REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT

SUPERIEUR ET DE LA RECHERCHE

SCIENTIFIQUE

UNIVERSITE AMAR TELIDJI - LAGHOUAT

Faculté de Technologie

Département d'Electronique

MEMOIRE DE MASTER

Présenté par :

Seryah Meriem Hadjer

DOMAINE : Sciences et Techniques

FILIERE : Electronique

OPTION : Electronique des systèmes Embarqués

Thème

Simulation de deux algorithmes : Le suivi d'une ligne sur le sol et la navigation entre deux murs

Jury de soutenance :

DJERFAF FATIMA	**Pr**	**Président**
OUBATI IBRAHIM ELKHALIL	**MAA**	**Examinateur**
FEKNOUS SAFIA	**MAA**	**Rapporteur**

Promotion : Septembre 2021-2022

Résumé

Dans ce projet on a étudié et simulé avec le simulateur CoppeliaSim, deux algorithmes de robotique ; le suivi d'une ligne sur le sol, et le déplacement dans un couloir. Pour chacun de ces algorithmes on présente le principe de contrôle, le programme de simulation et les résultats obtenus. Pour la première application, le robot doit être équipé d'un simple capteur de couleur placé sous le robot pour mesurer le niveau de gris du sol. Dans la deuxième application, le robot est équipé de capteurs de proximité de type laser placés sur les cotés gauche et droit du robot pour mesurer ses distances par rapport aux murs gauche et droite. Le robot utilisera ces mesures pour contrôler ses déplacements pour qu'il reste bien centré au milieu du couloir et qu'il suive les variations de direction des murs.

Mots clés :

Robot mobile – Simulateur CoppeliaSim – Capteurs de vision et de proximité – Suivie de chemin sur le sol – Déplacement entre deux murs

ملخص

في هذا المشروع درسنا وحاكينا باستخدام جهاز محاكاة CoppeliaSim ، وهما خوارزميتان للروبوتات ؛ تتبع خطًا على الأرض ، وتتحرك عبر الرواق. لكل من هذه الخوارزميات ، يتم عرض مبدأ التحكم وبرنامج المحاكاة والنتائج التي تم الحصول عليها. بالنسبة للتطبيق الأول ، يجب أن يكون الروبوت مزودًا بمستشعر لون بسيط يوضع تحت الروبوت لقياس المستوى الرمادي للأرض. في التطبيق الثاني ، تم تجهيز الروبوت بأجهزة استشعار القرب من نوع الليزر الموضوعة على الجانبين الأيسر والأيمن من الروبوت لقياس مسافاته من الجدران اليمنى واليسرى. سيستخدم الروبوت هذه القياسات للتحكم في هذه الحركات بحيث تتمركز جيدًا في منتصف الممر ويتبع تغيرات اتجاه الجدران.

الكلمات المفتاحية :

روبوت متحرك - CoppeliaSim - مستشعرات الرؤية والقرب - تتبع المسار على الأرض - الحركة بين جدارين.

Abstract

In this project we studied and simulated with the CoppeliaSim simulator, two robotics algorithms; following a line on the floor, and moving through a hallway. For each of these algorithms, the control principle, the simulation program and the results obtained are presented. For the first application, the robot must be equipped with a simple color sensor placed under the robot to measure the gray level of the ground. In the second application, the robot is equipped with laser type proximity sensors placed on the left and right sides of the robot to measure its distances from the left and right walls. The robot will use these measurements to control these movements so that it is well centered in the middle of the corridor and follows the direction variations of the walls.

Keywords :

Mobile robot – CoppeliaSim simulator – Vision and proximity sensors – Path tracking on the ground – Movement between two walls.

Remerciements

J’offre ma grande gratitude à Dieu qui m’a aidé à faire ce travail.

J’exprime ma profonde gratitude à mes parents pour leurs encouragements, leurs soutiens et pour les sacrifices qu’ils ont enduré.

Je remercie ma promotrice ***Mme FEKNOUS Safia*** *pour ses efforts qu’elle a déployé, pour m’aider, conseiller, encourager et me corriger.*

Je voudrais remercier les membres de jury d’avoir accepté d’examiner mon travail.

Je remercie aussi tout le corps enseignant dans le département d'électronique qui a contribué à ma formation universitaire.

Sans oublier tous mes amis.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail trouvent ici ma sincère reconnaissance

Sommaire

Chapitre I : Présentation des deux algorithmes étudiés

Chapitre II : Simulateur CoppeliaSim

Chapitre III : Simulation et résultats

Introduction général

Dans ce projet on se propose d'étudier deux algorithmes de base de la robotique mobile parmi d'autres algorithmes de base comme : l'algorithme du suivie d'une trajectoire avec système d'odométrie, l'algorithme de l'insecte, l'algorithme du champ de potentiel, l'algorithme de la navigation en utilisant une grille d'occupation, …etc.

Le premier algorithme étudié est celui du suivie d'un chemin tracé sur le sol. Cette application se base sur un simple capteur de couleur. Elle peut être utilisée dans n'importe quel domaine où un robot mobile pourra aider à transporter des objets d'un endroit à l'autre en suivant toujours le même chemin. Ce chemin pourra être indiqué au robot mobile par une simple ligne tracé sur le sol qu'il détecte avec son capteur de couleur.

Le deuxième algorithme étudié est la navigation dans un couloir (navigation entre deux murs). Dans cette application, le chemin que le robot doit suivre est décrit par le couloir. Donc le robot doit être bien centré dans le couloir et suivre les variations de direction des murs du couloir. Aussi, le robot doit commencer à ralentir avant de s'arrêter complètement quand il aboutit dans le couloir à un chemin fermé. Pour cela, le robot utilisera seulement des capteurs de proximité de chaque coté.

Pour chacun de ces algorithmes, on expliquera dans ce mémoire la technique de contrôle que l'on simulera par la suite avec le simulateur CoppeliaSim, et on montrera les résultats obtenus.

Dans le chapitre I on présentera les principes de contrôle pour les deux applications. Dans le chapitre II on présentera le simulateur CoppeliaSim qu'on utilisera pour simuler ces deux algorithmes. Et enfin, au chapitre III on donne les programmes et résultats de simulation.

Chapitre I

Présentation des deux algorithmes étudiés

I.1 Introduction

Dans cette partie on va présenter deux algorithmes de base pour robot mobile indépendamment du logiciel de simulation utilisé. Ces algorithmes sont « Le suivie d'une ligne tracée sur le sol » et « Le déplacement entre deux murs ». On appliquera ces algorithmes à un robot mobile unicycle nommé « Pioneer ».

I.2 Algorithme 1 : Suivi d'une ligne sur le sol en utilisant un capteur de couleur

I.2.1 Problématique

L'objectif de cet algorithme est que le robot mobile se déplace dans un environnement sans obstacles en suivant une ligne noir de forme quelconque et fermée tracée sur un sol de couleur claire. Pour se faire, le robot doit être équipé d'un capteur de couleur placé sous le robot très proche du sol et dirigé vers le sol.

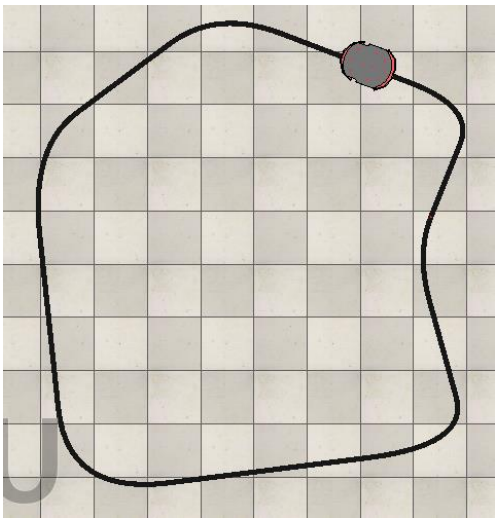


Figure I.1 – Exemple de chemin sur le

I.2.2 Le robot mobile

Le robot mobile utilisé dans nos simulations est un « Pioneer P3-DX ». C'est l'un des robots mobiles fabriqués par « Adept Mobile Robots». Le robot est largement utilisé comme plate-forme pour l'enseignement et la recherche en robotique. C'est un robot unicycle doté de deux roues motrices et d'une roulette libre à l'arrière. Le robot est équipé d'un ensemble de sonars pour détecter les obstacles, des encodeurs de roue pour odométrie et pare-chocs pour détecter les collisions.

Le Pioneer P3 DX est parfaitement capable de cartographier son environnement, de retrouver son chemin et d'effectuer d'autres tâches sophistiquées de planification de trajectoire.

Figure I.2 Robot Pioneer P3 DX

Spécifications techniques du robot P3-DX :

- Terrain traversable : intérieur
- Corps en aluminium laqué de 1,6 mm - Pneus en caoutchouc remplis de mousse
- Batteries 252Wh, 2 moteurs DC avec encodeurs
- Processeur : Microcontrôleur Hitachi HS-8
- Capteurs : Odomètre, 8 capteurs US à l'avant + options (pare-chocs, télémètre laser, gyroscope)
- Max. vitesse : 1,2 m/s (pics jusqu'à 1,6 m/s) - Vitesse de rotation : 300°/s
- Autonomie : 8-10 heures avec 3 batteries (sans charges)
- Dimensions (L x l x h) : 44cm x 38 x 22cm
- Poids : 9 kg (Charge utile maximale : 17 kg)

I.2.3 Modèle mathématique d'un robot mobile unicycle

Le robot unicycle est actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues libres (roulettes) pour assurer sa stabilité. Le schéma d'un robot de type unicycle est donné sur la figure I.3.

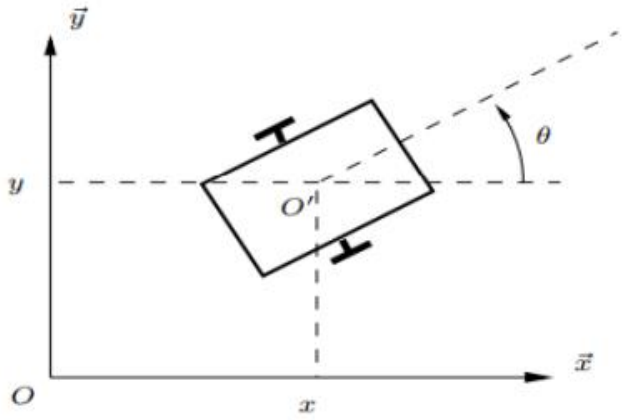
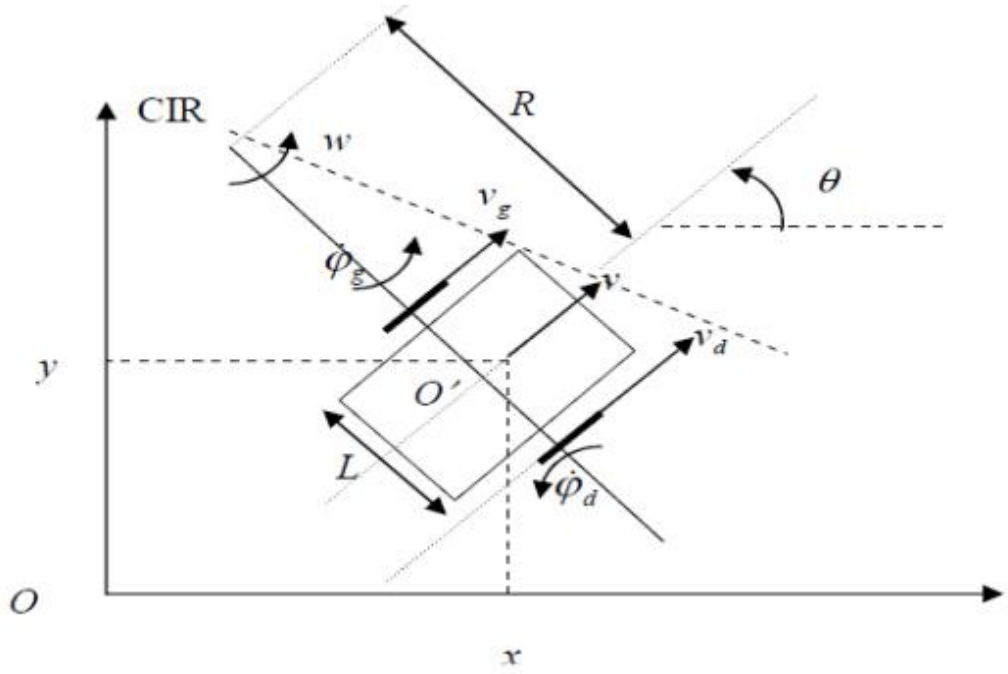


Figure I.3 Représentation de la posture du robot (Position et orientation) dans le système de coordonnées ($O, , y , z$)

Les paramètres du modèle sont :

- **r** : Le rayon des roues motrices
- L : La distance entre les deux roues.
- Le point **CIR** : le Centre Instantané de Rotation
- R : Le rayon de courbure de la trajectoire du robot.
- ω : La vitesse de rotation du robot autour du CIR
- vd et vg : Respectivement les vitesses linéaires des roues droite et gauche
- ωd et ωg : Respectivement les vitesses de rotation des roues droite et gauche.
- (x, y) et θ : La position et l'orientation du robot dans le système de coordonnées (O, x, , z). On appelle (x, y, θ) la posture du robot.



Les vitesses linéaires des roues droite et gauche sont calculées avec les équations (I.1) et (I.2).

$$vd = (R + L/2)\,\omega = r\omega d \qquad \textbf{(I.1)}$$

$$vg = (R - L/2)\,\omega = r\omega g \qquad \textbf{(I.2)}$$

$$v = R\omega \qquad \textbf{(I.3)}$$

Des équations **(I.1), (I.2)** et **(I.3)** on déduit la vitesse longitudinale et la vitesse de rotation du robot (v et ω) en fonction des vitesses angulaires des roues droite et gauche.

$$v = \frac{r}{2}(\omega d + \omega g) \qquad \textbf{(I.4)}$$

$$\omega = \frac{r}{l}(\omega d - \omega g) \qquad \textbf{(I.5)}$$

La vitesse de angulaire des roues droite et gauche en fonction de v et ω et les dimensions du robot (r et L)

$$\omega d = \frac{2v + L\omega}{2r} \qquad \textbf{(I.6)}$$

$$\omega g = \frac{2v - L\omega}{2r} \qquad \textbf{(I.7)}$$

La vitesse de rotation du robot est égale à sa vitesse de rotation autour du CIR :

$$\omega = \theta' \qquad \textbf{(I.8)}$$

La relation entre la dérivée de la posture du robot (x,y,θ) et la commande (v,ω) :

$$\begin{cases} x' = v\cos\theta \\ y' = v\sin\theta \\ \theta' = w \end{cases} \qquad \textbf{(I.9)}$$

I.2.4 Le capteur

Le capteur utilisé est un capteur de couleur. Il est composé d'une LED infrarouge à coté d'une photodiode (ou phototransistor). La LED émet la lumière sur le sol, et la photodiode reçoit la lumière IR réfléchie par le sol. L'intensité de la lumière reçue dépend de la couleur en niveau de gris du sol. Si le niveau de gris du sol est blanc, l'intensité de la lumière reçue est maximale. S'il est noir, l'intensité de la lumière reçue est nulle. Pour les autres niveaux de gris, l'intensité est comprise entre le maximum et zéro ; et plus le sol est claire plus l'intensité de la lumière réfléchie est grande. Dans notre simulation on assimile ce capteur à un capteur de vision à un seul pixel. Ce capteur sera placé sous le robot, et bien centré à l'avant du robot.

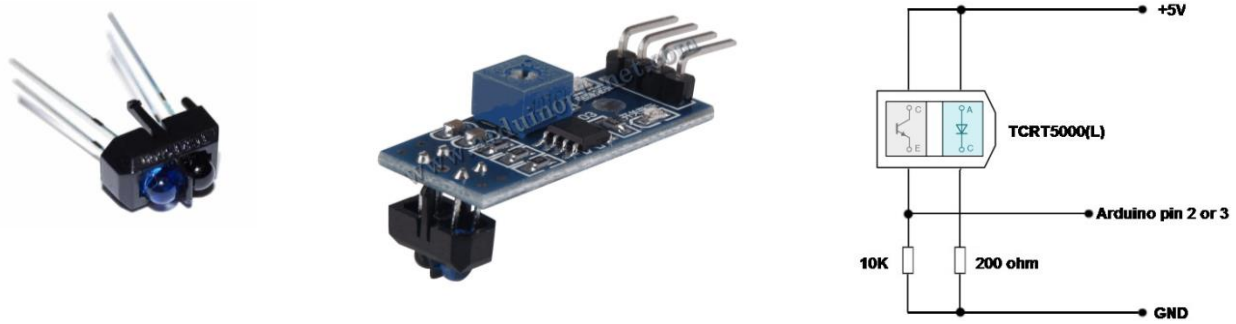


Figure I.5 Capteur de couleur IR (à gauche) – Carte électronique du capteur de couleur (à droite)

I.2.5 La technique de contrôle du robot

Pour se déplacer en suivant la ligne sur le sol, le robot se déplace avec une vitesse de translation V fixe et doit à chaque instant changer son orientation (en réglant convenablement sa vitesse de rotation ω) en fonction de la variation de la direction du chemin au lieu où se trouve le robot.

On caractérise la position du robot par rapport au chemin par l'image obtenue par son capteur de vision. Pour connaître les variations du chemin c'est le bord du chemin qu'on doit suivre.

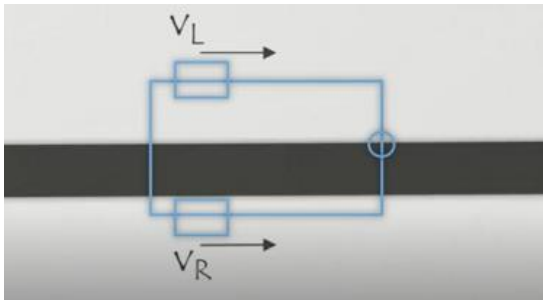


Figure I.6 le robot positionné exactement sur le bor

Lorsque le robot est placé exactement sur le bord du chemin la moyenne des niveaux de gris de l'image est un gris moyen résultant d'une moitié de l'image de couleur noir (couleur du chemin) et une moitié blanche (couleur du sol).

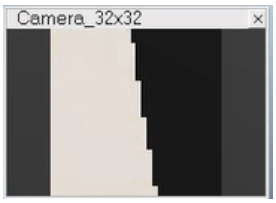


Figure I.7 Exemple d'image sur le bord du chemin capturée avec un capteur de vision 32x32 à la place du capteur de couleur

On peut donc estimer la moyenne des intensités des pixels de l'image par le calcul suivant :

$$I = \left(I_{parquet} - I_{chemin}\right)/2 \qquad \textbf{(I.10)}$$

Pour les valeurs des niveaux de gris on donnera des valeurs normalisées dans l'intervalle [0,1]. On prendra par exemple un sol de couleur blanc cassé et un chemin de couleur noir. Dans ce cas, on supposer que le niveau de gris du sol est d'environ 0.9 et celui du chemin d'environ 0.1. Dans ce cas, notre capteur de couleur va détecter un niveau de gris de 0.5 exactement sur les bords du chemin.

Le contrôle du robot pour le suivie de la ligne tracée sur le sol se fera en fonction du niveau de gris du sol, et la consigne est égale à 0.5 ce qui représente le bord du chemin. D'où le schéma de contrôle suivant :

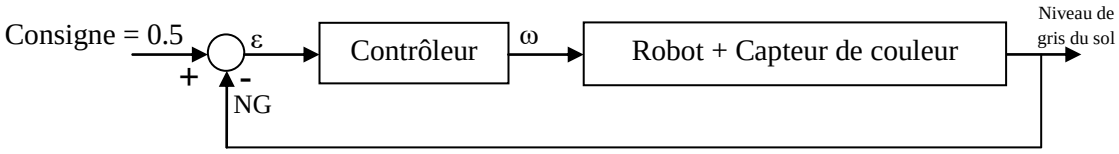


Figure I.8 Schéma de contrôle pour le suivi de ligne sur le sol

Le contrôleur qu'on utilise est de type proportionnel, c'est-à dire que la vitesse de rotation du robot sera proportionnelle à l'erreur. Le sens de rotation du robot dépend du bord de chemin à suivre, le bord de droite ou le bord de gauche. Lorsque l'erreur est négative, c'est-à dire plus de noir que de blanc, le robot doit tourner dans le sens positif ($\omega>0$) pour revenir sur la ligne. Et au contraire, si l'erreur est positive, c'est-à-dire plus de blanc que de noir, le robot doit tourner dans le sens négative ($\omega<0$) pour revenir sur la ligne. Donc la vitesse de rotation doit être inversement proportionnelle à l'erreur. Le contrôleur aura donc pour commande :

$$\omega = -k(NG_{sol} - 0.5) \qquad \textbf{(I.11)}$$

k : Coefficient

NG_{sol} : Niveau de gris du sol

ω : Vitesse de rotation du robot

I.3 Algorithme 2 : Navigation entre deux murs

I.3.1 Problématique

Le but de cet algorithme est que le robot mobile se déplace entre deux murs (un couloir) de forme quelconque. Le robot est équipé de quatre capteurs de distance (capteurs de proximité) de type Lidar placées sur les cotés du robot, deux à droite et deux à gauche et dirigés face aux murs. Le robot utilise aussi un cinquième capteur de type ultrason situé à l'avant du robot pour que celui-ci diminue sa vitesse dans les virages. Et lorsqu'il détecte un obstacle en face de lui, il diminue sa vitesse d'autant plus qu'il s'en approche jusqu'à s'arrêter complètement.

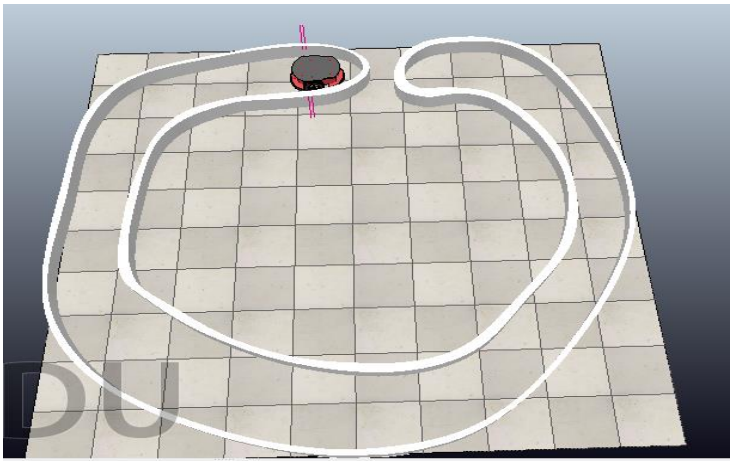


Figure I.9 Exemple de chemin entre deux murs

I.3.2 Les capteurs

I.3.2.1 Les capteurs Ultrason

C'est un dispositif qui utilise les ondes ultrason sonores (supérieures à 20 000 Hz), trop élevées pour être captées par l'oreille humaine, pour mesurer et calculer la distance du capteur à un objet cible spécifié.

Les capteurs ultrason fonctionnent en mesurant le parcours du son entre le détecteur et l'objet cible. Les applications de ces capteurs dans l'industrie sont :

- La détection d'objet mal positionné ou de forme irrégulière.
- Compter des objets difficilement détectables (objets transparents, bouteilles en verre ou en PET).
- Surveiller un niveau de remplissage dans une cuve ou tout autre type de réservoir

En robotique, on utilise le capteur ultrason comme un capteur de proximité qui permet au robot d'éviter des obstacles sur son chemin.

C'est en se basant sur la vitesse du son et de l'écho qu'un capteur ultrason va fonctionner et par conséquent pourra par exemple mesurer une distance. Un émetteur émet des impulsions qui se propagent dans l'air jusqu'à ce qu'elles rencontrent un objet qui les renvoie vers le capteur. la durée du trajet aller-retour des impulsions permet de déterminer la distance. En effet, la vitesse du son dans l'air est constante et ne varie pas, la distance peut donc facilement en être déduite.

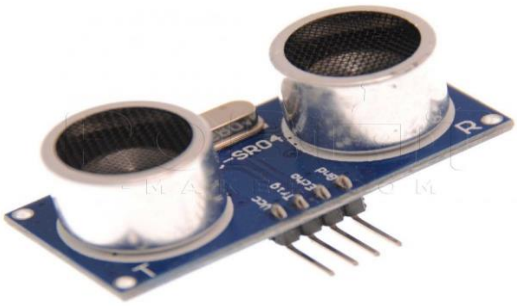


Figure I.10 Carte électronique d'un capteur de proximité ultrason

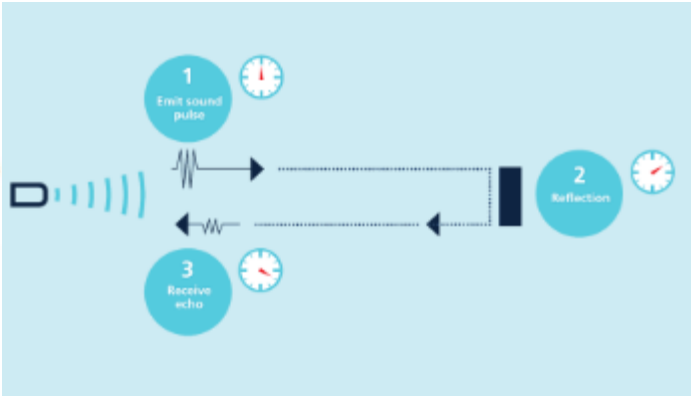


Figure I.11 Principe de fonctionnement d'un capteur de proximité ultrason

I.3.2.2 Les capteurs Lidar

L'acronyme LiDAR signifie (Light Detection And Ranging). C'est un capteur de distance qui utilise un faisceau laser.

Le Lidar est un capteur envoyant rapidement des impulsions de lumière, sous forme de rayons lasers, sans risque pour l'œil humain (Class 1 Eye Safe), sur les objets environnants qui à leur tour les réfléchissent. Ces rayons sont ensuite détecté par le LiDAR . La distance entre le Lidar et l'objet sur lequel il envoie son rayon laser est déterminée par le temps de parcours de la lumière émise entre le capteur et la cible et son retour vers le capteur. La vitesse de la lumière étant une constante, le Lidar fournit ainsi en temps réel une distance précise entre le capteur et un objet.

Un Lidar à haute densité envoie des centaines de milliers d'impulsions par seconde. Toutes ces mesures sont ensuite collectées et traitées pour créer un modèle 3D de l'environnement, appelé un nuage de points. Les Lidars couvrent des portées de quelques mètres à un kilomètre selon les modèles.

En réalité, le capteur mesure le temps entre l'émission de l'onde et son retour au capteur. La distance est déterminée par la formule suivante où V est la vitesse du laser :

$$V \text{ (vitesse)} = D \text{ (distance)} / T \text{ (temps)}$$

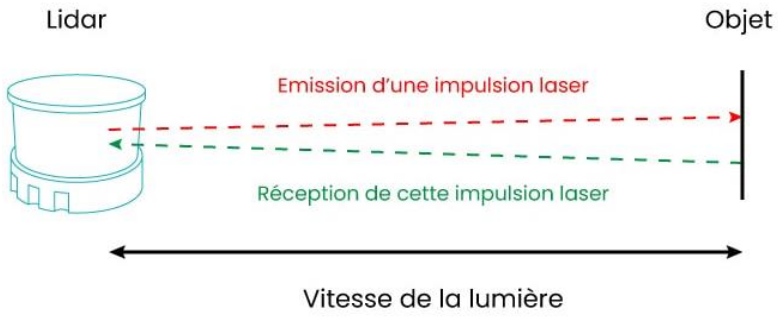


Figure I.12 Principe de fonctionnement d'un Lidar

I.3.3 La technique de contrôle du robot

I.3.3.1 Contrôle de la vitesse linéaire du robot

Dans le cas où le chemin défini par les deux murs est droit (absence de virage) le robot doit avoir une vitesse de translation fixe (Vmax) et une vitesse de rotation nulle (ω=0) tant qu'il ne détecte pas d'objets sur son chemin.

"dmax" est la porté du capteur ultrason. C'est la distance maximale que peut détecter le capteur. "dmin" est la distance minimale que peut détecter le robot. Celui-ci devra diminuer sa vitesse pour éviter frapper l'obstacle et s'appréter à s'arrêter avant de heurter l'obstacle. Lorsque la distance entre le robot et un objet en face de lui est supérieur à "dmax" le robot ne détecte rien et ne fourni aucune distance.

"d" étant la distance mesurée par le capteur. Celle doit est donc inférieure à "dmax".

Lorsque le robot détecte un obstacle en face de lui, il doit diminuer sa vitesse d'autant plus qu'il s'en approche jusqu'à s'arrêter complètement. La vitesse de translation du robot est donc calculée de la manière suivante :

$$V=\begin{cases} vmax\,\frac{d}{dmax} & \begin{cases} si & d > dmin \\ si & d < dmin \end{cases} \\ 0 \end{cases} \qquad \textbf{(I.12)}$$

Pour que le robot ait une vitesse maximale (Vmax) lorsqu'il ne détecte aucun objet en face de lui on met dans la formule (I.12) d=dmax quand on ne détecte aucun obstacle.

Figure I.13 Détection d'un obstacle avec le capteur ultrason se trouvant à l'avant du robot

I.3.3.2 Contrôle de l'orientation du robot

Pour suive le changement de direction des murs, le robot utilise les capteurs de proximité laser (Lidar) placés sur les coté droit et gauche du robot.

En effet, grâce aux mesures des distances entre ces capteurs et les murs qui leur font face (d_{RL}, d_{FL}, d_{RR}, et d_{FR}) (Figure I.13), on peut déterminer les directions des murs gauche et droite φl et φr. « a » est la distance entre deux capteurs situés sur le même coté du robot.

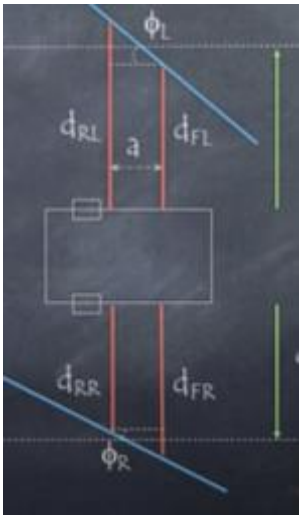


Figure I.14 Représentation des paramètres d'espace qui interviennent dans le calcul des angles de directions des murs

Angle d'orientation du mur gauche par rapport à la direction du robot

$$\varphi l = atan\left(\frac{drl - dfl}{a}\right) \qquad \textbf{(I.13)}$$

La distance moyenne entre les capteurs de proximité se trouvant à gauche et le mur gauche est :

$$dl = \frac{dfl + drl}{2} \qquad \textbf{(I.14)}$$

Angle d'orientation du mur droit par rapport à la direction du robot

$$\varphi r = atan\left(\frac{dfr - drr}{a}\right) \qquad \textbf{(I.15)}$$

La distance moyenne entre les capteurs de proximité se trouvant à droit et le mur droit est :

$$dr = \frac{dfr + drr}{2} \qquad \textbf{(I.16)}$$

Par analogie à la configuration d'un tricycle on ajoute une roue fictive (la troisième roue). Soit **α** la direction de la troisième roue par rapport au robot mobile (Figure I.14)

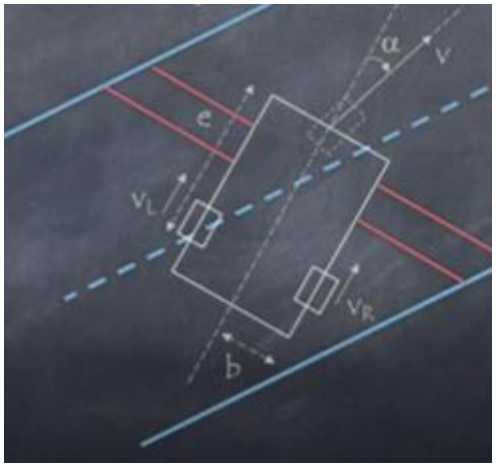


Figure I.15 Direction de la roue libre par rapport au robot

Pour que le robot suive la direction du chemin, la troisième roue doit être alignée avec la direction du chemin. Et donc, la troisième roue doit tourner d'un angle

$$\gamma = \alpha - \varphi \qquad \textbf{(I.17)}$$

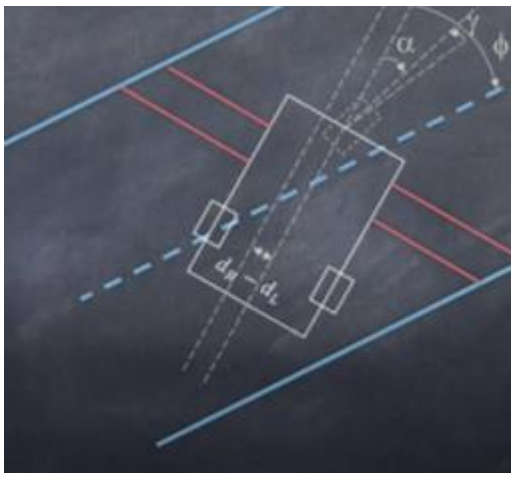


Figure I.15 Angle de rotation de la troisième roue

Le robot n'est placé exactement au milieu entre les deux murs et orienté dans la direction du couloir que si : $\boldsymbol{d_R = d_L}$. Donc on cherchera à ce que la différence $\boldsymbol{d_R - d_L}$ soit nulle.

Pour réguler la position du robot afin qu'il ait la direction et la position désirée, la troisième roues doit tourner d'un angle γ d'autant grand que la différence $\boldsymbol{d_R - d_L}$ est grande.

Lorsque $\boldsymbol{d_R - d_L = 0}$, la troisième roue ne doit plus être tournée**.**

On pose la loi de commande suivante : L'angle de rotation de la troisième roue γ doit être proportionnel à la différence $\boldsymbol{d_R - d_L}$.

$$\gamma = k\,(d_R - d_L) \qquad \textbf{(I.18)}$$

D'après le modèle mathématique d'un robot tricycle, les vitesses linéaires des roues gauche et droite auront les expressions suivantes :

$$vl = v\left(\cos\alpha + \frac{b}{e}\sin\alpha\right) \quad \textbf{(I.19)}$$

$$vr = v\left(\cos\alpha - \frac{b}{e}\sin\alpha\right) \quad \textbf{(I.20)}$$

Où **e** est la distance entre la troisième roue et les deux roues arrières, et **b** la moitié de la distance entre les deux roues arrières (Figure I.14).

D'où, l'expression des vitesses angulaires :

$$\omega l = \frac{vl}{R} \quad \textbf{(I.21)}$$

$$\omega r = \frac{vr}{R} \quad \textbf{(I.22)}$$

I.4 Conclusion

On a d'abord présenté dans ce chapitre le robot Pioneer et le modèle mathématique d'un robot unicycle. Ensuite, on a présenté la problématique, les capteurs utilisés et le principe du contrôle pour les deux applications : Suivi d'une ligne sur le sol et navigation entre deux murs.

Chapitre II

Simulateur CoppeliaSim

II.1 Introduction

Pour simuler le comportement du robot, on a utilisé des outils logiciels qui nous permettent de créer un environnement virtuel qui contient un robot mobile, une ligne noire sur le sol et un couloir et nous permet d'exécuter notre propre programme en langage LUA.

Dans ce chapitre, nous donnons un aperçu sur le logiciel de la simulation CoppeliaSim et quelques unes de ses fonctions de programmation

Figure II.1 CoppeliaSim logo

II.2 Définition

CoppeliaSim est un puissant simulateur de robot multiplateforme doté d'une version gratuite CoppeliaSim edu. CoppeliaSim a évolué à partir de V-REP, qui a été abandonné fin novembre 2019. La force de CoppeliaSim vient de plusieurs fonctionnalités :

- CoppeliaSim fournit un framework qui inclut des moteurs de simulation dynamique, des outils de calcul cinématique directe/inverse, bibliothèques de détection de collision, simulations de capteurs de vision, planification de chemin, outils de développement d'interface graphique et des modèles intégrés de nombreux robots courants
- Vous pouvez intégrer des scripts Lua directement dans une scène de simulation pour le traitement simulé des données de capteur ou des algorithmes de contrôle en cours d'exécution. Une API distante permet de développer des applications autonomes dans de nombreux langages de programmation (C/C++, Python, Java, Lua, Matlab) qui sont capables de transmettre des données dans et hors d'une simulation CoppeliaSim en cours d'exécution

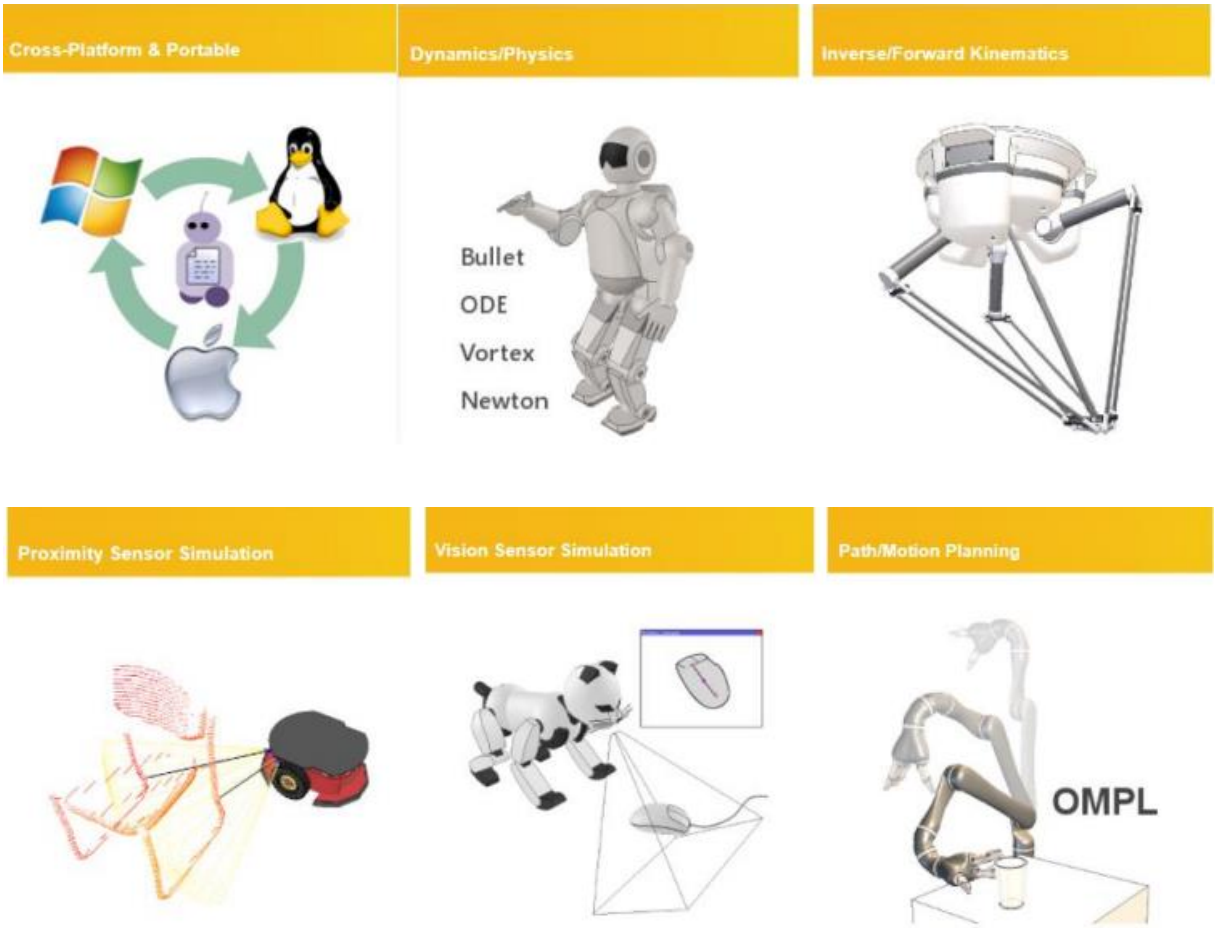


Figure II.2 Illustration de quelques caractéristiques du simulateur CoppeliaSim

II.3 Interface utilisateur graphique de CoppeliaSim

La figure II.3 montre les constituants de l'interface utilisateur de CoppeliaSim.

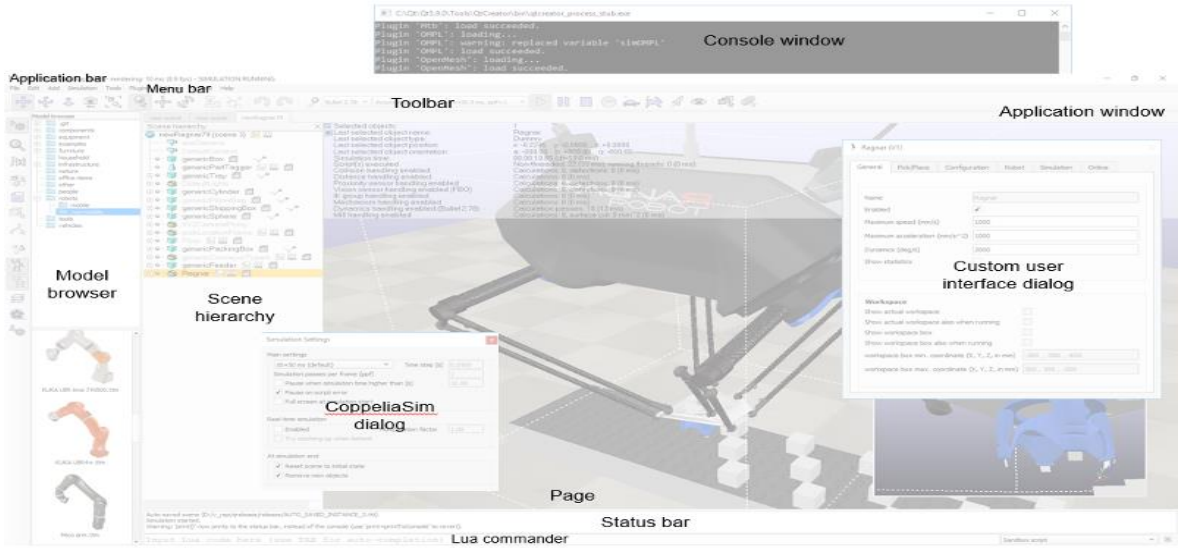


Figure II.3 Interface utilisateur CoppeliaSim

Les principaux éléments de fenêtres de l'interface utilisateur de CoppeliaSim sont :

- Barres d'outils

- Navigateur de modèles

- fenêtre de la hiérarchie de la scènes

- Vue de la scène

Le navigateur de modèles (figure II.3) affiche dans sa partie supérieure une structure de dossier de modèles CoppeliaSim, et dans sa partie inférieure, les vignettes des modèles contenus dans le dossier sélectionné. Les vignettes peuvent être glissées et déposées dans la scène.

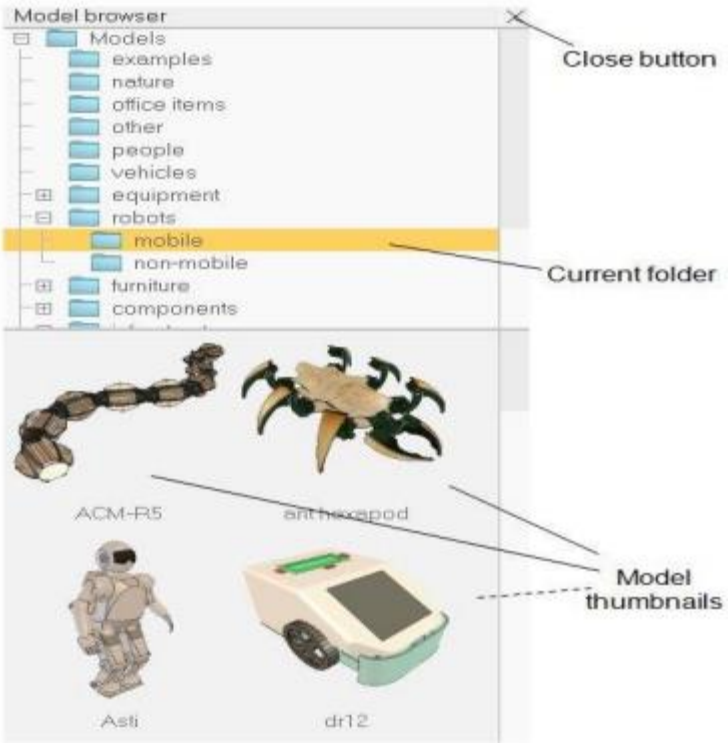


Figure II.4 Navigateur de modèles

Lors de la création d'une nouvelle simulation, la scène contiendra par défaut:

- Plusieurs objets caméra pour avoir des vues de la scène sous différents angles.
- Plusieurs objets lumineux utilisés pour éclairer la scène.
- Plusieurs vues : une vue est associée à une caméra et affiche ce que la caméra voit.
- Le sol.
- Le script principal par défaut : Il permet d'exécuter des simulations minimales, sans avoir besoin de scripts enfants.

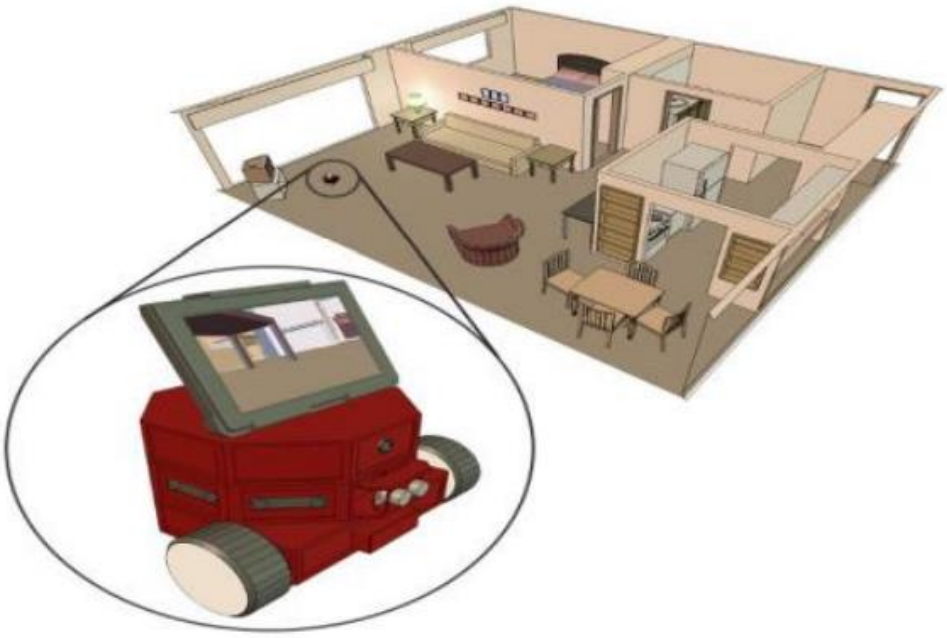


Figure II.5 Exemple de scène CoppeliaSim

La fenêtre hiérarchie de la scènes (figure II.6) affiche les constituants d'une scène. Puisque les objets de la scène sont construits dans une structure de type hiérarchie. La hiérarchie de scène affiche une arborescence de cette hiérarchie. Un objet dans la hiérarchie de la scène peut être glissée et déposée sur un autre objet, afin de créer un relation parent-enfant

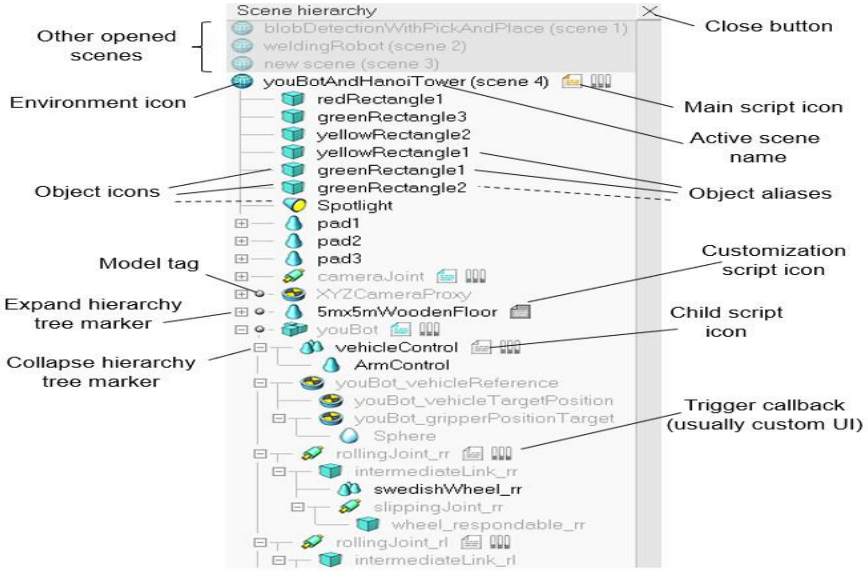


Figure II.6 Fenêtre hiérarchie des scènes

Si l'objet A dans la hiérarchie de la scène est construite au-dessus de l'objet B (figure II.7), l'objet B est le parent, et l'objet A est l'enfant. Ainsi, lorsque l'objet B se déplace, l'objet A suivra automatiquement, puisque l'objet A est attaché à l'objet B.

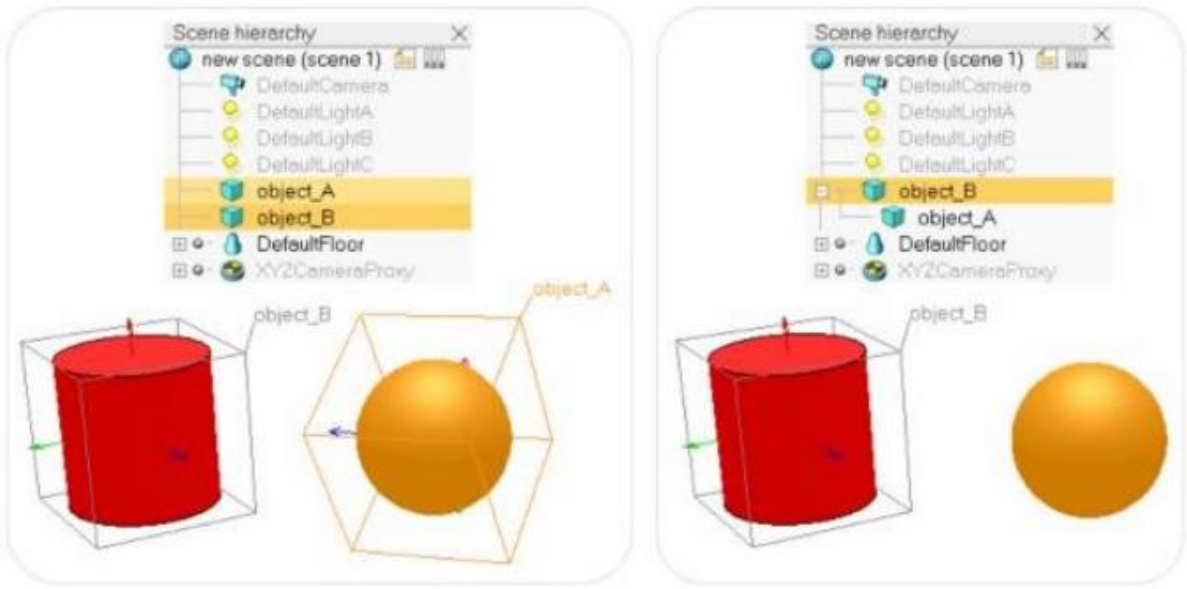


Figure II.7 Relation enfant/parent entre les objets

II.4 Les objets de la scène

Les objets de scène (figure II.8) sont les éléments utilisés pour construire une scène de simulation. Les objets sont visibles dans la hiérarchie de la scène et dans la vue de la scène.

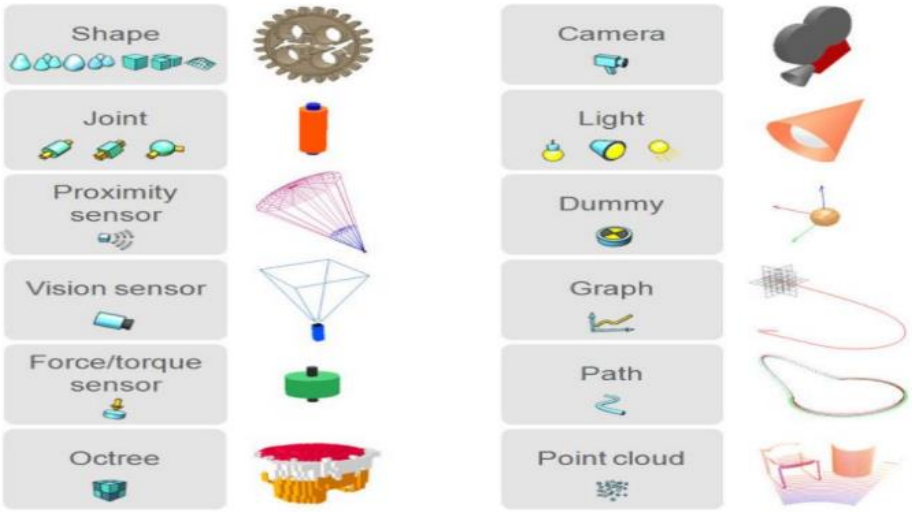


Figure II.8 Objets de la scène

La figure II.9 affiche les 5 formes primitives (plan, disque, cube, sphère et cylindre).

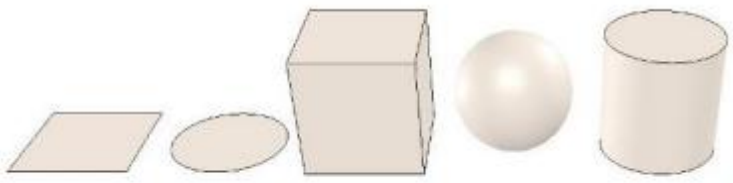


Figure II.8 Formes primitives

Une articulation (joint) est un objet qui possède au moins un degré de liberté intrinsèque (DoF). Les articulations servent à construire des mécanismes et déplacer des objets, les articulations (ou actionneurs) pris en charge sont de quatre types: révolution, prismatique, vis et sphérique.

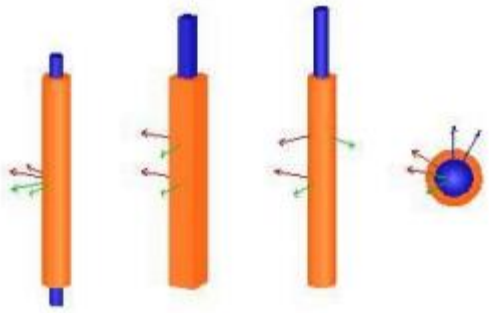


Figure II.9 Respectivement, les articulations rotoïdes, prismatiques, vissés et sphériques

Les capteurs de vision sont de 2 types différents et peuvent être ajustés à différentes fins :

- Type de projection orthographique
- Type de projection en perspective

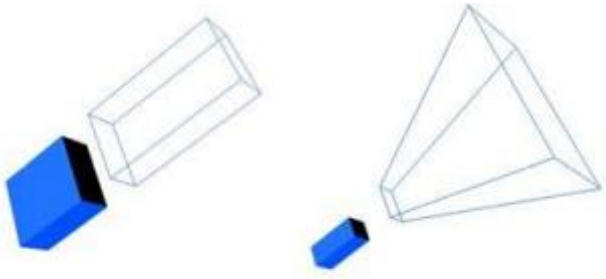


Figure II.10 Capteurs de vision, de type projection orthogonale et de type projection perspective

La boîte de dialogue de la figure II.11 affiche les réglages et les paramètres du capteur de vision. Parmi ces paramètres nous avons l'angle de perspective et la résolution de l'image

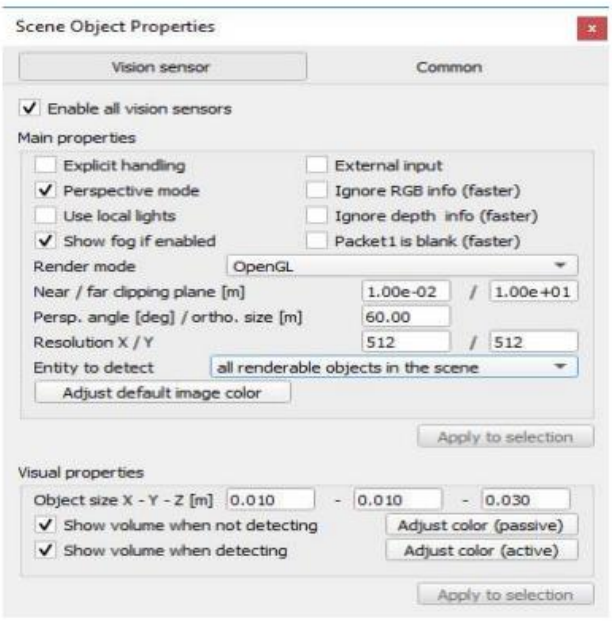


Figure II.11 Paramètres du capteur de vision

Les capteurs de proximité réalise en plus d'une simple détection de type de rayon des détections de type volume configurable qui peuvent être utilisé pour modéliser presque tous les types de capteurs de proximité, des ultrasons aux infrarouges.

Les capteurs de proximité sont disponibles en 6 types différents et peuvent être personnalisés dans une large mesure :

- Type rayon.
- Type de rayon randomisé
- Type pyramidal
- Type cylindre
- Type disque
- Type conique

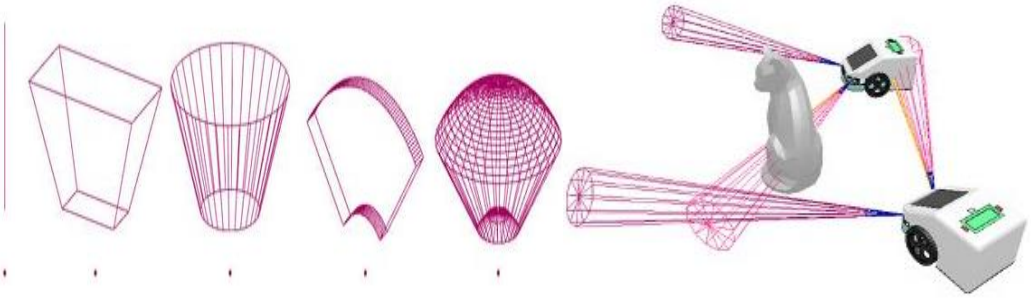


Figure II.12 Capteurs de proximité de type rayon, pyramide, cylindre, disque et rayon conique ou aléatoire

II.5 Programmation avec CoppeliaSim

Le faible encombrement de CoppeliaSim et son API élaborée font de CoppeliaSim un candidat idéal à intégrer dans des applications de haut niveau. L'interpréteur de script Lua ou Python fait de CoppeliaSim une application extrêmement polyvalente, laissant la liberté à l'utilisateur de combiner les fonctionnalités de bas/haut niveau pour obtenir de nouvelles fonctionnalités de haut niveau**.**

Dans notre projet, nous avons utilisé le langage LUA pour écrire le code de simulation. CoppeliaSim fournit plus de 100 fonctions. Nous présentons ici certains d'entre eux, ceux que nous avons utilisés dans notre code LUA.

Fonctions	Utilisation
Sim.getObjectHandle	Récupère un descripteur d'objet en fonction de son nom
Sim.getVisionSensorImage	Récupère l'image RGB (ou une partie de celle-ci) d'un capteur de vision
Sim.setJointTargetVelocity	Définit la vitesse cible intrinsèque d'une articulation non sphérique.
Sim.readProximitySensor	Lit l'état d'un capteur de proximité.

II.6 Conclusion

Dans ce chapitre, nous avons donné un aperçu sur le simulateur CoppeliaSim. On a présenté ses caractéristiques, son interface graphique, ses modèles de robot et d'infrastructures pour construire la scène, un exemple de l'arbre hiérarchique d'une scène, les différents types d'objets, et quelques fonctions de programmations.

Chapitre III

Simulations et résultats

III.1 Introduction

On a fait les simulations des algorithmes 1 et 2 en utilisant CoppeliaSim Version 4.1.0. On présente dans ce chapitre comment on a construit le modèle de la scène et les programmes de simulation pour les deux algorithmes.

III.2 Simulation de l'algorithme 1 : Suivi d'une ligne sur le sol

III.2.1 Construction de la scène

La figure III.1 de la hiérarchie de la scène qu'on a construite est constituée des éléments suivants :

- Le chemin tracé sur le sol (path),
- le robot mobile Pioneer
- Le capteur de couleur (Vision sensor) qui doit être ajouté au robot Pioneer comme child, et qui doit être placé sous le robot pour mesurer le niveau de gris sous le capteur.
- Le graphe pour montrer la trajectoire du robot pendant la simulation

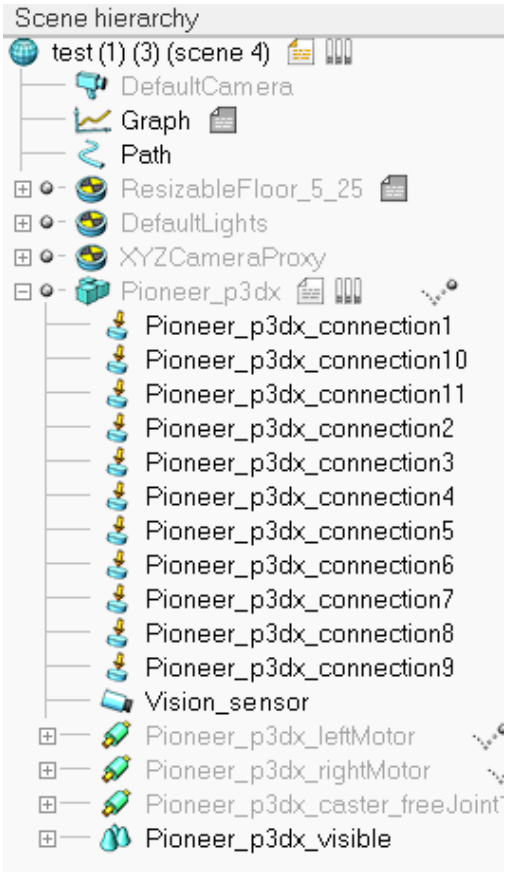


Figure III.1 Fenêtre de la hiérarchie de la scène construite

III.2.1.1 Elément de la scène « Path »

On ajoute l'objet path de la même façon qu'on ajoute tout autre objet comme les formes primitives (primitive shaps) ou les articulations (joints), capteurs de vision (vision sensor), etc. Le path utilisé dans cette simulation doit être de type segment, plat, et fermé de couleur noir assez large.

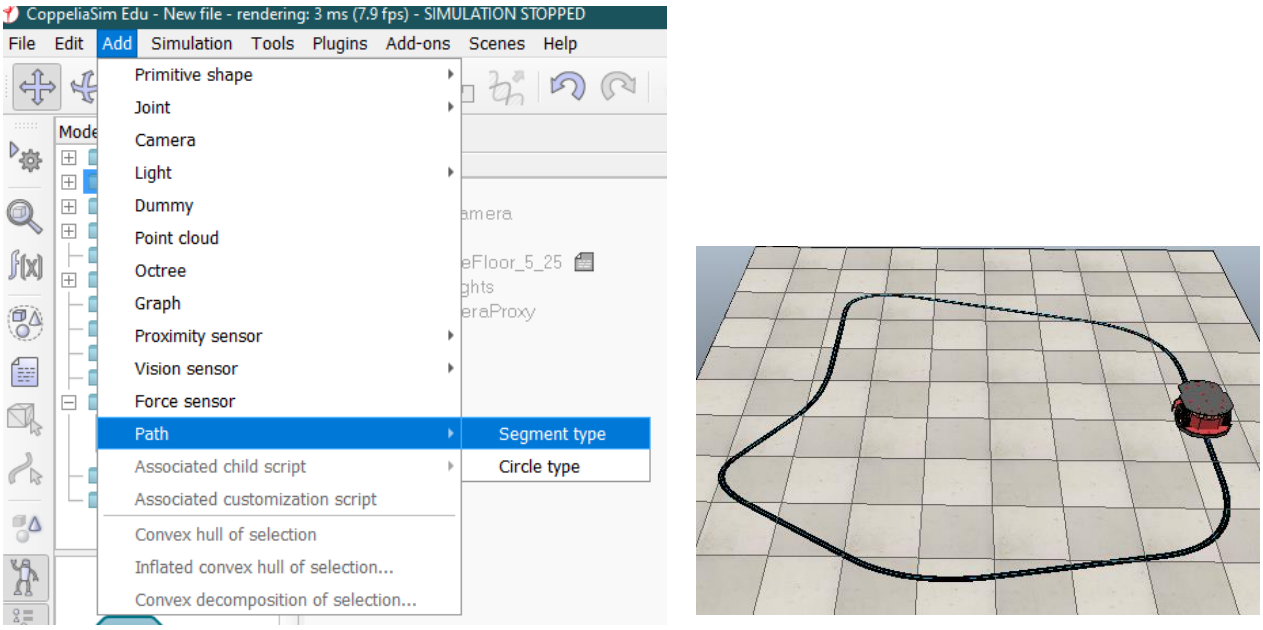


Figure III.2 Ajouter un élément path de type segments dans la scène (à gauche) – Forme du path créé (droite)

III.2.1.2 Elément de la scène « Vision sensor »

Le robot Pioneer n'est pas équipé d'un capteur de couleur pour le suivi de chemin sur le sol. Dans CoppeliaSim le capteur de couleur est un capteur de vision (Vision sensor). On doit l'ajouter dans la scène comme child du robot Pioneer.

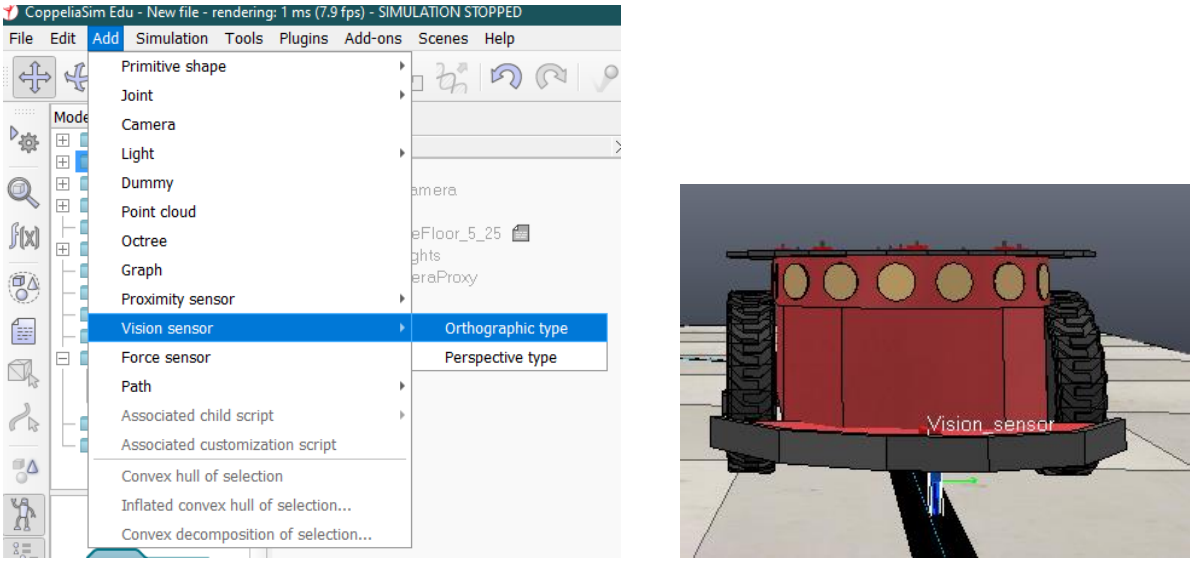


Figure III.3 Ajout d'un capteur de vision orthographique dans la scène (à gauche) – Capteur de vision placé sous le robot (à droite)

On le place sous le robot face au sol et centré à l'avant du robot. Ce capteur de couleur est de type orthographique constitué d'un seul pixel de largeur 5mm. Il détecte les objets à une distance comprise entre 1mm et 1cm.

III.2.1.3 Elément de la scène « graph »

On ajoute aussi dans la scène un graph pour tracer la trajectoire du robot pendant son déplacement. Le graph représente les différentes positions en 3D (X, Y, Z) occupées par le robot (figure III.4)

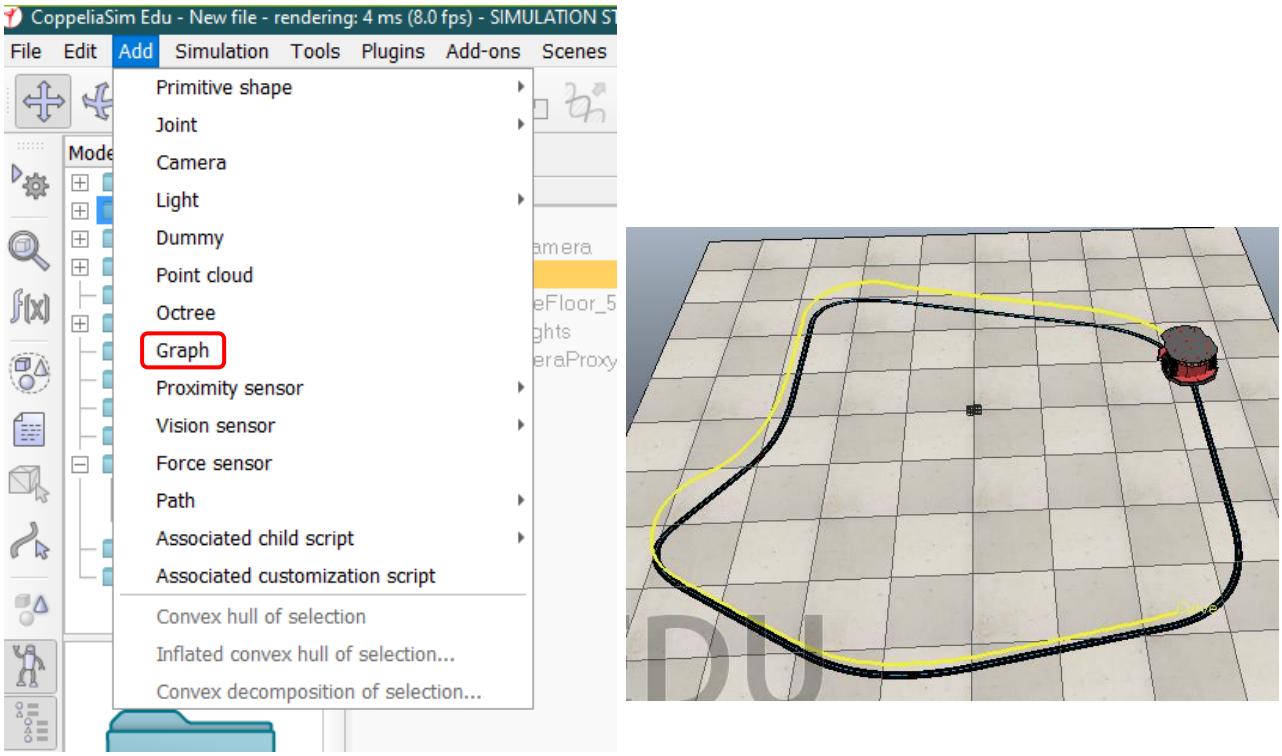


Figure III.4 Ajout d'un graphe dans la scène (à gauche) – Le graphe est représenté dans cette scène pendant l'exécution de la simulation en jaune (à droite)

III.2.2 Programme de simulation

La technique de contrôle utilisé dans cette application a été expliquée au chapitre I. On a écrit le programme en langage LUA de la figure III. conformément à cette technique.
L'organigramme de la figure III. Montre la structure et les étapes du programme.

L'organigramme

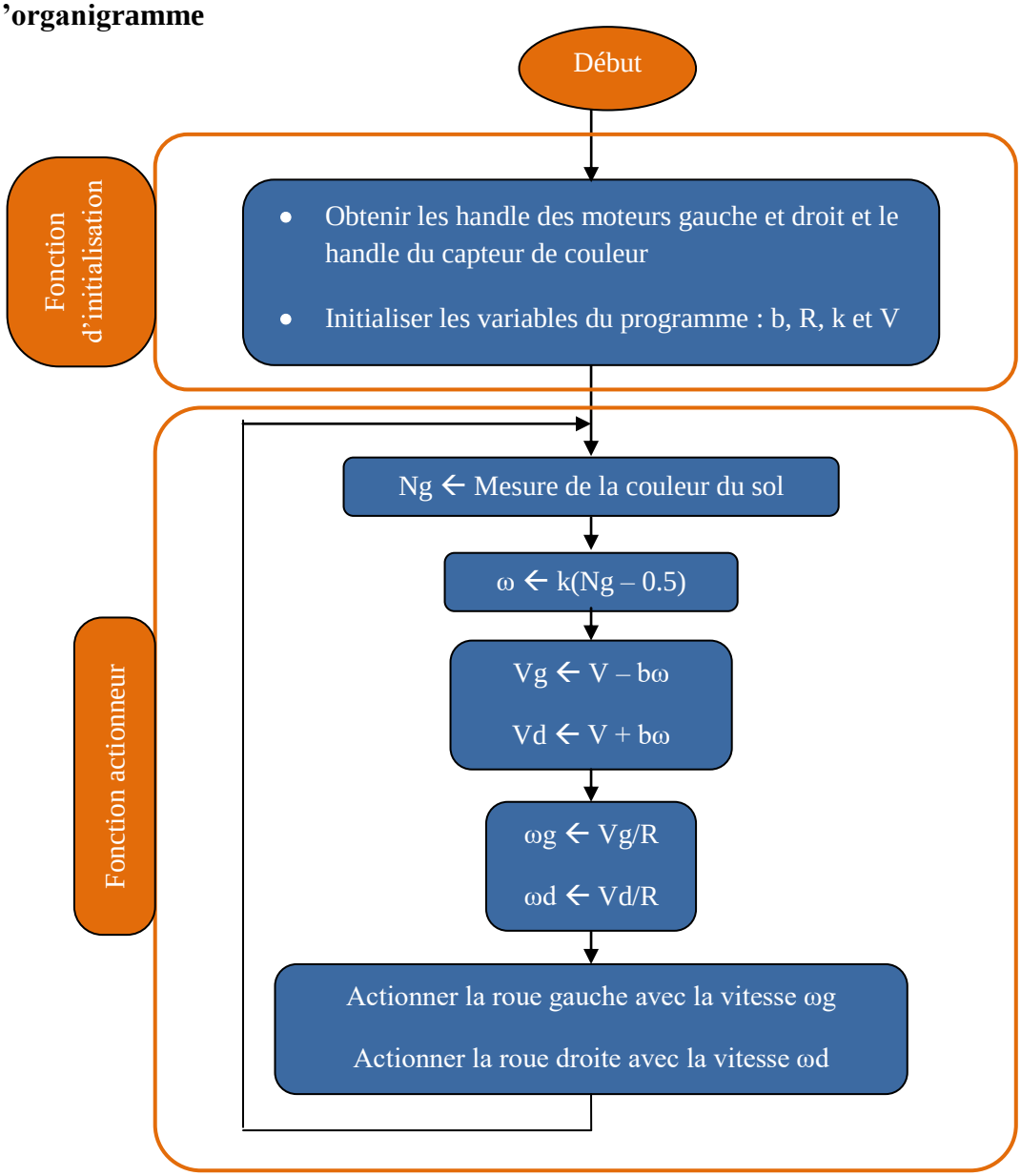


Figure III.5 Organigramme du suivie d'une ligne tracée sur le sol

Ng : Niveau de gris

ω : vitesse de rotation du robot

Vg et **Vd** : Vitesse linéaire des roues gauche et droite

ωg et **ωd** : Vitesses angulaires des roues gauche et droite

R : Rayon des roues

K : facteur

b : Moitié de la distance entre les deux roues

Le programme

```
function sysCall_init()
    motGauche=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
    motDroit=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
    ir=sim.getObjectHandle("IR_sensor")
    v=0.2       --La vitesse linéaire du robot
    k=0.6       --Le facteur de proportionalité du contrôleur
    b=0.33/2    --La moitier de la distance entre les deux roues
    R=0.1       --Rayon des roues du robot
end

function sysCall_actuation()
    --Mesure du niveau de gris du sol
    Ng=sim.getVisionSensorImage(ir+sim.handleflag_greyscale)

    --Calcule de la commande (la vitesse de rotation du robot)
    w=-k*(Ng[1]-0.4)

    --Calcul des vitesses linéaires des roues gauche et droite
    vg=v-(b*w)
    vd=v+(b*w)

    --Calcul des vitesses angulaires des roues gauche et droite
    wg=vg/R
    wd=vd/R

    --Appliquer les vitesses wg et wd aux roues gauche et droite
    sim.setJointTargetVelocity(motGauche,wg)
    sim.setJointTargetVelocity(motDroit,wd)

end
function sysCall_cleanup()

end
```

Figure III.6 Programme de suivie d'une trajectoire tracée sur le sol en langage LUA

On a utilisé dans ce programme en langage LUA la fonction sim.getVisionSensor() de CoppeliaSim. Cette fonction retourne le niveau de gris mesuré par le capteur. Par défaut Par défaut, cette fonction retourne les trois composantes RGB de la couleur mesurée.

Pour mesurer la couleur en niveau de gris on doit ajouter dans la fonction le paramètre « sim.handleflag_greyscale ». Dans les deux cas la fonction retourne un tableau de dimension 3. Pour obtenir l'intensité du pixel en niveau de gris on lit seulement la première valeur du tableau.

III.2.3 Résultats de simulation

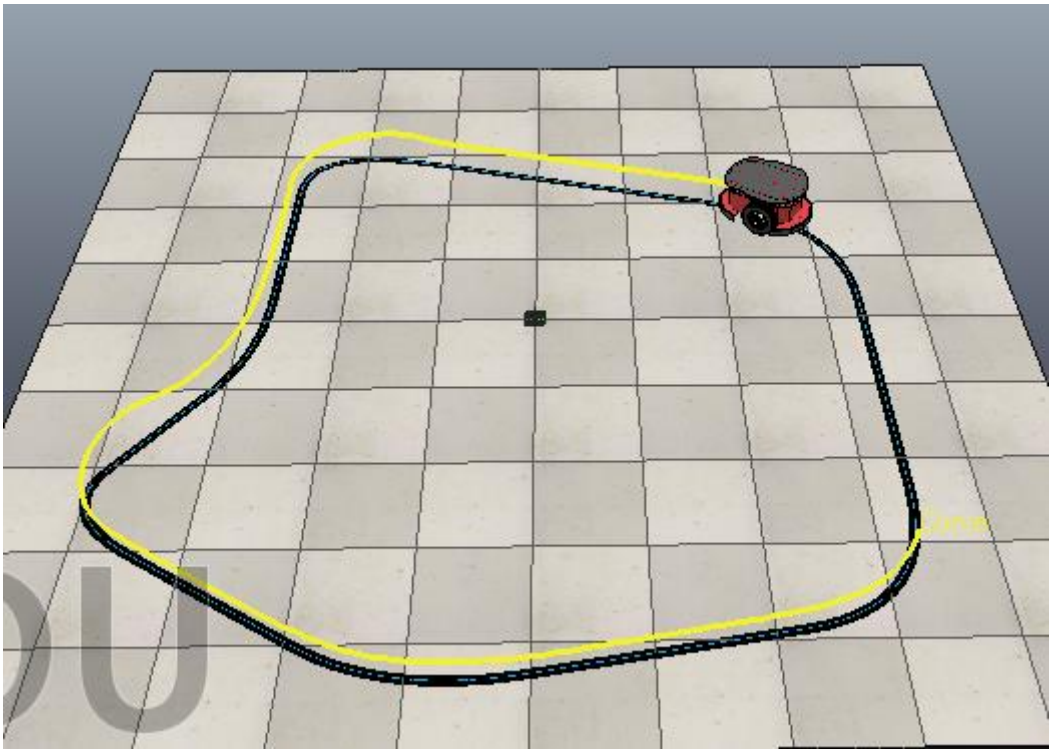


Figure III.7 Résultat de simulation

Le programme a bien fonctionné puisqu'on voit sur le résultat de la figure III.7 que la forme du graphe en jaune qui représente la trajectoire du robot mobile coïncide avec la forme du chemin tracé sur le sol.

III.3 Simulation de l'algorithme 2 : Navigation entre deux murs

III.3.1 Construction de la scène

La figure III.montre la fenêtre « Scene hierarchy » de la scène qu'on a construite. Cette scène est constituée des éléments suivants :

- Le couloir (nommé shape dans la scène)
- Le robot mobile Pioneer
- Le capteur ultrason (Pioneer_p3dx_ultrasonicSensor4) situé à l'avant du robot Pioneer

- Quatre capteurs de Proximité (Lidar) qu'on a ajouté au robot Pioneer comme child (Proximity_sensor, Proximity_sensor0, Proximity_sensor1, Proximity_sensor2)

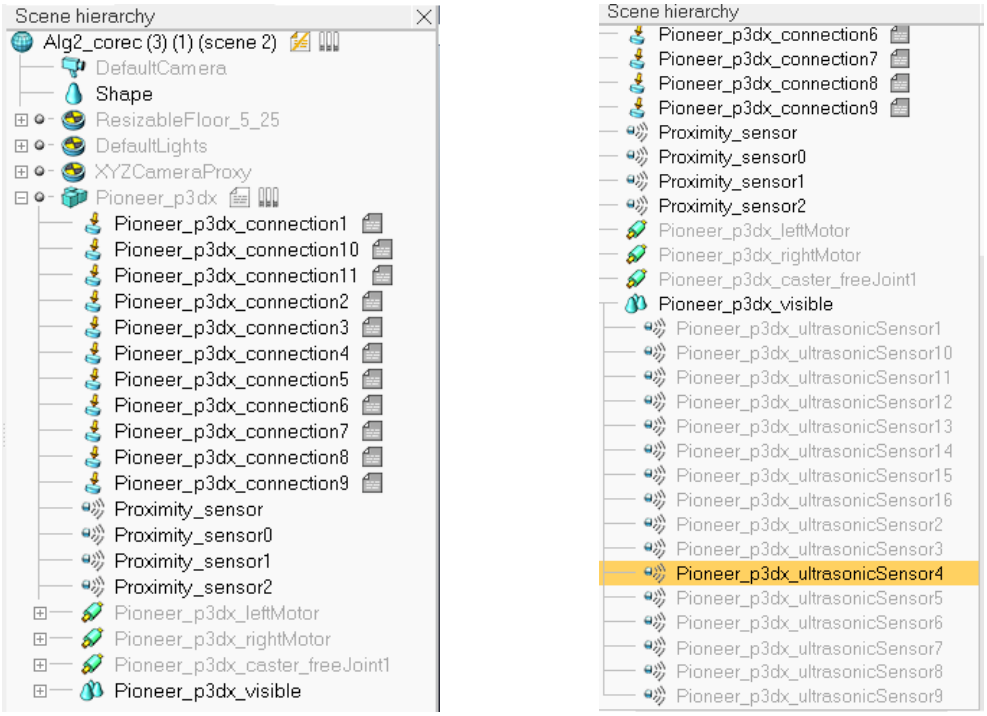


Figure III.8 Fenêtre de la hiérarchie de la scène construite

III.3.1.1 Elément de la scène « Shape » (couloir)

« Shape » (figure III.9) représente un couloir qui fait le tour du parquet (ResizableFloor). Ce couloir se termine par une impasse (chemin fermé).

Le modèle de ce couloir n'existe pas dans CoppeliaSim, on l'a téléchargé d'internet sous forme de fichier stl (stock les informations d'un modèle 3D) et l'a importé un dans CoppeliaSim. On a ajusté les dimensions du couloir pour qu'il soit assez large afin que le robot n'heurte pas les murs, et pas trop large pour les capteurs laser puissent détecter les murs. On a aussi ajusté la hauteur des murs pour que les rayons laser émis par les capteurs puissent toucher les murs.

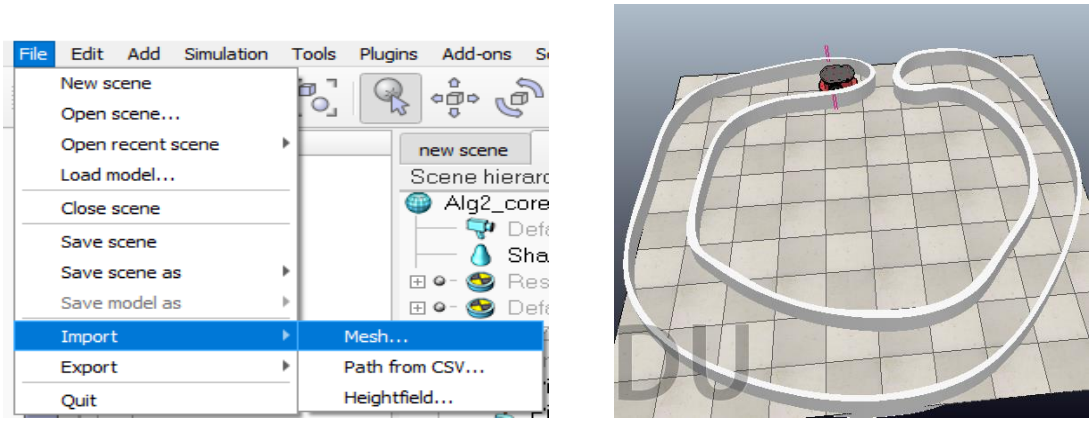


Figure III.9 Importer le modèle du couloir (à gauche) – Le couloir (redimensionné) dans la scène (à droite)

III.3.1.2 Elément de la scène « capteur ultrason »

On a utilisé le capteur ultrason située à l'avant du robot Pioneer pour détecter les obstacles. Ce capteur détecte les objets situés à une distance comprise 0 et 1m. et a un angle de vision de 30°.

Le robot mobile utilise ce capteur ultrason pour détecter les obstacles sur son chemin. Une fois qu'il détecte un obstacle le robot doit diminuer sa vitesse d'autant plus qu'il se rapproche de l'obstacle, jusqu'à s'arrêter complètement.

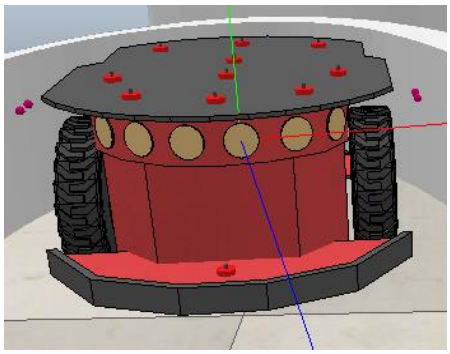


Figure III.10 Le capteur ultrason du robot Pioneer utilisé pour détecter les obstacles se trouvant sur son chemin

III.3.1.3 Elément de la scène « Proximity sensor »

On a utilisé 4 capteurs de Proximité de type ray (rayon) comme capteur LiDAR. Ces capteurs sont placées sur les cotés gauche et droite du robot, deux de chaque coté. La distance entre deux capteurs situés sur le même coté est 3 cm et. Ces capteurs détectent les objets situés à une distance comprise 5cm et 1.2m du capteur.

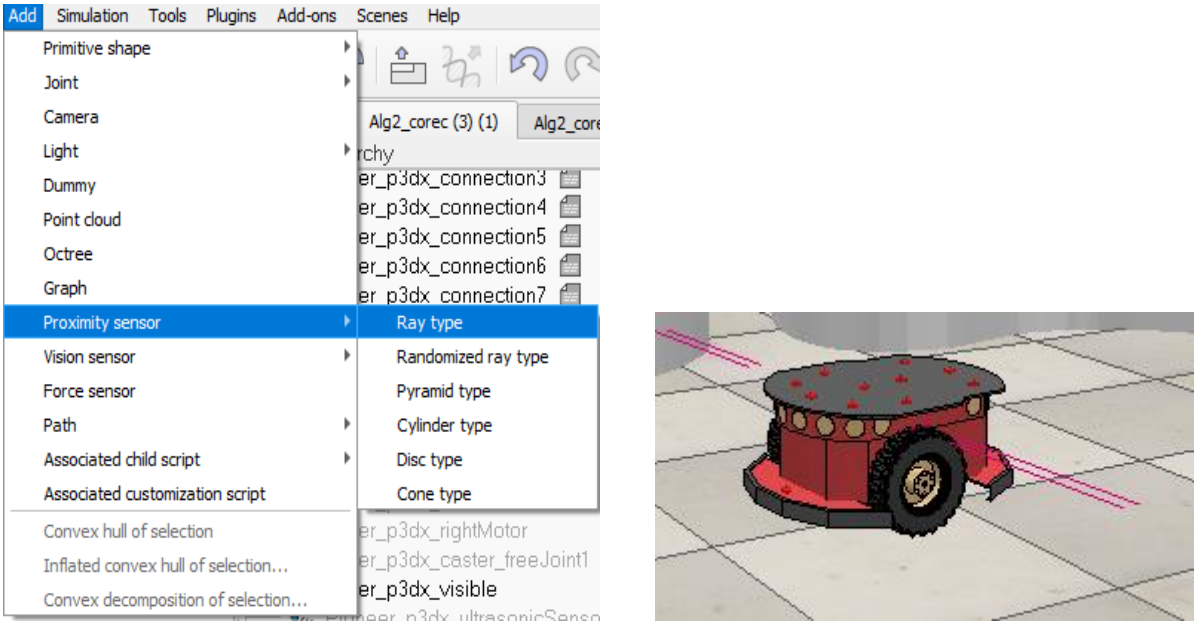


Figure III.11 Capteurs de proximité de type laser (Lidar) ajoutés au robot Pioneer

III.3.2 Programme de simulation

Organigramme

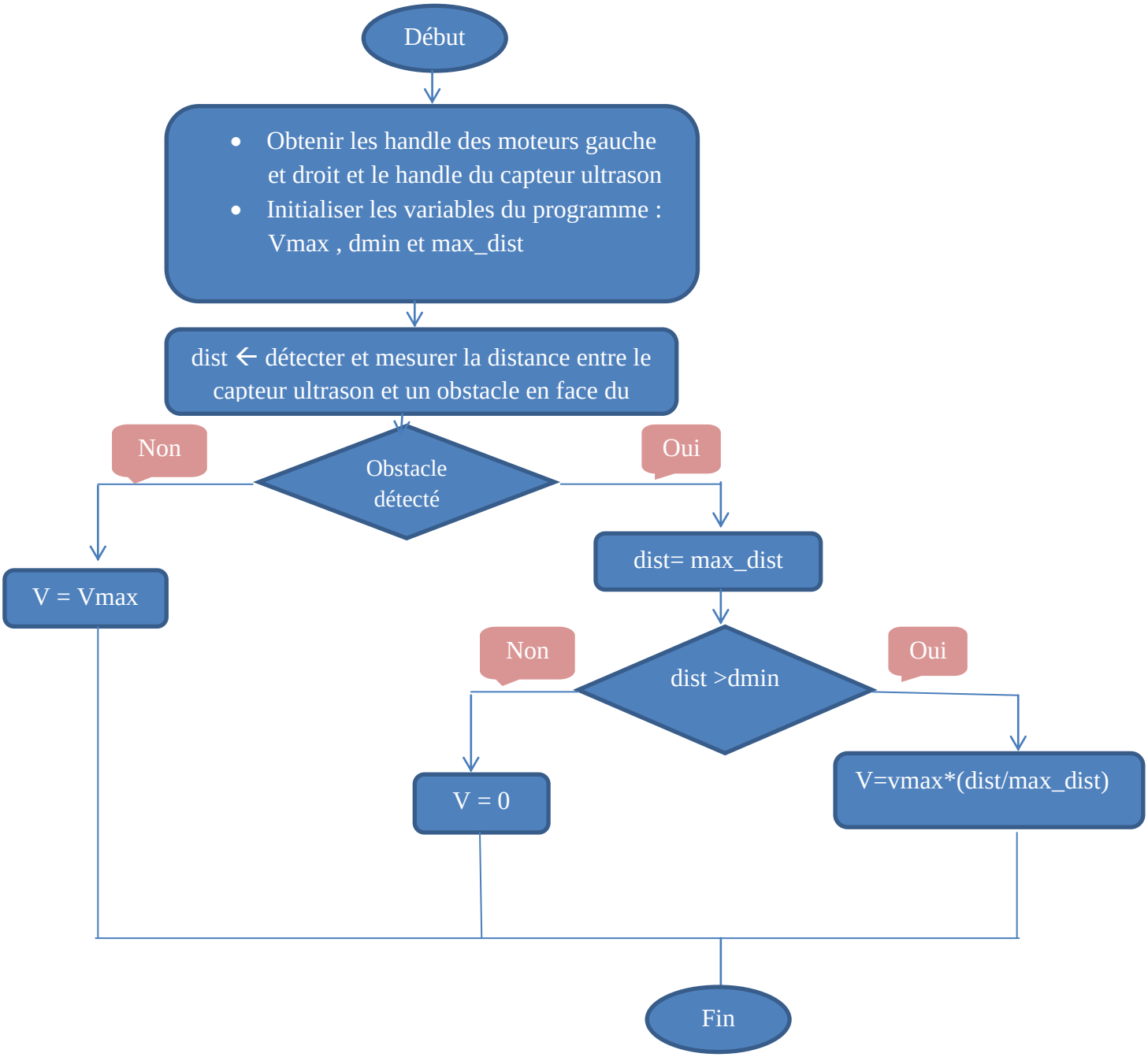


Figure III.13 Organigramme de la navigation d'un robot mobile entre deux murs

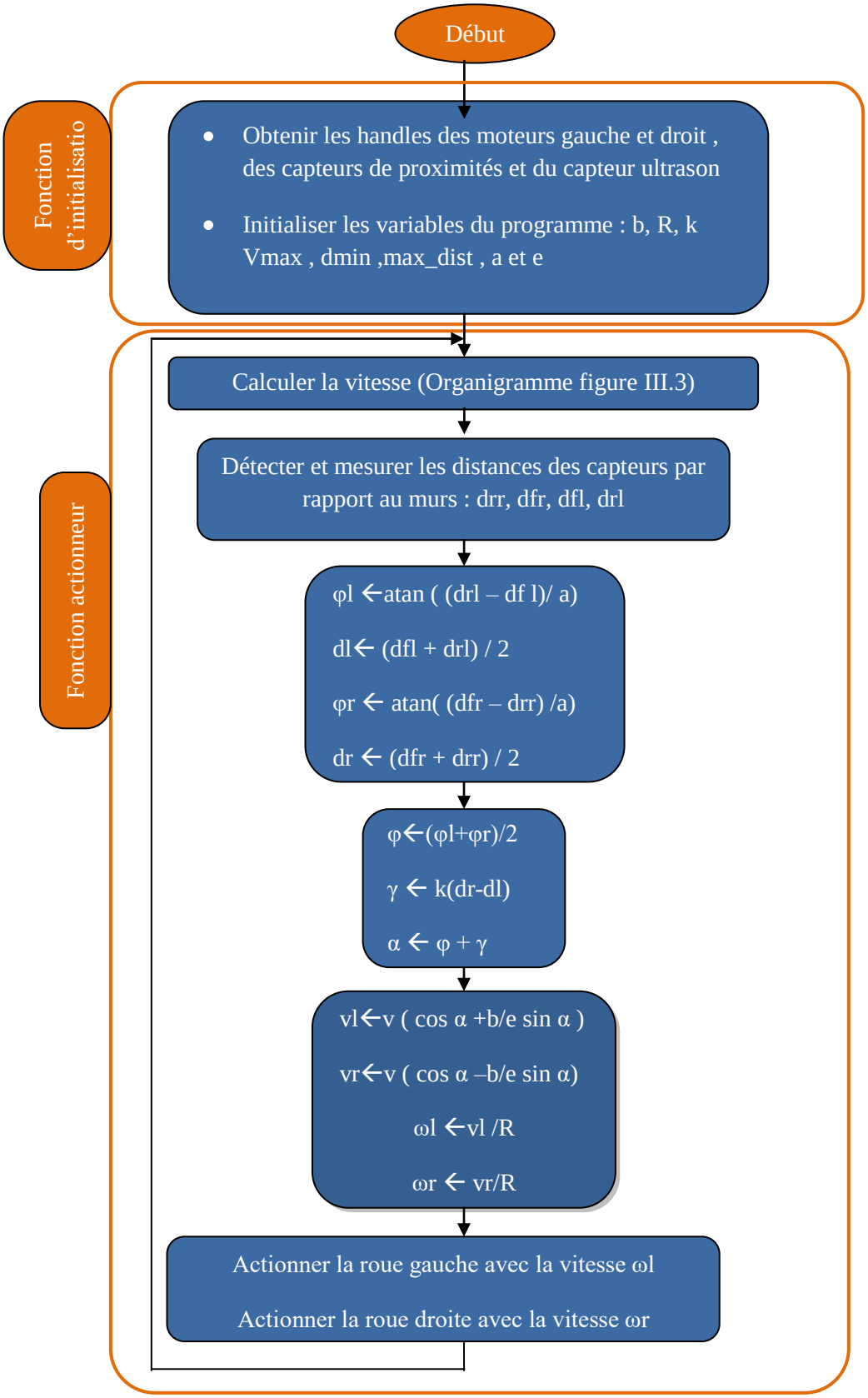


Figure III.12 Organigramme de la navigation d'un robot mobile entre deux murs

φl ,φr = Angle d'orientations des murs droit et gauche par rapport à la direction du robot

dfl, drl, dfr, drr = Distance entre les capteurs de proximité avant et arrière de part et d'autre du robot par rapport aux murs droit et gauche

α= Angle d'orientation de la troisième roue par rapport à la direction du robot

γ = Angle de rotation de la troisième roue

φ = Angle d'orientation du chemin par rapport à la direction du robot

Programme

Non-threaded child script (Pioneer_p3dx)

```
function sysCall_init()
    left_wheel=sim.getObjectHandle('Pioneer_p3dx_leftMotor')
    right_wheel=sim.getObjectHandle('Pioneer_p3dx_rightMotor')
    sensor=sim.getObjectHandle('Pioneer_p3dx_ultrasonicSensor4')
    sensor1=sim.getObjectHandle('Proximity_sensor')
    sensor2=sim.getObjectHandle('Proximity_sensor0')
    sensor3=sim.getObjectHandle('Proximity_sensor1')
    sensor4=sim.getObjectHandle('Proximity_sensor2')

    max_dist=5
    dmin=0.2
    vmax=0.6
    R=0.0975

    b=0.33/2
    e=0.1934
    a=0.03
    k=1.5
end
function sysCall_actuation()

    detected,dist=sim.readProximitySensor(sensor)

    if detected<1
    then
        dist=max_dist
    end
    if dist > dmin
    then
        v=vmax*(dist/max_dist)
    else
        v=0
    end

  det1,drr=sim.readProximitySensor(sensor1)
  det2,dfr=sim.readProximitySensor(sensor2)
  det3,fdl=sim.readProximitySensor(sensor3)
  det4,drl=sim.readProximitySensor(sensor4)

  phiL=math.atan(drl-dfl/a)
  dl=(dfl+drl)/2
  phiR=math.atan(dfr-drr/a)
  dr=(dfr+drr)/2
  phi =(phiL+phiR)/2
  gama=k(dr-dl)
  alpha=gama+phi

    vl=v*(math.cos (alpha) + (b/e) * math.sin (alpha) )
    vr=v*(math.cos (alpha) - (b/e) *math.sin (alpha) )
    wl=vl/R
    wr=vr/R

    sim.setJointTargetVelocity(left_wheel,wl)
    sim.setJointTargetVelocity(right_wheel,wr)
end
```

Figure III.14 Programme de navigation entre deux murs

III.3.3 Résultats de simulation

Pour vérifier le fonctionnement de l'organigramme de la figure III.13 on a fait la simulation suivante :

- Le robot se déplace en ligne droite.
- On place en face de lui, à une distance de 3m un mur.

On peut voir dans la simulation que le robot commence à rouler avec une vitesse fixe et dès qu'il commence à ce rapprocher du mur (à partir de la distance dmin), la vitesse du robot diminue progressivement jusqu'à devenir nul.

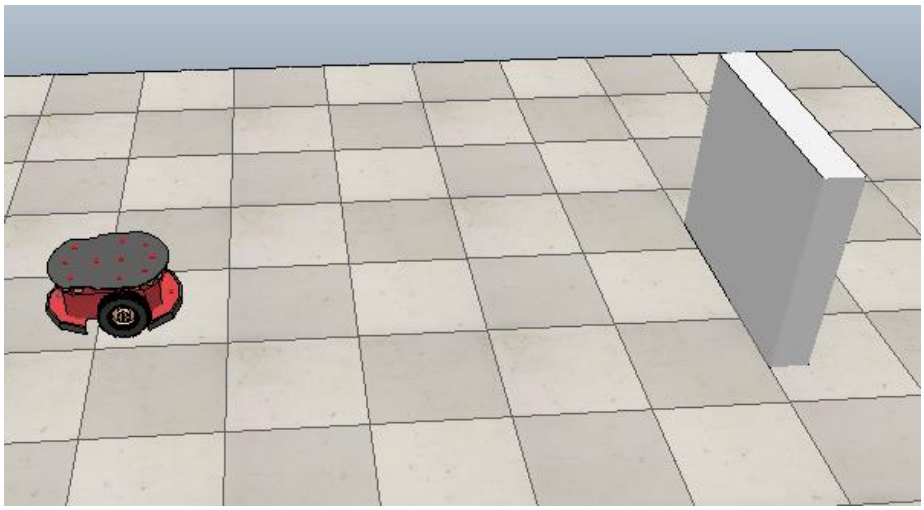


Figure III.15 Résultat pour le contrôle de la vitesse quand le capteur ultrason détecte un obstacle

III.4 Conclusion

Dans ce chapitre, on présente les programmes écrits en langage LUA pour chacune des deux applications étudiées. Ces programmes se basent sur les concepts donnés au chapitre I. On montre aussi les résultats de simulation.

Conclusion général

Ce sujet nous a donné l'occasion d'appliquer deux algorithmes de contrôles de robot mobile « le suivi d'une ligne sur le sol » et « la navigation entre deux murs » avec des outils qui nous ont beaucoup simplifié la tâche : le langage LUA et le simulateur CoppeliaSim. Ce dernier nous a permis de tester ces deux algorithmes dans un environnement virtuel où l'espace, les objets et leur dynamique, et les capteurs sont modélisés. Il ne reste qu'à les réunir et les organiser dans un même environnement, régler leurs paramètres, et ensuite écrire le programme qui fera les calculs et assignera un certain comportement au robot mobile. Et enfin, on assiste à la simulation dans un environnement virtuel presque réel.

Ces deux algorithmes sont une bonne initiation à la robotique qu'on découvre à travers ces deux exemples.

ANNEXE

Langage LUA

Quelques caractéristiques du langage LUA

- Langage interprété basé sur le C
- Type de données : nil, bouléen, nombre, string, tables
- Pas besoin de déclarer les variables dans un programme
- Une fonction peut avoir plusieurs valeurs de retour
- Extensible avec des librairies
- Langage très simple à apprendre et à utiliser

Mots réservés

`and`	`break`	`do`	`else`
`elseif`	`end`	`false`	`for`
`function`	`if`	`in`	`local`
`nil`	`not`	`or`	`repeat`
`return`	`then`	`true`	`until`
`while`			

Remarques

- Dans le langage Lua, contrairement aux autres langages, toutes les variables sont considérées comme **globales**, sauf si elles sont explicitement déclarées en tant que local

Opérateurs logiques

`and`, `or`, et `not`

Valeurs logiques

`true` et `false`

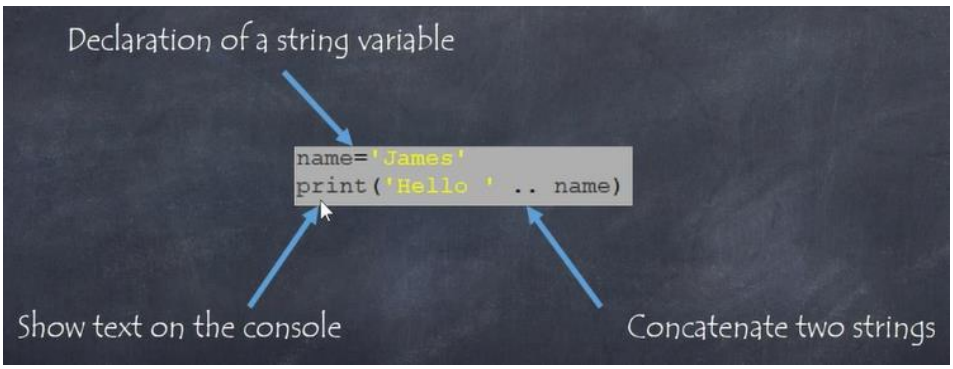
Les autres règles et commandes du langage LUA à travers des exemples

Exemple 1

Dans cet exemple, on a utilisé :

- L'affectation
- Une chaine de caractères

- L'affichage



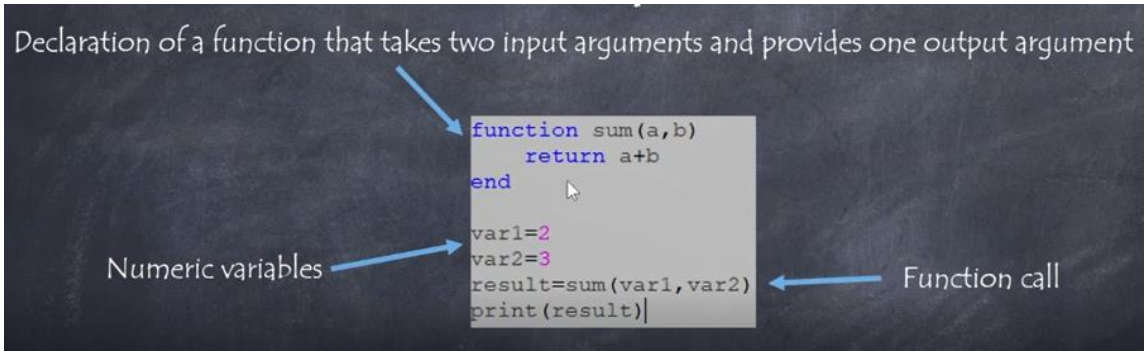
Remarques

- Fonction d'affichage : **`print()`**
- Contrairement aux autres langages, le type caractère n'existe pas. On utilise dans LUA le type général `string`. Une valeur `string` s'écrit entre deux cotes.
- Pas de « ; » à la fin des instructions

Exemple 2

Dans cet exemple, on a utilisé :

- Une fonction avec une seule valeur de retour.



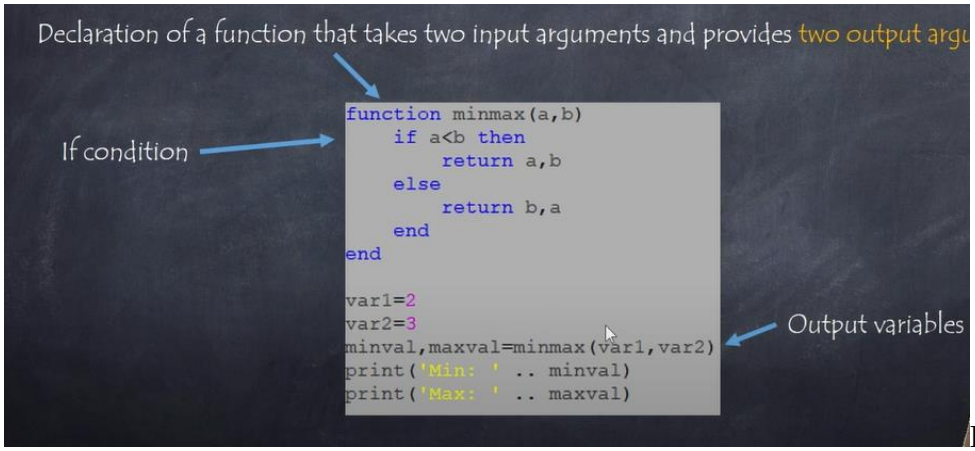
Remarque

- La fonction a la même syntaxe qu'en C sauf qu'elle se termine avec un « `end` »

Exemple 3

Dans cet exemple, on a utilisé :

- Une fonction avec deux valeurs de retour



- La structure de contrôle **`if`**

Remarques

- La syntaxe de `if` :

```
if condition then
    instruction
else
    instruction
end
```

- Syntaxe de return pour deux valeurs de retour :

```
return v1,v2
```

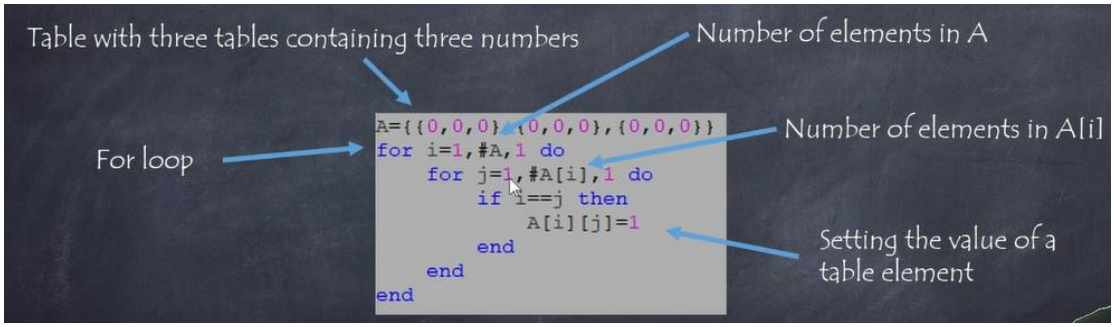
- Syntaxe de l'appel de la fonction :

```
Variable1,Variable2=nom_fonct(val_entrée1,val_entrée2)
```

Exemple 4

Dans cet exemple, on a utilisé :

- Une boucle `for`



Remarque

- Syntaxe de for :

```
for V_compt = val_init,val_fin,pas do
    instruction
end
```

- La première valeur d'une table a l'indice 1.
- Nombre d'élément de la table T : `#T`

Références

[1] Coppeliasim user manual (https://www.coppeliarobotics.com/helpFiles/)

[2] Site internet ''ultrasons'' (https://www.bannerengineering.com/fr/fr/company/expert-insights/ultrasonic-sensors-101.html#/)

[3] Site internet ''Le principe des ultrasons '' (https://www.microsonic.de/fr/support/capteurs-%C3%A0-ultrasons/principe.htm)

[4] Lidar (https://www.cadden.fr/fonctionnement-technologie-lidar/)

[5] Les videos de Leopoldo Armesto en youtube (Programming in Lua | CoppeliaSim (V-REP) (https://www.youtube.com/watch?v=cOk5DuJj-3M&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=16)

[6] Creating Graphs | CoppeliaSim (V-REP) https://www.youtube.com/watch?v=V1X3UJc716s&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=19

[7] Creating paths on the floor | CoppeliaSim (V-REP) https://www.youtube.com/watch?v=P6-qrxsaChg&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=29

[8] How to Use Proximity Sensors To Stay Between Walls | CoppeliaSim (V-REP) https://www.youtube.com/watch?v=hqZ3YRW16KA&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=27

[9] Staying between walls (demo) | CoppeliaSim (V-REP) https://www.youtube.com/watch?v=fI3RbVBqC_8&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=28

[10] How to Use Vision Sensors for Line Tracking with Proximity Sensors| CoppeliaSim (V-REP) https://www.youtube.com/watch?v=vSl1Ga80W7w&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=29