POLYTECH NICE SOPHIA — Robotique, Systèmes autonomes — UNIVERSITÉ CÔTE D'AZUR

# Rapport de projet R&D :
## Robot livreur dans les laboratoires

Ahmed LAKHLIFI
Thomas PRADINAT

## I) INTRODUCTION

Le robot présenté ici a pour mission de transporter des échantillons au sein d'un laboratoire de recherche. Pour remettre le robot dans son contexte, il faut savoir que les laboratoires de recherche en médecine sont composés de plusieurs salles d'étude dans un même bâtiment. Chaque salle est dédiée à un type d'analyse qui lui est propre. En pratique, tous les échantillons devant être analysés dans les différentes salles arrivent au même endroit ; souvent au secrétariat. C'est donc le rôle de chaque chercheur de venir récupérer les échantillons leur correspondant.
Cette organisation coûte du temps et de la concentration aux chercheurs. De plus, il n'y a pas de traçabilité des échantillons, ils peuvent donc être confondus par les chercheurs, perdus, etc. Notre robot intervient pour simplifier cette organisation puisque son rôle est de distribuer les échantillons reçus au secrétariat à travers tout le laboratoire.

## II) CAHIER DES CHARGES

Pour compléter sa mission, le robot doit suivre un cycle de tâches :

- Récupérer des échantillons au laboratoire et les charger pour les transporter.
- Identifier les échantillons pour connaître leur salle de destination.
- Naviguer à travers le bâtiment pour distribuer les échantillons.
- Décharger les échantillons dans les salles.
- Revenir à son point de départ pour prendre de nouveaux échantillons.

Pour simplifier le robot, le laboratoire est considérer comme étant sur un seul étage, de plus aucune porte n'est considérée comme fermée.

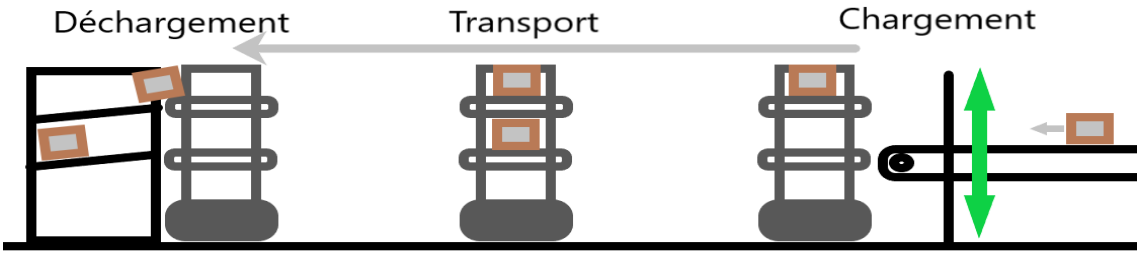
# III) CHARGEMENT ET DÉCHARGEMENT

Il est à noter que cette partie est le sujet de projet d'un autre élève. Le contenu de cette partie est donc purement théorique.

Pour faciliter le transport des échantillons, ceux-ci sont placés dans des boîtes uniformes de taille connue. De plus, pour que le transport soit plus efficace, le robot transporte plusieurs boîtes à la fois. Celles-ci sont stockées sur le robot sur plusieurs étages, ce qui lui donne l'apparence des célèbres robot serveurs.

Le rôle du chargement des boîtes sur le robot revient à une borne de chargement qui est située dans le secrétariat et qui dépose les boîtes sur chaque étage du robot.

Aux différents points de déchargement, se trouvent des meubles adaptés où le robot peut pousser les boîtes depuis n'importe quel étage.




# IV ) DÉTECTION DE CODES-BARRES

### 1 ) Introduction

Dans le domaine moderne des soins de santé et de la logistique pharmaceutique, la détection de codes-barres est un outil essentiel pour garantir l'authenticité des médicaments et une gestion efficace des stocks. Ce projet vise à développer un système robotisé utilisant des techniques de vision par ordinateur pour scanner les codes-barres des médicaments. L'objectif principal est d'intégrer la détection de codes-barres dans une plateforme robotique afin d'automatiser la reconnaissance des échantillons, augmentant ainsi l'efficacité du robot.

### 2 ) Contexte

Les codes-barres, tels que les formats EAN-13 et DataMatrix couramment utilisés, servent d'identifiants uniques pour les produits. Dans le secteur pharmaceutique, ces codes permettent non seulement d'identifier les produits, mais aussi d'encoder des informations essentielles comme les numéros de lot et les dates de péremption, conformément aux exigences réglementaires. Grâce à l'utilisation de techniques avancées d'imagerie et de décodage, les robots peuvent être conçus pour lire rapidement et avec précision ces codes-barres, ce qui les rend particulièrement utiles dans des environnements tels que les hôpitaux, les pharmacies et les usines pharmaceutiques.

### 3 ) Vue d'ensemble du système

Le système robotique développé pour ce projet se compose des éléments clés suivants :

- Module caméra : Le robot est équipé d'une caméra haute résolution qui capture des images en temps réel des emballages de médicaments.
- Logiciel de traitement d'image : Les images capturées sont traitées à l'aide de bibliothèques de vision par ordinateur, principalement OpenCV, afin d'être pré traitées (redimensionnées, filtrées) pour une détection optimale des codes-barres.
- Bibliothèques de décodage de codes-barres : Le système utilise des bibliothèques telles que pyzbar pour le décodage des codes-barres unidimensionnels (EAN-13) et pylibdmtx pour les codes bidimensionnels de type DataMatrix.
- Base de données des médicaments : Une base de données locale au format CSV stocke les informations des médicaments, incluant le code-barres, le nom du produit, le numéro de lot et la date de péremption. Cette base est utilisée pour vérifier l'authenticité des produits scannés.

**4 ) Méthodologie**

- Acquisition et prétraitement des images : La caméra capture des images en direct des emballages. Pour réduire la charge de calcul et améliorer la vitesse de traitement, chaque image est redimensionnée avant d'appliquer les algorithmes de détection de code-barres. Cette étape est cruciale pour minimiser le délai tout en maintenant un niveau de détail suffisant pour un décodage précis.
- Détection et décodage des codes-barres : La détection des codes-barres se fait sur l'image prétraitée via les étapes suivantes :
  - Décodage des codes 1D : La bibliothèque pyzbar est utilisée pour détecter et décoder les codes-barres unidimensionnels comme l'EAN-13.
  - Décodage des codes 2D : La bibliothèque pylibdmtx est utilisée pour décoder les codes DataMatrix, courants dans les emballages pharmaceutiques européens.

  Une fois un code-barres détecté, ses données numériques ou alphanumériques sont extraites.
- Vérification des données : Les données extraites sont comparées à la base de données locale. L'entrée correspondante fournit des détails tels que le nom du médicament, le numéro de lot et la date de péremption. Si une correspondance est trouvée, le système détermine si le médicament est valide ou périmé selon la date actuelle.
- Retour d'information en temps réel : Pour améliorer l'utilisabilité, le système superpose le résultat du décodage et les détails du médicament directement sur la vidéo en direct. Ce retour visuel immédiat permet aux opérateurs de vérifier rapidement le produit scanné et d'agir en conséquence en cas de problème (par exemple, médicament périmé).

**5 ) Implémentation expérimentale**

Les premiers essais du système ont révélé des défis liés au décalage de la caméra et à la vitesse de traitement. Ces problèmes ont été résolus en dissociant l'affichage vidéo du processus de scan des codes-barres grâce à l'utilisation de techniques de multithreading. La solution optimisée permet au robot de maintenir un flux vidéo fluide tout en traitant les images en arrière-plan. Les optimisations clés comprenaient :

- Saut d'images (Frame Skipping) : Traitement d'une image sur n pour équilibrer la performance en temps réel avec la précision de la détection.
- Multithreading : Utilisation d'un thread dédié au scan afin que la détection des codes-barres n'interfère pas avec la boucle principale de capture vidéo.

Ces optimisations ont significativement amélioré la réactivité du système sans compromettre la précision de la détection des codes-barres.

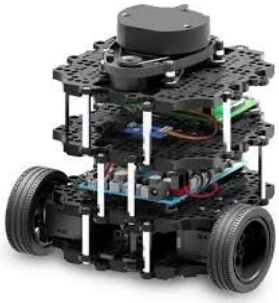
# V ) TRANSPORT DES ÉCHANTILLONS

### 1 ) Rappel des conditions à remplir

Les boîtes doivent être transportées depuis un point de départ unique vers des destinations multiples. Grâce à la lecture de code-barres présentée ci-dessus, la destination de chaque boîte déposée sur le robot est connue. Le robot a pour objectif de rejoindre automatiquement ces différents emplacements dans le bâtiment avant de revenir au point de chargement.
Dans cette partie est détaillée, dans l'ordre, la technique qu'utilise le robot pour se repérer dans le laboratoire, la méthode qu'il utilise pour rejoindre différentes destinations dans celui-ci, et enfin la manière dont il optimise ses trajets dans le bâtiment pour parcourir le moins de distance possible.

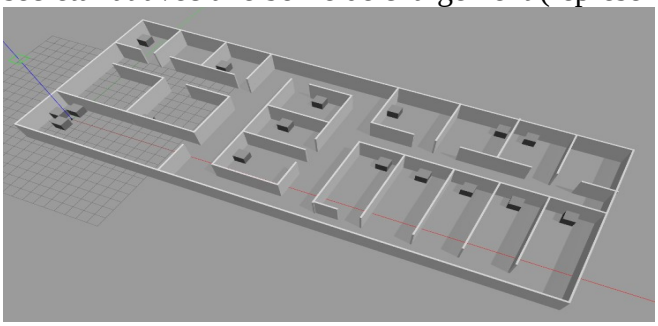
### 2 ) Repérage dans le bâtiment

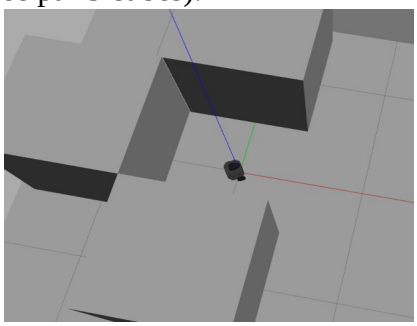
La navigation autonome en intérieur a été très étudiée dans le domaine de la robotique. Parmi toutes les solutions possibles, celle retenue est la méthode du SLAM.
Le SLAM (Simultaneous Localization And Mapping) est une méthode qui permet de cartographier un environnement tout en se déplaçant. Pour cela, les robots utilisent un capteur de distance (lidar, caméra stéréo, etc.) pour percevoir leur environnement, et des roues codeuses pour percevoir leurs propres déplacements. Grâce à la carte alors créée, le robot peut ainsi se repérer dans son environnement et s'y déplacer automatiquement.
Au lieu de créer un robot avec les éléments nécessaires au SLAM, puis d'adapter un code trouvé sur Internet, il a été décidé de reprendre un robot qui est déjà capable de faire du SLAM et de la navigation : il s'agit du TurtleBot.



Le TurtleBot 3 est un robot en kit créé par ROS pour y implémenter facilement de nombreux programmes ROS. Le SLAM et la navigation en font partie. Ainsi, un TurtleBot 3, format Burger, a été récupéré pour découvrir le SLAM. Un autre avantage est que le TurtleBot est également conçu pour fonctionner dans des simulations Gazebo. Il est donc possible de créer un environnement de toute pièce pour y faire des tests.

En suivant les guides et tutoriels fournis par ROS, il a été possible d'utiliser le TurtleBot pour cartographier un environnement en 2D puis de s'y diriger, soit en utilisant le robot physique, soit en simulation. En effet, un laboratoire a été simulé sur Gazebo. Il s'agit d'un grand bâtiment avec 17 salles différentes, chacune avec une borne de déchargement (représentées par des cubes), ainsi qu'un secrétariat avec une borne de chargement (représentée par 3 cubes).





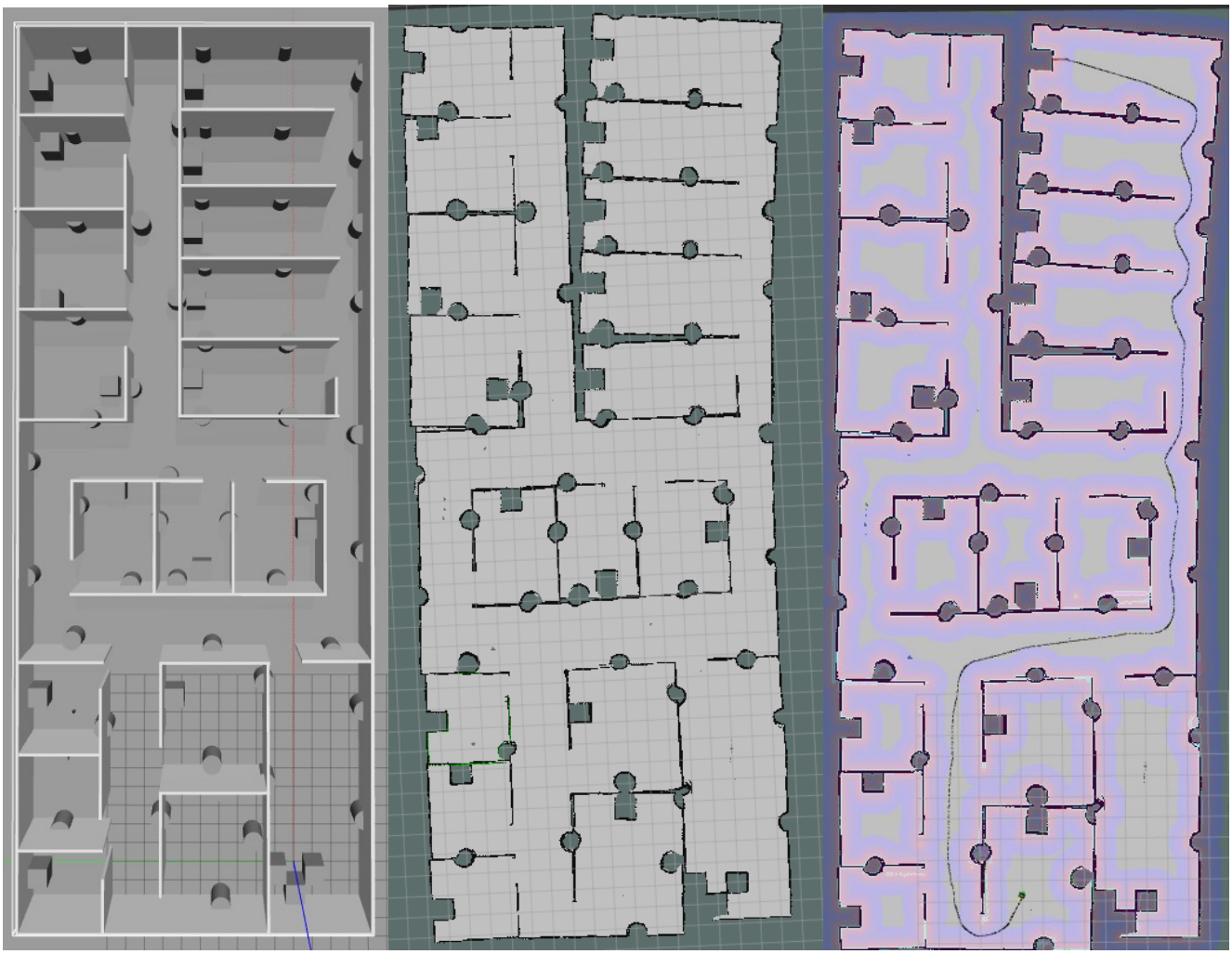
### 3 ) Déplacement vers un objectif

Ce laboratoire a été cartographié sur le logiciel Rviz. Ce dernier permet de créer une carte via le SLAM et de l'utiliser pour la navigation. En effet, il est possible d'utiliser Rviz pour transmettre des instructions au robot via l'interface du logiciel : on peut lui indiquer une position et une direction à atteindre. Ensuite, le programme de navigation utilise la carte pour estimer un chemin à suivre jusqu'à l'objectif, que le robot suit automatiquement. Le programme utilise en continu le lidar pour vérifier la position du robot et la présence d'obstacles imprévus tels que des meubles, des personnes, etc.

La méthode SLAM + Navigation est donc idéale pour que le robot se repère dans le laboratoire, car une fois le bâtiment cartographié, il ne reste qu'à lui indiquer un objectif pour qu'il le rejoigne, malgré la complexité des lieux ou la présence imprévue d'objets ou de personnes sur le chemin.

Sur les images ci-dessous sont montrés (de gauche à droite) : le laboratoire simulé sur Gazebo, la carte que le TurtleBot simulé a générée en le parcourant, et enfin l'utilisation de cette carte pour se déplacer à travers le laboratoire.

Il est possible de noter la présence de cylindres au niveau des murs du bâtiment. Ils ont été placés afin de casser la monotonie des murs et de permettre au SLAM de percevoir l'avancée du robot dans les couloirs. Sinon, le robot ne comprenait pas qu'il avançait lorsque les murs étaient plats de chaque côté.



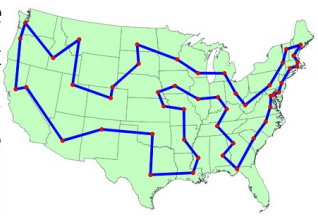
Pour que le robot soit autonome, il ne faut plus que son objectif soit indiqué à travers la fenêtre du logiciel Rviz, mais via un topic des programmes ROS. En effet, le topic, identifié comme porteur des informations d'un objectif sur la carte, s'appelle « move_base_msgs::MoveBaseGoal ». Un publisher vers ce topic permet donc d'indiquer un objectif au robot depuis un programme.

Un programme a ainsi été créé pour que le robot se rende successivement à des positions données au préalable. Les coordonnées de chaque borne sont enregistrées dans un fichier, chacune avec un identifiant. En transmettant au programme une liste d'identifiants, le robot parcourt les points dans l'ordre.

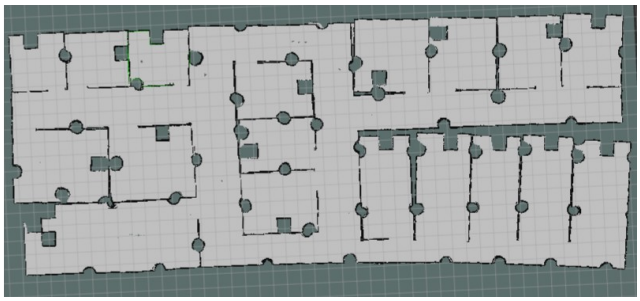
### 4 ) Optimisation du parcoure

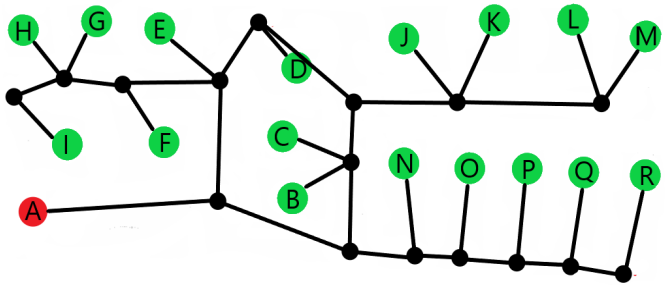
Suite à la réussite de ce programme, la question posée est : « Comment choisir l'ordre des identifiants à transmettre au robot ? » Cette question relève du domaine du pathfinding. En effet, si deux objectifs sont côte à côte, on souhaite que le robot les atteigne l'un après l'autre, et non pas qu'il parte dans une autre zone du bâtiment pour y revenir plus tard. Il faut donc trouver un moyen de déterminer l'ordre optimal des salles à visiter pour que le robot parcoure une distance minimale.

Il s'agit d'un problème bien connu appelé le problème du « voyageur de commerce ». Le voyageur souhaite faire du commerce dans plusieurs villes et cherche l'ordre dans lequel il doit les visiter avant de revenir chez lui, tout en parcourant la plus petite distance possible. Cela revient à trouver la plus petite boucle passant par tous les points.



Ce problème a été résolu grâce à la théorie des graphes. En représentant le laboratoire comme un graphe, on peut le résoudre : chaque borne et chaque carrefour du laboratoire sont considérés comme des sommets du graphe ; et pour chacun sont relevées ses coordonnées ainsi que ses voisins.






L'algorithme du voyageur de commerce fonctionne à partir d'une matrice de distances entre les différents objectifs finaux. Le programme commence donc par calculer les distances les plus courtes entre les objectifs demandés en utilisant la méthode de Dijkstra. Une fois la matrice créée, la méthode du voyageur de commerce renvoie l'ordre optimal des objectifs à parcourir.
Par exemple, si les objectifs demandés en entrée du programme sont : [A, D, E, J, G, C, L, O, H, R, M, Q], le programme retournera la liste : [A, E, G, H, D, J, L, M, C, O, Q, R, A].

Cette liste d'objectifs est ensuite transmise au programme de navigation, qui amène le robot à chacun de ces points, successivement.

# VI ) DÉTECTION DE PERSONNES

### 1 ) Détection Faciale avec Détection des Points de Repère Faciaux

La première tâche que nous avons abordée consistait à mettre en œuvre la détection faciale à l'aide d'une caméra, avec un accent particulier sur la réduction des faux positifs. Pour améliorer la précision de la détection, nous avons adopté la détection des points de repère faciaux, une technique qui identifie les principales caractéristiques du visage telles que les yeux, le nez, la bouche, les sourcils et la mâchoire. Cette approche permet de confirmer si une boîte englobante détectée contient réellement un visage.

Si le détecteur de points de repère échoue à identifier un ensemble cohérent de points, la détection est probablement un faux positif et peut être ignorée en toute sécurité. De plus, les points de repère faciaux permettent d'affiner la boîte englobante du visage en fournissant une géométrie faciale précise. Cela s'avère particulièrement utile dans des conditions difficiles telles que la rotation du visage, l'occlusion partielle ou un mauvais éclairage.

En combinant la détection faciale traditionnelle avec une validation par points de repère, nous avons considérablement amélioré la robustesse et la fiabilité de la chaîne de détection, en nous assurant que seules les visages bien alignés et valides sont traités aux étapes suivantes.

### 2 ) Estimation Approximative de la Distance du Visage

L'objectif de cette partie était de détecter le visage d'une personne et d'estimer approximativement la distance par rapport à la caméra. Le but plus large est de fusionner les données de la caméra et des capteurs LiDAR sur le TurtleBot pour améliorer la précision des tâches de perception.

Initialement, nous avons envisagé d'utiliser une caméra de profondeur, qui capture à la fois des images RGB et des informations de profondeur (c'est-à-dire la distance aux objets de la scène). Cependant, en raison du coût élevé de ce type d'équipement, nous avons choisi d'utiliser une webcam Logitech déjà disponible au laboratoire.

Pour estimer la distance entre la caméra et un visage détecté en utilisant une seule caméra (monoculaire), nous avons mis en œuvre le modèle de caméra à sténopé. Ce modèle relie la taille physique réelle d'un objet à sa taille dans l'image en utilisant le principe des triangles semblables.

Formule d'Estimation de la Distance :

Soient :

- D = Distance entre la caméra et le visage (en cm)
- W = Largeur réelle du visage humain (constante connue, typiquement ~14 cm)
- F = Longueur focale de la caméra (en pixels)
- P = Largeur perçue du visage dans l'image (en pixels)

En utilisant le modèle de sténopé, la formule pour estimer la distance est : $D = \frac{W \times F}{P}$

Calibration de la Longueur Focale :

Pour garantir la précision, nous avons d'abord calibré la longueur focale (F) à l'aide d'une image de référence. Pour cette calibration :

- La distance réelle entre la caméra et le visage ($D_{ref}$) est connue.
- La largeur réelle du visage (W) est connue.
- La largeur du visage dans l'image ($P_{ref}$) est mesurée en pixels.

Avec ces valeurs, nous calculons la longueur focale selon la formule : $F = \frac{P_{ref} \times D_{ref}}{W}$

Cette approche constitue une base pour l'intégration des données visuelles avec d'autres capteurs embarqués sur notre TurtleBot, tels que le LiDAR. Une fois la fusion de capteurs pleinement mise en œuvre, le robot pourra atteindre une meilleure précision et fiabilité dans la détection et l'estimation de la position des humains ou des objets dans son environnement — améliorant ainsi ses capacités globales de perception et de navigation.

# VII ) SUITE DU PROJET

La suite du projet se fera autour de 4 axes très différents :

### 1) Intelligence générale

Maintenant que la caméra est capable de lire des codes-barres, elle peut connaître la destination des échantillons. Ce qui manque encore dans le processus, c'est la transmission des objectifs au robot, afin qu'il puisse ensuite créer son itinéraire grâce au pathfinding.
Il s'agirait, plus globalement, d'un système général qui gérerait simultanément le scan des codes-barres, le chargement du robot, le transport et le déchargement.
Il permettrait également de suivre les échantillons ou de les localiser : savoir si un certain échantillon se trouve au secrétariat, dans une salle d'étude ou en cours d'acheminement, etc.

### 2) Détection de personnes, suite

Une caméra placée sur le robot permet de détecter la présence de personnes sur sa trajectoire et d'estimer leur distance. À partir de ces informations, le robot pourrait anticiper la trajectoire des piétons pour les éviter. En effet, ceux-ci sont plus rapides que le robot et doivent donc être anticipés, à la différence des objets statiques qui peuvent être détectés par le lidar en temps réel.
Il reste donc à créer un programme capable de modifier la trajectoire du robot s'il détecte des humains devant lui.

### 3) Navigation à vue

La navigation par carte est très utile pour tracer des itinéraires dans le laboratoire. En revanche, elle s'avère moins précise lorsqu'il s'agit de se positionner avec exactitude. Le problème se pose notamment lors du chargement et du déchargement du robot, où ce dernier doit être parfaitement positionné par rapport aux bornes.
Dans ce cas, une navigation plus précise serait nécessaire. La piste de la navigation basée sur des repères visuels et une caméra pourrait être envisagée.

### 4) Amélioration mécanique

Pour l'instant, les algorithmes de navigation dans le laboratoire sont implémentés sur le TurtleBot. Malheureusement, le TurtleBot3 Burger utilisé mesure à peine 20 cm de hauteur pour 15 cm

de largeur, et atteint une vitesse maximale de seulement 1 km/h. Ces performances ne sont pas adaptées au robot final : il faut donc revoir les caractéristiques mécaniques.
Pour pouvoir supporter la structure qui gérera le chargement, le stockage et le déchargement des boîtes, le robot devra être plus large et plus solide. C'est donc tout le châssis qui devra être remplacé.
Afin d'augmenter sa vitesse (et son couple moteur, car le robot sera plus lourd), de nouveaux moteurs devront être installés sur les roues, et les codes de gestion des moteurs devront être adaptés en conséquence dans ceux du TurtleBot.

# VIII) CONCLUSION

Le projet sur lequel nous nous sommes engagé nous a amené à travailler dans différents domaines : vision par ordinateur, navigation, pathfinding, base de donnée, etc. Pour la plupart, les réponses aux problèmes que nous avons rencontré existaient déjà et notre travaille consistait à aller chercher ses solutions existantes et les adapter pour notre robot.

L'utilisation du Turtlebot a été notamment très importante dans le projet car elle a permis de simplifier certaines étapes complexes tel que le SLAM et la navigation, tout en étant très ouvert aux modifications grâce à l'environnement ROS. D'un autre côté, la prise en main du Turtlebot a pris beaucoup de temps et a nécessité du matériel adapté.

Avec plus de temps, des expérimentations plus complètes auraient permis de lier les différentes parties du projet sur lesquelles nous avons travaillé.