale de Symeter V2, il est quand même nécessaire de comprendre en détail les tenants et les aboutissants du problème de la localisation.

Ce problème est complexe et requiert d'en comprendre au moins les bases afin de permettre de configurer les composants correctement.

4.1 Présentation du problème

Comme indiqué dans le paragraphe ??, le but du processus de localisation est d'estimer en temps réel pour chaque instant t_k le vecteur d'état

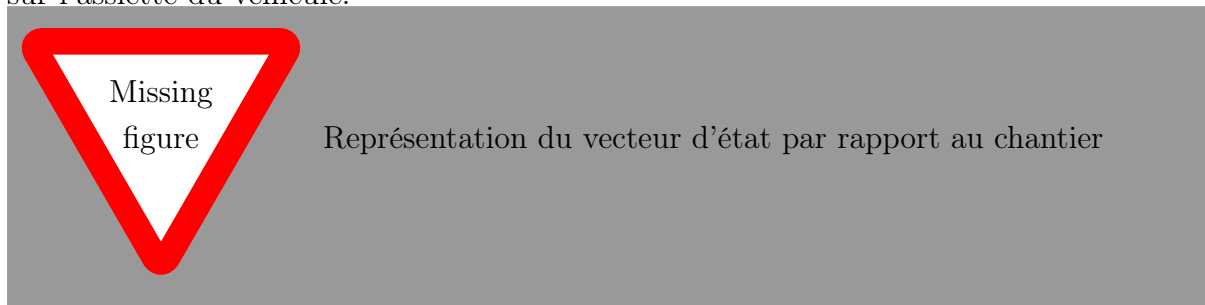
$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ z_k \\ \theta_{x_k} \\ \theta_{y_k} \\ \theta_{z_k} \end{bmatrix} \quad (4.1)$$

sur la base d'informations fournies par l'IMU et le GPS en mode RTK, ce vecteur d'état étant exprimé dans un référentiel fixe par rapport au chantier.

L'IMU fournit des informations sur l'inclinaison du véhicule, ainsi que les accélérations linéaires qu'il subit à une fréquence de l'ordre de 30 Hz. Cependant comme indiqué ci-dessus, calculer la position du véhicule à partir de la mesure des accélérations subies requiert une double intégration, sur un signal d'accélération qui est généralement très bruité. Ces éléments impliquent une forte probabilité de dérive dans le temps.

Le GPS en mode RTK peut fournir une mesure de la position du véhicule dans la scène avec une précision de l'ordre de quelques centimètres, mais à un taux de rafraîchissement

beaucoup plus bas, de l'ordre de 2 à 5 Hz. De plus le GPS ne fournit aucune information sur l'assiette du véhicule.



Selon [Gus15], il est possible de mettre en place un procédé qui maintient ce vecteur d'état sur la base de ces deux instruments, communément appelé la "fusion de capteurs". Nous mettrons en oeuvre un tel procédé.

Note Comme le GPS fournit une position absolue dans le référentiel de la scène nous ne sommes pas dans une problématique SLAM, où l'on tente de déterminer de manière imbriquée la position et l'environnement du véhicule. Dans notre cas, les deux problèmes sont décorrélés : d'abord nous déterminons la localisation et ensuite nous déterminons l'environnement.

4.1.1 Repères

L'IMU et le GPS peuvent fournir des informations à certains instants t_k qui permettront de mettre à jour le vecteur d'état pour l'incrément de temps suivant t_{k+1} . Cependant, les données exposées par les capteurs ne sont pas toutes dans le référentiel fixe par rapport au chantier.

L'IMU par exemple expose des composantes d'accélération linéaire qui sont attachées au référentiel de L'IMU lui-même, supposé fixe par rapport au véhicule, mais qui est donc mobile par rapport au chantier.

Le GPS lui n'est en mesure que de donner des mesures de positions que dans un référentiel lié à la surface de la Terre, en terme de latitude, longitude et altitude (λ, ϕ, h) et il faudra donc transformer cette position géocentrique en coordonnées du référentiel de chantier.

Pour travailler au problème de localisation, nous avons donc besoin d'au moins trois référentiels différents pour maintenir le vecteur d'état \mathbf{x}_k à partir des mesures de l'IMU et du GPS.

Selon [Gus15], nous pouvons donc distinguer les 4 référentiels suivants

4.1.1.1 Repère Inertiel

4.1.1.2 Repère ECEF

4.1.1.3 Repère de navigation local

4.1.1.4 Repère du véhicule

4.1.2 Système de Navigation Inertielle

Selon [Gus15], les données de l'IMU peuvent être exploitées par un Système de Navigation Inertiel. Un tel

4.1.3 Transformation des coordonnées géocentriques en coordonnées locale

4.2 Filtres de Kalman

Selon [Gus15] et [MS16], la fusion de données IMU et GPS peut être effectuée sur la base d'un filtre de Kalman. Cette section s'attache à décrire les principes de fonctionnement d'un tel filtre pour en permettre l'utilisation pour le projet Symeter V2.

Un filtre de Kalman est un algorithme qui permet d'estimer les états d'un système dynamique sur la base de mesures incomplètes ou bruitées. Il a été développé à la fin des années 50 et comprend de très nombreuses applications dans tous les domaines liés à l'instrumentation.

4.2.1 Filtres de Kalman Linéaires

Selon [ZM09], les filtres de Kalman sont applicables sur des systèmes dynamiques linéaires modélisés par un ensemble d'équations différentielles, décrites par la relation suivante :

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \mathbf{w} \quad (4.2)$$

où

- \mathbf{x} est le vecteur colonne des états du système,
- \mathbf{F} est la dynamique du système,
- \mathbf{u} un vecteur connu, souvent appelé vecteur de contrôle.
- \mathbf{w} est un bruit blanc, lui aussi exprimé sous forme d'un vecteur.

La matrice de bruit de process \mathbf{Q} est par définition liée au vecteur \mathbf{w} selon la formule :

$$\mathbf{Q} = \mathbf{E}[\mathbf{w}\mathbf{w}^T] \quad (4.3)$$

($\mathbf{E}[\bullet]$ représente la fonction "espérance de $[\bullet]$ ".)

Dans notre cas, le système Symeter V2 n'aura pas accès au contrôle, nous choisissons donc l'intégrer la composante \mathbf{Gu} dans le bruit de process \mathbf{w} . Le modèle dynamique 4.2 devient donc

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{w} \quad (4.4)$$

Toujours selon [ZM09], l'état du système est estimé à partir de mesures \mathbf{z} dont la formulation de Kalman impose qu'elles soient linéairement liées aux variables d'état par la relation

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v} \quad (4.5)$$

où

- \mathbf{z} est le vecteur des mesures
- \mathbf{H} est la matrice de transfert des mesures
- \mathbf{v} représente le bruit de mesure.

La matrice de bruit de mesure \mathbf{R} est reliée au vecteur de bruit de mesure selon la relation :

$$\mathbf{R} = \mathbf{E}[\mathbf{v}\mathbf{v}^T] \quad (4.6)$$

4.2.1.1 Discrétisation

Les filtres de Kalman étant des algorithmes discrets, il convient de discrétiser les relations précédentes afin de pouvoir monter les relations de récurrence des filtres.

Toujours selon [ZM09], si la période de mesure est T_s nous pouvons discrétiser la dynamique du système en trouvant la matrice fondamentale $\Phi(y)$ de 2 manières équivalentes

Evaluation de $\Phi(t)$ à l'aide de la Transformée de Laplace Inverse

Pour un système dynamique invariant dans le temps décrit par une matrice \mathbf{F} la matrice fondamentale $\Phi(t)$ peut être trouvée à l'aide de la formule suivante :

$$\Phi(t) = \mathcal{L}^{-1}[(s\mathbf{I} - \mathbf{F})^{-1}] \quad (4.7)$$

où \mathbf{I} est la matrice identité, \mathbf{F} est la dynamique du système et \mathcal{L}^{-1} est la transformée de Laplace inverse.

Evaluation de $\Phi(t)$ par expansion en séries de Taylor

La matrice fondamentale peut aussi être retrouvée en effectuant l'expansion en série de Taylor suivante :

$$\Phi(t) = e^{\mathbf{F}t} = \mathbf{I} + \mathbf{F}t + \frac{(\mathbf{F}t)^2}{2!} + \dots + \frac{(\mathbf{F}t)^n}{n!} + \dots \quad (4.8)$$

Nous pouvons enfin calculer la matrice fondamentale Φ_k pour une période d'échantillonnage T_s en prenant :

$$\Phi_k = \Phi(T_s) \quad (4.9)$$

La forme discrète de l'équation des mesures 4.5 devient

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \quad (4.10)$$

et 4.6 devient

$$\mathbf{R}_k = \mathbf{E}(\mathbf{v}_k \mathbf{v}_k^T) \quad (4.11)$$

où \mathbf{R}_k est une matrice composée des variances de chacune des sources de bruits de mesure.

4.2.1.2 Boucle de Kalman

Le filtre de Kalman pourrait être écrit en une seule équation mais il est généralement conceptualisé en deux phases distinctes, la **prédiction** et la **mise à jour**.

L'état du filtre de Kalman un instant donné est composé de

- \mathbf{x}_k l'estimation de l'état à l'instant k
- \mathbf{P}_k la matrice de covariance de l'erreur sur les composantes de \mathbf{x}_k .

Selon [MS16] La phase de **prédiction** utilise l'état estimé de l'instant précédent pour calculer une estimation de l'état actuel, en se basant sur le modèle dynamique. En partant de 4.4, la prédiction utilise les formules 4.12 et 4.13 pour déterminer $\hat{\mathbf{x}}_k$ l'état prédit et $\hat{\mathbf{P}}_k$ la matrice de covariance prédite.

$$\hat{\mathbf{x}}_k = \Phi_k \mathbf{x}_{k-1} \quad (4.12)$$

$$\hat{\mathbf{P}}_k = \Phi_k \mathbf{P}_{k-1} \Phi_k^T + \mathbf{Q}_k \quad (4.13)$$

Dans la phase de **mise à jour**, les observations de l'état courant (c'est à dire les mesures \mathbf{z}_n) sont incorporées pour corriger l'état prédit dans le but d'obtenir une meilleur précision.

Tout d'abord le **gain de Kalman** \mathbf{K} est calculé selon l'équation 4.14.

$$\mathbf{K} = \hat{\mathbf{P}}_k \mathbf{H}^T (\mathbf{H} \hat{\mathbf{P}}_k \mathbf{H}^T + \mathbf{R}_k)^{-1} \quad (4.14)$$

Ensuite l'état du filtre, représenté par \mathbf{x}_k , le vecteur d'état et \mathbf{P}_k la matrice de covariance selon les formules 4.15 et 4.16 respectivement.

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k) \quad (4.15)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\hat{\mathbf{P}}_k(\mathbf{I} - \mathbf{K}\mathbf{H})^T + \mathbf{K}\mathbf{R}\mathbf{K}^T \quad (4.16)$$

4.2.1.3 Fonctionnement conceptuel du Filtre de Kalman

Pour comprendre conceptuellement ce qu'accomplit un filtre de Kalman, il suffit d'analyser les formules 4.14 du gain de Kalman et 4.15 de mise à jour du vecteur d'état.

En effet, dans 4.14, si la matrice de covariance des mesures \mathbf{R}_k est prépondérante par rapport à $\hat{\mathbf{P}}_k$, la matrice de covariance de l'erreur du vecteur d'état, la magnitude de \mathbf{K} sera petite, l'influence des mesures \mathbf{z}_k dans 4.15 sera petite, c'est à dire que le filtre fera plus confiance à son estimation du vecteur d'état $\hat{\mathbf{x}}_n$ qu'aux mesures.

Par contre, si \mathbf{R}_k est petite par rapport à $\hat{\mathbf{P}}_k$, le gain de Kalman aura une grande magnitude, reflétant le fait que la mesure sera plus précise que l'estimation. Et donc \mathbf{x}_k dans 4.15 sera plus influencée par \mathbf{z}_k que par $\hat{\mathbf{x}}_n$.

Selon [ZM09], la formule du gain de Kalman 4.14 est en fait appelée **Gain de Kalman Optimal** et est obtenue par minimisation la variance de l'erreur d'estimation.

4.2.2 Estimation de Pose 3D : Filtres de Kalman Etendus

Dans la plupart des applications d'estimation d'état, la dynamique des systèmes à estimer n'est pas linéaire et le filtre de Kalman ne peut pas être appliqué en l'état car toute sa formulation se base sur des hypothèses de linéarités fortes.

Selon [MS16] et [ZM09], il est possible de généraliser les filtres de Kalman pour estimer des systèmes dont la dynamique est non-linéaire.

Selon [MS16], un tel procédé non linéaire peut être décrit par l'équation 4.17 suivante

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1} \quad (4.17)$$

où \mathbf{x}_k est le vecteur d'état, f est une fonction de transition d'état non linéaire et \mathbf{w}_{k-1} est le bruit de processus.

Dans le cas qui nous intéresse où Symeter V2 doit déterminer la pose 3D à 6 degrés de liberté du véhicule, \mathbf{x}_k représente la dite pose, f sera un modèle cinématique dérivé de la mécanique Newtonienne.

Selon [ZM09] l'extension des filtres de Kalman à une dynamique non-linéaire est effectuée en calculant à chaque itération \mathbf{F} le jacobien de f selon la formule 4.18

$$\mathbf{F} = \left. \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \quad (4.18)$$

et en discrétisant ce jacobien en utilisant la formule 4.19

$$\Phi_k = e^{\mathbf{F}t} = \mathbf{I} + \mathbf{F}t + \frac{(\mathbf{F}t)^2}{2!} + \dots + \frac{(\mathbf{F}t)^n}{n!} + \dots \quad (4.19)$$

En pratique l'approximation 4.20 est le plus souvent suffisante

$$\Phi_k \approx \mathbf{I} + \mathbf{F}t \quad (4.20)$$

La boucle de Kalman telle que décrite dans 4.2.1.2 peut ensuite être utilisée en remplaçant Φ par Φ_k la matrice fondamentale du jacobien de f .

4.3 Mise en oeuvre sous ROS.

Comme les sections précédentes le montrent, l'implémentation d'un système de localisation sur la base mesure par IMU et GPS ne sont pas une mince affaire et est plus un sujet de thèse qu'une sous partie de stage.

L'environnement ROS propose des modules prêts à l'emploi pour implémenter un processus de localisation pour les systèmes de robotique mobile. L'offre est cependant nombreuse et il a fallu vérifier que les solutions proposées répondent bien aux besoins de Symeter V2.

Le besoin le plus important est que la localisation doit être effectuée sur un vecteur d'état compatible avec une scène 3D. Cela veut dire que le vecteur d'état estimé doit être d'au moins 6 dimensions : 3 pour la position spatiale du véhicule et 3 pour son assiette.

Il a donc fallu éliminer tous les modules qui se limitaient à des évolutions 2D du système, d'autre qui n'acceptaient qu'un nombre limité de capteurs, ou alors qui ne supportaient pas le GPS, par exemple.

Le choix final s'est finalement reposé sur le module `robot_localization`, qui est décrit plus en détail dans les sections suivantes.

4.3.1 Module `robot_localisation`

Selon le site internet de `robot_localisation` (lien), ce module est une collection de nodes dédiés à l'estimation d'état. Chacun de ces nodes implémente un estimateur d'état non linéaire pour des véhicules évoluant dans un espace en 3 dimensions.

Il contient notamment le node `ekf_localization_node` qui est une implémentation d'un filtre de Kalman Etendu, et le node `navsat_transform_node` qui aide à l'intégration des données GPS.

Les caractéristiques principales du node `ekf_localization_node` sont les suivantes :

- Possibilité de fusionner un nombre arbitraire de capteurs
- Possibilité de prendre en compte de nombreux types de données différentes : odométrie, IMU, Pose avec covariance, etc.
- Possibilité de régler la configuration du node capteur par capteur
- Estimation en continu : dès que le node reçoit une mesure, il estime en continu le vecteur d'état, même en l'absence prolongée de nouvelle mesure, en utilisant un modèle interne pour la dynamique du véhicule.

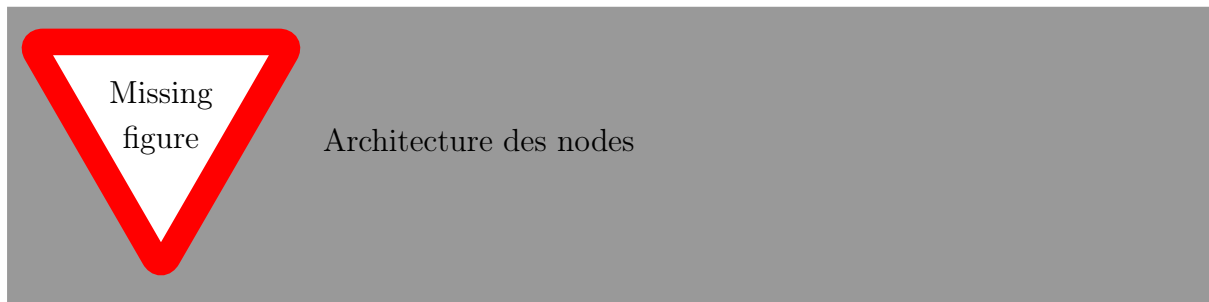
Pour estimer en continu l'état du véhicule, le node `ekf_localization_node` maintient en interne un vecteur d'état à 15 dimensions :

$$\begin{bmatrix} x & y & z & \theta_x & \theta_y & \theta_z & \dot{x} & \dot{y} & \dot{z} & \dot{\theta}_x & \dot{\theta}_y & \dot{\theta}_z & \ddot{x} & \ddot{y} & \ddot{z} \end{bmatrix} \quad (4.21)$$

4.3.2 `navsat_transformation_node`

REDIGER `navsat_transformation_node`

4.3.3 Configuration des Capteurs



REDIGER Les variables d'état influencées par IMU

REDIGER les variables d'état influencées par le GPS

4.3.4 Quelques tests

Chapitre 5

Exploitation des données LIDAR

Ce chapitre présente la fonction acquisition et exploitation des données LIDAR pour le projet Symeter V2.

Nous nous attachons à mettre en place les principes de traitement des données LIDAR en vue de construire une représentation 3D du chantier, d'abord en montant le pipeline de traitement du flux de données LIDAR, puis en mettant en oeuvre une structure de données adaptée au stockage d'une scène 3D.

5.1 Présentation de la chaine de traitement des données LIDAR


La chaîne de traitement des données LIDAR est représentée dans la figure et comporte les éléments suivants :

AJOUTER
figure

- Une source de données de type `laser`, issue des drivers ROS qui pilotent l'équipement LIDAR
- Un module de mise en forme du signal, qui prend en entrée les données `laser`, les filtre de manière à uniformiser la repartition des points de mesure, à réduire la bande passante et transformer les données laser en un nuage de points.
- Un module d'accumulation de nuages de points, qui prend en entrée les nuages de points issus du module de mise en forme en vue de les agréger et reconstruire la chaîne.



Chaine de traitement LIDAR -> Nuage de Point



Missing
figure

Transformer un ensemble de lignes en un nuage de points cohérent

5.2 Acquisition et mise en forme des données LIDAR

L'acquisition initiale des données LIDAR est effectuée par un driver approprié, généralement fourni par le constructeur de l'équipement. Les données sont structurées selon un format comportant la distance mesurée jusqu'au premier obstacle pour chaque incrément d'angle du LIDAR.

5.2.1 Transformation trame LIDAR en un nuage de points

La première étape du traitement est donc de transformer ces données de type `laser` (au sens de l'environnement ROS) pour générer un nuage de points dans le référentiel du chantier.

REDIGER LIDAR : ensemble d'angles, temps de vol, puissance reçue

REDIGER Nuage de points : positionnement x,y,z dans la scène

5.2.2 Filtrage de la ligne de point par downsampling

A raison de 40 scans par secondes et de 1080 points mesurés par scan, un LIDAR génère 43200 points par secondes, ce qui est beaucoup.

De plus, du fait que la hauteur du LIDAR monté sur un tracteur sera au maximum d'environ 3 à 4 mètres de hauteur, et que l'acquisition des mesures est effectuée sur la base d'un incrément d'angles, la densité des points de mesure sera beaucoup plus hétérogène que pour Symeter V1. Cela veut dire que nous aurons beaucoup plus de points de mesure directement sous le LIDAR que sur les côtés.

Nous allons donc effectuer un traitement qui permettra de moyenniser les mesures de points autour de positions régulièrement réparties tous les 5 cm (par exemple). Cela permettra de réduire le nombre de points à incorporer dans la scène, tout en régularisant la répartition des points dans l'espace.

D'après [ML16], il existe plusieurs méthodes de filtrage de nuages de points. Ceci est effectué en utilisant un procédé basé sur les Voxels

COMPLETER principe de fonctionnement des Voxels

Le résultat final de ce traitement est que d'un nuage de points irrégulièrement espacé et avec beaucoup de bande passante nous obtenons un nombre réduit de points régulièrement espacés, idéalement conditionné pour être pris en compte de manière fiable par l'accumulateur.

REDIGER Il est nécessaire de connaître précisément la pose du LIDAR pour générer le nuage de points.



Illustration nature donnée LIDAR



Illustration nature donnée Nuage de Point



Diagramme fonctionnel LIDAR -> Nuage de Point

5.3 Accumulation des nuages de points

Une fois les données LIDAR mises en forme, il convient maintenant de les exploiter pour reconstituer la représentation 3D du chantier. Cette tâche est effectuée par un module que l'on appelle le `point_cloud_aggregator`. Comme son nom l'indique ce module agrège et accumule les nuages de points issus de l'acquisition pour générer la représentation 3D du chantier.

5.3.1 Spécifications pour le `point_cloud_aggregator`

Le `point_cloud_aggregator` dans un premier temps agrège les nuages de points, c'est à dire qu'il prend un nuage de point en entrée, et tente de les ajouter au nuages de points qu'il a déjà en mémoire.

S'il tente d'ajouter un point à une position dans l'espace pour laquelle un point existe déjà en mémoire (collision), ce nouveau point n'est pas ajouté. Cela évite ainsi les points doubles.

Cela met cependant une contrainte forte sur la structure de données stockant les points déjà accumulés : il faut pouvoir identifier de manière unique tous les éléments de volume inclus dans la représentation de la scène afin de pouvoir détecter les collisions.

[SXZ17]

5.3.2 Principe de stockage des données 3D

REDIGER les structures de données de stockage de l'information 3D

TROUVER REF EXTERNE : les structures de données 3D

5.3.2.1 Les B-Trees

REDIGER Les b-trees, principes

REDIGER Point forts : calculs performants

REDIGER points faible : arbre équilibrés → difficile d'ajouter de nouveaux points.

5.3.2.2 Les Octrees

REDIGER Octree, principes

REDIGER points forts : structure déséquilibrées sans problème, possibilité d'ajouter de nouveaux points avec une bonne performance

REDIGER point faible : beaucoup d'overhead de mémoire si pas implémenté correctement.

5.3.2.3 Le choix : octree correspond à notre besoin.

5.3.3 Mise en oeuvre : octomap

REDIGER Utilisation d'octomap car composant sur étagère

REDIGER permet d'implémenter rapidement la chaîne de traitement pour vérifier la validité de la faisabilité

5.4 Mise en oeuvre sous Gazebo

La section précédente décrivait les principes de traitement des données LIDAR en vue de construire une représentation 3D d'un chantier d'ensilage. La présente section va illustrer la mise en oeuvre de ces principes à l'aide de la plateforme de simulation décrite dans le chapitre 3 évoluant dans un chantier d'ensilage simulé.

5.4.1 Modélisation d'un chantier d'ensilage

Le modèle simulé du chantier d'ensilage est représenté pour 2 murs chacun long de 10 mètres, haut de 3 mètres et d'épaisseur 0,2 mètre, espacés de 9 mètres. Ce modèle est créé en utilisant les primitives XML fournies par le logiciel Gazebo.



Capture d'écran représentant le mur sous Gazebo.

Un tas d'ensilage peut aussi être ajouté au chantier. Ce modèle de tas d'ensilage a été construit en utilisant dans un premier temps le logiciel de modélisation 3D **blender**. Ce modèle 3D est ensuite incorporé dans le chantier simulé en l'important dans la description du world 3D de manière appropriée.



Capture d'écran modèle d'ensilage sous blender + capture d'écran mur et ensilage dans Gazebo

5.4.2 Test sous gazebo

Une fois la simulation de chantier montée, il est possible de faire évoluer le tracteur simulé dans le chantier. L'opérateur peut ainsi diriger le tracteur de manière à ce que le LIDAR monté à l'arrière du tracteur puisse scanner séquentiellement dans son ensemble le silo.



Capture Gazebo du chantier avec tracteur et mur

REDIGER Une fois l'acquisition effectuée, on invoque le node de sauvegarde de nuage, qui le sauve dans un fichier

5.4.3 Analyse du nuage de point généré

5.4.3.1 Outil pour l'analyse de nuage de points : Paraview

REDIGER Outil de visualisation utilisé : Paraview

REDIGER besoin de convertir pcd en vtk à l'aide de l'outil approprié

REDIGER permet de naviguer de manière efficace dans le nuage de point, d'effectuer des projections, etc, etc

5.4.3.2 Points à améliorer sur le nuage de points

REDIGER Le nuage est épais

REDIGER octomap génère un nuage de point basé sur le centre des voxel occupés -> problème de quantification volumique implique perte de précision de la mesure

REDIGER Soit réduire la taille du voxel -> augmentation de la taille

REDIGER Soit retourne le point moyen de chaque voxel -> non supporté par octomap -> à implémenter nous même.

AJOUTER
octomap

Chapitre 6

Mise en oeuvre à partir de mesures réelles

Les chapitres précédents ont montrés comment le système Symeter V2 a été initialement conçu et monté à l'aide d'outils de simulation. Après avoir obtenu une première série de résultat par ce biais, quelques questions ont émergé concernant la validité de certains aspects de la simulation, et de savoir si la simulation représentait un chemin réaliste de développement.

C'est pour cela qu'une campagne d'acquisition de données réelle, à l'aide des capteurs physiques réels a été effectuée.

6.1 Protocole de test

6.1.1 Instruments d'acquisition)

REDIGER TellusCar avec Instrument montés sur l'arrière

REDIGER données GPS, données IMU et données LIDAR, toutes acquises en même temps



6.1.2 Exploitation

REDIGER Capture Bagfile alors que la TellusCar évolue dans le parking de adjacent à TellusEnv

REDIGER La partie logiciel de Symeter V2 peut exploiter des rejeux des captures pour estimer l'évolution de la TellusCar, et tenter de reconstituer le parking.

6.2 Données générée - visualisation sous google maps

En rejouant le bagfile nous pouvons extraire et visualiser les données brutes.



trajectoire sous Googlemap



graphe des accélérations et de l'assiette issues de l'IMU



données LIDAR brutes

6.3 Exploitation des données

6.3.1 Configuration : description des poses des instruments par rapport au repère du véhicule

6.3.2 Intégration des données GPS et IMU pour localisation

REDIGER Il a été compliqué de faire fonctionner la partie localisation de Symeter V2 avec les données réelle

REDIGER grande sensibilité des données IMU si la pose de l'IMU n'est pas rigoureusement décrite dans la configuration de symeter V2

REDIGER Obligé de traiter les données IMU pour rétablir la bonne orientation

REDIGER L'imu mesure le cap avec 0 au nord alors que robot_localization s'attend à un 0 pour un cap à l'Est

6.3.3 Reconstitution du parking

REDIGER une fois le système de localisation OK, pb avec Octomap qui n'a pas l'air de fonctionner au delà d'une certaine distance

6.4 Récapitulation

REDIGER Illustre les problématiques du passage de la simulation à une intégration en grandeur

REDIGER IMU simulé n'est pas fiable : il faudra en trouver un autre

REDIGER A l'heure actuelle pas encore terminé

Chapitre 7

Reste à faire et Axes Améliorations

PEUT ETRE caméra video pour odométrie visuelle ?

7.1 Difficulté de Simulation de l'action de tassage

La simulation peut permettre la mise au point de la localization, de l'acquisition initiale du chantier d'ensilage. Mais elle n'est pas directement utilisable pour simuler l'opération du tracteur dans le chantier d'ensilage. En effet pas possible de simuler physiquement le tassage à l'aide de Gazebo.

Il est cependant possible de mettre en oeuvre nos outils de manière à simuler logiquement l'évolution du chantier en implémentant un mécanisme de sauvegarde / restauration de l'état du chantier

REDIGER Decrire le cycle de simulation



Diagramme d'état simulation de l'acte de tassage.

Chapitre 8

Conclusion

A l'issue de se stage

Confirmé que la simulation pouvait permettre le développement du produit symeter2, nous avons implémenté une plateforme de test simulé sous Gazebo,

nous avons mis en place le système de localisation à partir en mettant en oeuvre un composant pré-existant de fusion de capteurs à partir des signaux de l'IMU et du GPS.

nous avons mis en place la chaîne de traitement des données Lidar en utilisant principalement les composants "sur étagère" Point Cloud Library et Octomap.

nous avons généré un nuage de points 3D d'un chantier d'ensilage simulé, validant ainsi la faisabilité de Symeter V2.

Nous avons aussi tenté d'utiliser le prototype de Symeter V2 pour exploiter des captures réelles.

Nous avons ainsi pu dégager des axes d'améliorations pour parvenir à améliorer la localisation des points, et ainsi améliorer la qualité du nuage de point.

Annexe A

Représentation de l'Attitude et des Rotations

Selon [Gus15], si nous considérons un véhicule mesurant des mouvements par rapport à son propre repère et un autre système mesurant les positions et vitesses etc du même véhicule par rapport à un repère $\{e\}$, et si la navigation est effectuée par rapport à un repère local de navigation n , il devient très vite apparent qu'un outil pour décrire l'orientation du véhicule par rapport à ces différents repères est nécessaire.

Deux des manières les plus communes de représenter les attitudes et les rotations dans \mathbb{R}^3 sont les *Angles d'Euler* et les *quaternions*.

A.1 Angles d'Euler

Selon [Nü09], alors que les trois coordonnées Cartésiennes x, y et z représentent la position, les trois angles θ_x, θ_y et θ_z décrivent l'orientation dans \mathbb{R}^3 . L'union de la position et de l'orientation est appelée la *pose*. L'orientation θ_x, θ_y et θ_z est la rotation autour des axes principaux du repère orthonormé, c'est à dire $(1, 0, 0)$, $(0, 1, 0)$ et $(0, 0, 1)$. Les matrices de rotation sont les suivantes :

$$\begin{aligned}\mathbf{R}_x &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{pmatrix} \\ \mathbf{R}_y &= \begin{pmatrix} \cos \theta_y & 0 & \sin \theta_y \\ 0 & 1 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y \end{pmatrix} \\ \mathbf{R}_z &= \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix}\end{aligned}$$

La matrice de rotation globale est calculée comme $\mathbf{R} = \mathbf{R}_{\theta_x, \theta_y, \theta_z} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z$.

Note : La matrice ci-dessus dépend de l'ordre de la multiplication. Des matrices de rotation différentes sont obtenues par $\mathbf{R} = \mathbf{R}_{\theta_z, \theta_y, \theta_x} = \mathbf{R}_z \mathbf{R}_y \mathbf{R}_x$ ou $\mathbf{R} = \mathbf{R}_{\theta_y, \theta_x, \theta_z} = \mathbf{R}_y \mathbf{R}_x \mathbf{R}_z$. De plus un blocage de cardan peut intervenir si les axes

A.2 Quaternions Unitaires

Toujours selon [Nü09], les quaternions sont un nombre complexe à 4 dimensions qui peuvent être utilisés pour représenter des rotations en 3D. Ils sont générés en postulant une racine additionnelle à -1 nommée j qui ne soit pas i ou $-i$. Il existe alors nécessairement un élément k tel que $ij = k$ qui aussi une racine de -1. Les relations suivantes sont aussi vraies :

$$\begin{aligned} i^2 = j^2 &= k^2 = ijk = -1 \\ ij = k, & \quad ji = -k \\ jk = i, & \quad kj = -i \\ ki = j, & \quad ik = -j \end{aligned}$$

i, j, k et -1 forment la base du quaternion \mathbb{H} . Tout quaternion $\dot{\mathbf{q}}$ peut être représenté sous la forme $\dot{\mathbf{q}} = q_0 + q_x i + q_y j + q_z k$. Ainsi tout quaternion définit un unique 4-vecteur $(q_0, q_x, q_y, q_z)^T \in \mathbb{R}^4$.

Comme pour les nombres complexes, les quaternions possèdent un unique quaternion conjugué. Le quaternion conjugué de $\dot{\mathbf{q}} = q_0 + q_x i + q_y j + q_z k$ est $\dot{\mathbf{q}}^* = q_0 - q_x i - q_y j - q_z k$.

En utilisant le quaternion conjugué et le produit scalaire, nous pouvons calculer la norme du quaternion $\dot{\mathbf{q}}$:

$$\|\dot{\mathbf{q}}\| = \sqrt{\langle \dot{\mathbf{q}}, \dot{\mathbf{q}}^* \rangle} = \sqrt{q_0^2 + q_x^2 + q_y^2 + q_z^2}$$

Le sous-ensemble des quaternions unité est important car ses éléments peuvent représenter les rotations dans \mathbb{R}^3 . Un quaternion unité $\dot{\mathbf{q}}$ vérifie la relation $\|\dot{\mathbf{q}}\| = 1$. Un quaternion unité a pour propriété que son inverse $\dot{\mathbf{q}}^{-1}$ est égal à son conjugué $\dot{\mathbf{q}}^*$.

$$\langle \dot{\mathbf{q}}, \dot{\mathbf{q}}^* \rangle = \|\dot{\mathbf{q}}\|^2 = 1$$

Les rotations utilisant les quaternions sont formalisées comme suit : soit $\dot{\mathbf{q}}$ tel que $\dot{\mathbf{q}} = (q_0, q_x, q_y, q_z) = (q_0, \mathbf{q})$. Pour effectuer la rotation du point 3D $\mathbf{p} = (p_x, p_y, p_z)^T \in \mathbb{R}^3$, nous écrivons ce point sous la forme d'un quaternion aussi : $\dot{\mathbf{p}} = (0, p_x, p_y, p_z)^T = (0, \mathbf{p})$. Alors nous pouvons démontrer que le point \mathbf{p}_{rot} est le résultat d'une rotation du point \mathbf{p} par la formule

$$\mathbf{p}_{\text{rot}} = \dot{\mathbf{q}} \dot{\mathbf{p}} \dot{\mathbf{q}}^*$$

[Nü09] montre qu'une rotation d'un angle θ autour d'un vecteur \mathbf{n} sera représenté par le quaternion

$$\dot{\mathbf{q}} = \left(\cos \frac{\theta}{2}, \sin \frac{\theta}{2} \mathbf{n} \right)$$

Pour passer d'une représentation en quaternion en une représentation de rotation autour d'un axe, il suffit de calculer les composantes suivantes :

$$\begin{aligned}\theta &= 2 \arccos q_0 \\ n_x &= \frac{q_x}{\sin \theta} \\ n_y &= \frac{q_y}{\sin \theta} \\ n_z &= \frac{q_z}{\sin \theta}\end{aligned}$$

Bibliographie

- [Gus15] Kenneth Gustavsson. UAV Pose Estimation using Sensor Fusion of Inertial, Sonar and Satellite Signals, 2015.
- [Jef10] Charles Jeffrey. *An introduction to GNSS : GPS, GLONASS, Galileo and other Global Navigation Satellite Systems*. NovAtel, Calgary, 2010. OCLC : 1036065024.
- [KH06] Elliott D. Kaplan and C. Hegarty, editors. *Understanding GPS : principles and applications*. Artech House mobile communications series. Artech House, Boston, 2nd ed edition, 2006. OCLC : ocm62128065.
- [MF13] Aaron Martinez and Enrique Fernández. *Learning ROS for robotics programming : a practical, instructive, and comprehensive guide to introduce yourself to ROS, the top-notch, leading robotics framework*. Community experience distilled. Packt Publ, Birmingham, 2013. OCLC : 861540246.
- [ML16] Carlos Moreno and Ming Li. A Comparative Study of Filtering Methods for Point Clouds in Real-Time Video Streaming. page 5, 2016.
- [MS16] Thomas Moore and Daniel Stouch. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, volume 302, pages 335–348. Springer International Publishing, Cham, 2016.
- [NKG13] Aboelmagd Noureldin, Tashfeen B. Karamat, and Jacques Georgy. *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [Nü09] Andréas Nüchter. *3D Robotic Mapping : The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*. Springer, 2009.
- [RS12] TAMAKI RS. Scanning Laser Range Finder UTM-30lx-EW Specification. page 7, December 2012.
- [Ste14] Gerald Steinbauer. TEDUSAR White Book - State of the Art in Search and Rescue Robots, 2014.
- [SXZ17] Shashi Shekhar, Hui Xiong, and Xun Zhou, editors. *Encyclopedia of GIS*. Springer International Publishing, Cham, 2017.

- [YSA⁺13] O. Yalcin, A. Sayar, O.F. Arar, S. Akpinar, and S. Kosunalp. Approaches of Road Boundary and Obstacle Detection Using LIDAR. *IFAC Proceedings Volumes*, 46(25) :211–215, 2013.
- [ZM09] Paul Zarchan and Howard Musoff. *Fundamentals of Kalman filtering : a practical approach*. Number v. 232 in Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, Reston, Va, 3rd ed edition, 2009. OCLC : ocn457170744.