ChatGPT

Étude bibliographique des concepts du projet d'autonomie robotique

1. Concepts implémentés dans le projet

Plateforme ROS 2 et robot LIMO Pro : Le projet s'appuie sur ROS 2 (Robot Operating System, v2), un framework logiciel open-source très répandu en robotique depuis sa première version en 2007 [1]. ROS 2, lancé en 2017, utilise une architecture DDS (Data Distribution Service) offrant une communication temps-réel plus fiable et scalable que ROS 1 (qui était basé sur TCP) [2]. La plateforme robotique matérielle est LIMO Pro, un petit robot mobile à quatre roues motrices. Le LIMO Pro embarque un ordinateur NVIDIA Orin Nano, un LiDAR 2D EAI T-mini Pro et une caméra de profondeur Orbbec, fournissant les capacités de perception et de calcul nécessaires pour la navigation autonome, l'évitement d'obstacles et la reconnaissance visuelle [3]. Le projet utilisera également Gazebo pour la simulation 3D du robot et de son environnement, ainsi que RViZ2 pour la visualisation des capteurs et du positionnement, afin de développer et tester les algorithmes avant de passer aux essais réels sur LIMO.

Perception par caméra (détection d'obstacles par YOLO) : La composante perception du projet utilise une caméra RGB couplée à un modèle de détection d'objets en temps réel de type YOLO (You Only Look Once). Les modèles YOLO font partie des détecteurs d'objets monétaires à un seul étage (single-stage) développés initialement par Redmon *et al.* et sont réputés pour leur grande rapidité d'inférence tout en conservant une bonne précision [4]. En particulier, YOLOv5 (mentionné dans le plan du projet, avec variantes *nano/tiny* allégées) est largement adopté pour la détection embarquée car il offre des temps d'inférence très courts et une faible exigence en ressources de calcul, le rendant efficace pour des applications temps réel sur des robots mobiles [5]. Ce réseau de neurones convolutionnel, bâti sur PyTorch avec un backbone CSPDarknet53, extrait des caractéristiques multi-échelles et produit en une passe les *bounding boxes* des obstacles avec leurs probabilités de classe [6] [7]. Grâce à ces techniques (ancres, suppression non-maximale, etc.), YOLOv5 peut détecter en temps réel divers objets (piétons, véhicules, obstacles divers) dans l'image, ce qui en fait un atout pour la navigation autonome où il est crucial d'identifier rapidement les obstacles sur la route du robot. Dans notre projet, un tel modèle de vision permettra de **détecter les obstacles et zones d'intérêt** par caméra frontale, complétant ainsi les données du LiDAR par une interprétation sémantique de la scène (par exemple distinguer un piéton d'un objet statique).

Perception par LiDAR (télédétection laser) : En parallèle de la caméra, le robot utilise un capteur LiDAR 2D (télémètre laser rotatif) qui fournit une mesure précise des distances des obstacles autour du robot sur un plan horizontal. Le LiDAR est un capteur très prisé en robotique mobile pour sa capacité à fournir des mesures de distance très fiables et à 360° autour du robot, indépendamment des conditions de luminosité. Il produit un nuage de points (ou des scans 2D) indiquant la position des obstacles proches. Toutefois, le LiDAR à lui seul ne distingue pas la nature des obstacles et opère à une fréquence plus basse et avec moins de « features » qu'une caméra [8] [9]. C'est pourquoi l'approche du projet combine les **deux capteurs** : la caméra apporte des informations riches visuellement (couleur, forme) et une fréquence élevée pour les objets proches, tandis que le LiDAR offre une portée plus longue et une meilleure précision de profondeur sur 360° [9]. Cette complémentarité capteur caméra/LiDAR est essentielle pour une perception robuste en environnement complexe.

Fusion de données capteurs (estimation d'état par EKF) : Afin de produire une **estimation robuste de l'environnement et de l'état du robot**, le projet implémente un module de fusion de capteurs, typiquement via un Filtre de Kalman Étendu (EKF) en C++. La fusion de capteurs consiste à combiner de manière optimale les informations redondantes ou complémentaires issues de multiples capteurs (odométrie, IMU, LiDAR, caméra) pour améliorer la précision et la fiabilité de l'estimation de l'état (position, orientation, vitesses) du robot. Au cœur de tout système multi-capteurs, on retrouve en effet des méthodes de fusion telles que les filtres de Kalman, filtres à particules ou des approches basées sur l'IA, ces méthodes étant déterminantes pour la performance globale du système [10]. Le filtre de Kalman standard fournit une estimation récursive en temps réel de l'état d'un système linéaire en minimisant l'erreur de covariance, mais pour un système non-linéaire (comme le mouvement d'un robot mobile avec mesures polarimétriques LiDAR/caméra), on utilise son extension non linéaire, le Filtre de Kalman Étendu (EKF) [11]. L'EKF lisse les incertitudes de l'odométrie en corrigeant la position du robot grâce aux observations capteurs : dans notre cas, on pourra intégrer les mesures de distance du LiDAR (détection d'obstacles ou alignement sur les bords de murs) et éventuellement des observations de la caméra (par ex. position d'un obstacle détecté) comme entrées de correction du filtre. Une alternative possible est le filtre à particules (PF), souvent utilisé pour la localisation en robotique mobile et en SLAM, qui représente la distribution de probabilité de l'état par un ensemble de particules plutôt que par une gaussiane unique. D'une manière générale, **la fusion multi-capteurs par EKF** est une technique classique et éprouvée pour augmenter la précision de localisation d'un robot mobile tout en filtrant le bruit des capteurs [12] [13]. Notre module de fusion combinera ainsi l'odométrie du robot (estimée à partir des encodeurs de roue ou de l'IMU) avec les mesures du LiDAR (et possiblement les données de détection caméra) pour obtenir en continu la meilleure estimation possible de la **pose du robot et/ou des obstacles** autour de lui.

Localisation et SLAM (cartographie simultanée) : Le projet vise à ce que le robot se localise de façon autonome tout en construisant la carte de l'environnement inconnu – c'est le problème classique de SLAM (*Simultaneous Localization and Mapping*). Le SLAM est défini comme le problème pour un robot de construire ou mettre à jour une carte d'un environnement inconnu tout en conservant l'estimation de sa propre position dans cette carte [14]. Bien qu'il s'agisse d'un problème difficile (dilemme de « la poule et l'œuf » : localisation nécessaire pour cartographier, mais carte nécessaire pour se localiser), de nombreux algorithmes approchés existent pour y parvenir en temps raisonnable. Parmi les méthodes populaires figurent les filtres à particules Monte Carlo (par ex. *AMCL* en 2D), les filtres de Kalman étendus (EKF SLAM) ou les approches par optimisation graphe (*GraphSLAM*) [14]. Concrètement, dans notre projet, on pourra utiliser des packages ROS 2 existants comme **SLAM Toolbox ou Cartographer** (de Nav2) pour effectuer le SLAM 2D à partir des scans LiDAR, construisant ainsi une carte 2D de l'environnement tout en calculant la pose du robot. Ces outils utilisent en général une combinaison de scan-matching laser (ICP ou corrélation) et d'un filtre (Kalman ou particulaire) pour estimer simultanément la trajectoire du robot et la carte des obstacles. Par ailleurs, l'intégration de la caméra pourrait ouvrir la voie à du **SLAM sémantique** ou à la gestion des obstacles dynamiques : en filtrant les obstacles en mouvement détectés par YOLO, on peut améliorer la consistance de la carte statique. Des travaux récents explorent l'incorporation de détecteurs d'objets dans le SLAM pour rendre celui-ci robuste aux environnements dynamiques – par exemple, l'algorithme YOLO-SLAM utilise un réseau YOLO léger pour détecter et retirer les objets mobiles des calculs de carte [15]. Dans notre contexte, le SLAM servira principalement à la **localisation globale** du robot et à fournir une carte pour la planification, avec l'aide potentielle du LiDAR (et des caméras pour identifier ou ignorer certains obstacles mobiles).

Planification de trajectoire (path planning global et local) : Une fois le robot capable de se localiser et de percevoir obstacles et libres espaces, il doit planifier un chemin sûr vers un objectif. Le projet prévoit un **planificateur global** (par exemple un algorithme de plus court chemin comme Dijkstra ou A*) couplé à un planificateur local pour l'évitement d'obstacles dynamiques. L'algorithme de Dijkstra, proposé en*

1959, calcule le plus court chemin dans un graphe pondéré (par exemple la grille de la carte) et garantit d'atteindre la solution optimale. L'algorithme A *(prononcé "A star"), publié en 1968 par Hart, Nilsson et Raphael, est une amélioration heuristique de Dijkstra : il utilise une fonction heuristique pour orienter la recherche, ce qui le rend plus efficace tout en conservant l'optimalité du chemin trouvé* [16]*. A est complet et optimiste optimal, et a été initialement appliqué à la planification de routes pour le robot Shakey – il est aujourd'hui un standard de la planification de trajectoire robotique* [16]*. Dans un contexte ROS 2, le* Nav2 planner *utilise par défaut A* sur une carte de coûts 2D pour produire le trajet global optimal du point de départ au but en évitant les zones occupées par des obstacles.

En complément du plan global, il est nécessaire d'avoir un module de **planification locale / contrôle d'obstacle** pour réagir aux obstacles imprévus (par exemple un obstacle mobile non pris en compte dans la carte globale). Le projet mentionne un éventuel usage d'un algorithme comme **Dijkstra/A** *(plutôt global) et laisse entendre qu'un contrôle simple ou un planificateur local sera implémenté. Dans la navigation ROS classique, un algorithme tel que le Dynamic Window Approach (DWA) sert de planificateur local (c'est d'ailleurs l'option par défaut du* move_base *ou Nav2 de ROS). Le DWA est une méthode d'évitement local très répandue qui effectue une recherche dans l'espace des vitesses du robot : il simule différents couples de vitesses angulaires et linéaires du robot dans une fenêtre temporelle courte, et choisit celui qui permet le meilleur progrès vers la cible tout en évitant les collisions immédiates* [17]*. En d'autres termes, à chaque instant, le DWA évalue plusieurs trajectoires circulaires possibles (selon les vitesses) et sélectionne celle qui est sans collision***** et la plus avantageuse pour se diriger vers le but. Ce type d'approche réactive permet d'éviter les obstacles dynamiques en temps réel. Comparé à d'autres méthodes locales (p.ex. champs potentiels), DWA est basé sur l'espace des vitesses du robot et assure un mouvement cinématiquement réalisable, ce qui en fait un choix efficace pour un robot différentiel [17]. Dans notre projet, le planificateur local pourrait être implémenté via le DWA de Nav2 ou de manière simplifiée par un contrôle réactif (ex: ralentir/s'arrêter devant un obstacle soudain détecté par le LiDAR/caméra).

Boucle de contrôle et commande du robot : Enfin, pour exécuter le trajet prévu, le robot doit convertir le chemin en commandes moteur. Le projet utilisera une **boucle de commande en temps réel** (un nœud ROS2 en C++) envoyant périodiquement des consignes de vitesse linéaire et angulaire (`geometry_msgs/Twist`) au robot. Cette boucle pourra être régulée par un contrôleur PID (Proportionnel–Intégral–Dérivé) simple ou même un correcteur proportionnel, afin de suivre le chemin désiré avec précision. Le contrôleur PID est un mécanisme de rétroaction classique très utilisé en robotique car il permet de minimiser l'erreur entre la vitesse souhaitée et la vitesse réelle du robot en ajustant dynamiquement la puissance des moteurs [18]. Par exemple, pour suivre une trajectoire, un PID angulaire peut corriger l'orientation du robot vers le point suivant du chemin, tandis qu'un PID linéaire maintient la vitesse adéquate, assurant un suivi de trajectoire fluide. Ces contrôleurs seront **réglés (tunés)** empiriquement pour garantir la stabilité et éviter les oscillations. Le nœud de commande devra tourner à haute fréquence (par ex. 50 Hz) pour réagir rapidement, d'où l'intérêt de le développer en C++ pour minimiser la latence. L'ensemble « perception → fusion → planification → commande » forme une boucle complète de navigation autonome qui sera testée d'abord en simulation Gazebo puis en conditions réelles sur le robot LIMO.

2. Potentiel de contribution du projet et perspectives de recherche

Le projet tel que décrit rassemble **plusieurs techniques connues** en robotique mobile (détection d'obstacles par vision, fusion LiDAR-caméra, SLAM 2D, planification A*, etc.). Pris individuellement, chacun de ces concepts est bien établi dans la littérature. Néanmoins, l'originalité et le potentiel de contribution scientifique résident dans l'intégration novatrice de ces composantes et l'amélioration de leurs performances dans un contexte précis (navigation autonome ROS2 sur robot LIMO en environnement potentiellement*

dynamique). Au vu de l'état de l'art actuel, il y a un intérêt de la communauté de recherche pour des solutions combinant l'IA de perception et la navigation robotique : par exemple, Adiuku et al. *(2024) ont montré qu'en intégrant un détecteur YOLOv5 dans la boucle de navigation ROS, le robot améliore significativement sa capacité à détecter et éviter les obstacles sans collision lors de déplacements dans un hangar industriel* [19]*. De même, des travaux récents explorent la fusion caméra + LiDAR pour la détection d'obstacles dynamiques en temps réel : Xu* et al. *(2025) proposent un cadre fusionnant les données d'une caméra RGB-D et d'un LiDAR sur un robot indoor, combinant plusieurs détecteurs légers et un filtre de Kalman pour suivre les obstacles et identifier lesquels sont en mouvement* [20]*. Ces études confirment que le thème de la navigation autonome en environnement dynamique** à l'aide de capteurs multiples et de l'IA est un domaine de recherche actif et pertinent.

Pour que notre projet apporte une contribution publiable, il pourrait se concentrer sur **une innovation ou une évaluation approfondie** dans l'un de ces domaines. Plusieurs pistes potentielles se dégagent :

- **Intégration de la détection d'objets dans le SLAM (environnements dynamiques)** – Une contribution possible serait de proposer une méthode améliorée de SLAM robustifié aux obstacles mobiles en utilisant YOLO. Par exemple, en détectant les objets dynamiques (véhicules, piétons) via la caméra et en les retirant des données LiDAR utilisées pour la cartographie, on peut obtenir une carte plus fidèle des éléments statiques. Jeong *et al.* (2023) ont suivi cette approche en utilisant YOLOv4 pour détecter et supprimer les véhicules dans le processus de cartographie LiDAR, ce qui **augmente la précision et la fiabilité de la carte** en zone urbaine dense [15]. Notre projet pourrait étendre cette idée sur un scénario indoor ou semi-structuré (par exemple un entrepôt ou un campus) avec le robot LIMO, et évaluer quantitativement l'amélioration du SLAM en présence d'obstacles mouvants (baisse de l'erreur de localisation, carte plus propre, etc.). Une telle étude comparative, appuyée par des résultats expérimentaux, aurait tout à fait sa place dans une publication sur la navigation robotique en environnement dynamique.

- **Nouvelle stratégie de fusion de capteurs légère et temps réel** – Le fait de combiner caméra et LiDAR via un EKF pour le suivi d'obstacles est prometteur, mais encore peu exploré sur de petits robots à ressources limitées. L'un des défis identifiés par Xu *et al.* est justement de concilier précision de perception et faible puissance de calcul : les méthodes de pointe en vision (comme les réseaux profonds 3D) utilisées en voitures autonomes sont trop lourdes pour de petits robots [21]. Il faut donc innover avec des algorithmes **légers**. Notre projet pourrait proposer une architecture de fusion originale où, par exemple, les détections YOLO (rapides mais 2D) sont combinées astucieusement avec le nuage de points LiDAR pour suivre les obstacles sans recourir à des réseaux profonds coûteux. L'utilisation d'un **filtre de Kalman multi-objet** pour fusionner la position des obstacles détectés (par caméra et par segmentation du scan LiDAR) et prédire leur trajectoire future pourrait être un angle de contribution. Si nous parvenons à implémenter un tel suivi multi-capteur en ROS2 et à montrer qu'il améliore la sécurité du robot (réduction du risque de collision) tout en restant temps-réel sur l'ordinateur embarqué du LIMO, cela constituerait un résultat publiable. D'ailleurs, Xu *et al.* offrent une base (code open-source LV-DOT) pour comparer notre approche [22]. Une amélioration possible serait d'introduire un **module de classification du mouvement** (statique vs dynamique) dans l'EKF : en identifiant robustement quels obstacles bougent (via l'historique des positions filtrées), on peut adapter la planification en conséquence – c'est un sujet de recherche en lui-même, lié à la notion de *dynamic obstacle avoidance*.

- **Optimisation et évaluation sur ROS 2/robot réel** – Un apport plus pratique mais valorisable serait de documenter *les leçons tirées de l'implémentation sous ROS 2* et sur la plateforme LIMO, en particulier sur l'**optimisation temps réel**. Par exemple, l'inférence d'un modèle YOLOv5-tiny sur

l'Orin Nano du LIMO, couplée à la fréquence du LiDAR et de l'EKF, doit tenir la cadence en temps réel. Il serait pertinent de mesurer l'impact de chaque composant sur la latence du système et de proposer des optimisations (compression de modèle, pipeline multi-thread, QoS DDS de ROS 2 ajustée, etc.). Ce type de retour d'expérience technique peut intéresser la communauté ROS 2. Bien qu'il soit moins théorique, un article de conférence appliqué (par ex. IEEE IROS *Tutorial/App* paper) pourrait accepter une contribution décrivant la **première intégration de ce genre sur LIMO** et les solutions apportées pour atteindre la navigation autonome fluide. Notamment, on pourrait aborder la gestion de la qualité de service DDS pour synchroniser les données caméra/LiDAR dans ROS 2, ou l'utilisation de *computing graph* réparti (ex. faire tourner le réseau neuronal sur un GPU séparé) – autant de détails qui, une fois maîtrisés, constituent un savoir-faire contributif.

En synthèse, le potentiel de contribution du projet se situe dans l'**avancée marginale** qu'il peut apporter par rapport aux travaux existants. Intégrer simplement YOLO, SLAM et A *de manière séquentielle constitue surtout une démonstration d'ingénierie (très utile pédagogiquement, mais pas forcément inédite scientifiquement). En revanche, focaliser une partie du projet sur* « comment rendre la navigation plus intelligente grâce à la vision et à la fusion de capteurs » *ou sur* « comment assurer la navigation autonome en présence d'obstacles dynamiques imprévisibles » *peut aboutir à des résultats originaux. Par exemple, une étude montrant qu'en retirant proactivement les obstacles mobiles de la carte (grâce à YOLO), le robot évite X% de collisions de plus qu'une navigation classique, apporterait un insight nouveau dans la littérature. De même, proposer une approche de fusion LiDAR-caméra innovante améliorant de Y% la précision de localisation ou la réactivité d'évitement constituerait un apport notable. Il faudra bien sûr valider expérimentalement ces contributions (d'abord en simulation Gazebo puis sur le robot réel LIMO, comme prévu) et les comparer à des bases de référence (par ex. navigation ROS2 standard sans fusion avancée). Si les résultats confirment un gain mesurable***** et que la méthode est générale, alors une publication est envisageable.

Enfin, il convient de noter les **limitations potentielles** qui pourraient être explorées et tournées en axes de recherche. L'une d'elles est la contrainte de calcul embarqué : l'utilisation de réseaux profonds comme YOLO en temps réel peut saturer les ressources d'un petit robot, limitant la fréquence de mise à jour. Il a été observé par exemple que YOLOv4, bien qu'efficace, nécessite une puissance de calcul élevée pour du temps réel, ce qui peut poser problème sur des systèmes robotisés standards [23]. Cela ouvre la porte à des contributions sur l'allègement des modèles (quantification, distillation) ou l'adaptation du taux d'inférence en fonction du contexte (par ex. n'activer la détection que quand un obstacle est suspecté via le LiDAR). Une autre limite est la **fiabilité des détections** : les caméras peuvent être aveuglées en basse lumière ou par des conditions visuelles difficiles, ce qui pourrait être compensé en fusionnant avec le LiDAR. On pourrait donc contribuer en proposant un schéma de fusion adaptatif (par exemple, accorder plus de poids au LiDAR la nuit, plus à la caméra le jour, etc.).

En conclusion, le projet a le potentiel d'apporter une contribution scientifique **si** l'on cible un aspect précis à améliorer par rapport à l'état de l'art. Les directions prometteuses incluent la navigation autonome en environnements dynamiques grâce à la vision (domaine où plusieurs publications très récentes indiquent un fort intérêt [19] [20]), la fusion multi-capteurs optimisée pour robots mobiles légers, ou encore l'optimisation des pipelines ROS 2 en conditions réelles. En articulant bien notre travail autour d'une problématique de recherche actuelle (par exemple *« comment combiner efficacement un détecteur d'objets appris et un SLAM LiDAR pour naviguer en présence d'obstacles mobiles »*), et en démontrant clairement les bénéfices de notre solution par rapport aux approches existantes, nous maximiserons les chances que ce projet débouche sur une **publication** dans une conférence ou revue spécialisée en robotique autonome. Les premiers résultats attendus (simulation Gazebo puis tests sur LIMO) serviront de preuve de concept, qu'il faudra ensuite formaliser et comparer aux méthodes concurrentes pour prétendre à une contribution originale dans le domaine.

Sources citées :

- Document de présentation du projet (objectifs et planning fournis)
- Adiuku *et al.* (2024) – *"Mobile Robot Obstacle Detection and Avoidance with NAV-YOLO"*, Int. J. of Mech. Eng. & Robotics Research [19] [24]
- Xu *et al.* (2025) – *"LV-DOT: LiDAR-Visual Dynamic Obstacle Detection and Tracking for Autonomous Navigation"*, prépublication arXiv [20] [21]
- Jeong *et al.* (2023) – *"Enhancing LiDAR Mapping with YOLO-Based Potential Dynamic Object Removal"*, *Sensors*, vol.24 [15] [23]
- Documentation ROS 2 DDS vs ROS 1 (Asia Research News, 2025) [2]
- Wikipedia (algorithmes A* et SLAM) [16] [14]
- Revue de littérature sur la fusion de capteurs en navigation mobile (Dzedzickis *et al.*, 2025) [10] [11]
- Spécifications AgileX LIMO Pro [3]
- Paper NAV-YOLO – détails YOLOv5 et DWA (Adiuku 2024) [4] [17]

---

[1] [4] [5] [6] [7] [17] [19] [24] (PDF) Mobile Robot Obstacle Detection and Avoidance with NAV-YOLO
https://www.researchgate.net/publication/379215317_Mobile_Robot_Obstacle_Detection_and_Avoidance_with_NAV-YOLO

[2] Smarter, More Accurate Robots! ROS 2 Technology Revolutionizes Real-Time Robot Communication | Asia Research News
https://www.asiaresearchnews.com/content/smarter-more-accurate-robots-ros-2-technology-revolutionizes-real-time-robot-communication

[3] LIMO PRO
https://global.agilex.ai/products/limo-pro

[8] [9] [20] [21] [22] LV-DOT: LiDAR-visual dynamic obstacle detection and tracking for autonomous robot navigation
https://arxiv.org/html/2502.20607v1

[10] [11] [12] [13] Sensor-Fusion Based Navigation for Autonomous Mobile Robot - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC11861736/

[14] Simultaneous localization and mapping - Wikipedia
https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

[15] [23] Enhancing LiDAR Mapping with YOLO-Based Potential Dynamic Object Removal in Autonomous Driving
https://www.mdpi.com/1424-8220/24/23/7578

[16] A* search algorithm - Wikipedia
https://en.wikipedia.org/wiki/A*_search_algorithm

[18] Proportional–integral–derivative controller - Wikipedia
https://en.wikipedia.org/wiki/Proportional%E2%80%93integral%E2%80%93derivative_controller