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 :

Boumekouez Nadjat

Lahreche Nour El houda

DOMAINE : Sciences et Techniques

FILIERE : Electronique

OPTION : Electronique des systèmes Embarqués

Thème

Étude et simulation de trois algorithmes de base de robotique mobile appliqués à un robot unicycle

Jury de soutenance :

CHOUIREB FATIMA	**PR**	**Président**
BELLAKEHAL SALIHA	**MAA**	**Examinateur**
FEKNOUS SAFIA	**MAA**	**Rapporteur**

Promotion : Septembre 2022-2023

Résumé

Dans ce projet on a étudié et simulé avec le simulateur CoppeliaSim, trois algorithmes de robotique ; « Suiveur de ligne proportionnel », « déplacement dans un couloir » et « Suivi de ligne avec évitement d'obstacles ». 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 les deuxième et troisième applications, le robot est équipé d'un capteur de proximité ultrasonore pour détecter la fin du couloir ou un obstacle dur son chemin, et des capteurs de proximité de type laser placés sur les cotés gauche et droit du robot pour mesurer ses distances par rapport aux murs. Le robot utilisera ces mesures et les algorithmes de contrôle présentés dans ce mémoire pour réaliser la tâche demandée dans chaque application

Mots clés : Robot mobile unicycle, CoppeliaSim, langage LUA, capteur de couleur, capteur de proximité ultrasonore, capteurs de proximité de type laser

ملخص

في هذا المشروع قمنا بدراسة ومحاكاة ثلاث خوارزميات روبوتية باستخدام برنامج محاكاة CoppeliaSim؛ "متابع الخط المتناسب" و"الحركة في الممر" و"متابعة الخط مع تجنب العوائق". لكل من هذه الخوارزميات نقدم مبدأ التحكم وبرنامج المحاكاة والنتائج التي تم الحصول عليها. بالنسبة للتطبيق الأول، يجب أن يكون الروبوت مزودًا بمستشعر ألوان يوضع أسفل الروبوت لقياس مستوى اللون الرمادي للأرض. وفي التطبيقين الثاني والثالث تم تجهيز الروبوت بمستشعر تقارب فوق صوتي لكشف نهاية الممر أو عائق في طريقه، ومستشعرات تقارب من نوع الليزر توضع على الجانبين الأيسر والأيمن للروبوت لقياس المسافات بين الروبوت و الجدران أو العوائق الموجودة في مساره. سيستخدم الروبوت هذه القياسات وخوارزميات التحكم المقدمة في هذه الأطروحة لتنفيذ المهمة المطلوبة في كل تطبيق.

الكلمات المفتاحية : روبوت unicycle ، CoppeliaSim، لغة برمجة LUA، مستشعر ألوان ,مستشعر مسافة فوق صوتي مستشعرات مسافة من نوع الليزر

Abstract

In this project we studied and simulated with the CoppeliaSim simulator, three robotics algorithms; “Proportional line follower”, “movement in a corridor” and “Line following with obstacle avoidance”. For each of these algorithms we present the control principle, the simulation program and the results obtained. For the first application, the robot must be equipped with a simple colour sensor placed under the robot to measure the gray level of the ground. In the second and third applications, the robot is equipped with a ultrasonic proximity sensor to detect the end of the corridor or an obstacle in its path, and laser-type proximity sensors placed on the left and right sides of the robot to measure its distances from the walls. The robot will use these measurements and the control algorithms presented in this thesis to carry out the requested task in each application.

Keywords : Unicycle mobile robot, CoppeliaSim, LUA programming language, colour sensor, Ultrasonic proximity sensor, laser-type proximity sensors

Sommaire

Table des figures

Liste des abréviations

b : Moitié de la distance entre les deux roues motrices du robot unicycle

a : Distance entre les LIDAR avant et arrière d'un même coté

Couleur_sol : Couleur du sol mesurée en niveau de gris

dfr, drr, dfl, drl : Distances entres les capteurs LIDAR et les murs droit et gauche

e : Distance entre les roues motrices et la roue libre,

ɣ différence entre l'angle de la roue libre et l'orientation du couloir.

k : Gain du contrôleur proportionnel

r : Rayons des roues motrices

V : Vitesse linéaire du robot

Vg,Vd : Vitesses linéaires des roues gauche et droite

α angle de la roue libre par rapport au robot.

ω : Vitesse angulaire de rotation du robot

ωg , ωd ou **ωl, ωr :** Vitesses angulaires des roues gauche et droite

φ angle d'orientation du couloir par rapport au robot.

Introduction général

Dans ce projet, nous proposons d'étudier et de simuler trois algorithmes de robotique mobile : le « Suiveur de ligne proportionnel », la « Navigation entre deux murs » et le « Suiveur de ligne avec évitement d'obstacles ». Ces applications sont qualifiées d'algorithmes de base car elles constituent de bons exemples d'introduction à la robotique mobile pour les débutants qui souhaitent s'initier à ce domaine. En effet, ce type d'applications est très courant dans les concours de robotique pour jeunes amateurs.

La première application étudiée, le « Suiveur de ligne proportionnel », repose sur la mesure du niveau de gris du sol grâce à un capteur de couleur. Cet algorithme peut être utile à un robot mobile qui doit transporter des objets d'un endroit à un autre en suivant toujours le même chemin. Ce chemin pourrait être indiqué au robot mobile par une simple ligne tracée 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. Le robot doit donc être bien centré dans le couloir et suivre les variations de direction des murs du couloir. De plus, le robot doit commencer à ralentir avant de s'arrêter complètement lorsqu'il arrive à la fin du couloir. Dans cette application, le robot utilise un capteur de proximité ultrason à l'avant et des capteurs de proximité laser sur les côtés.

Le troisième algorithme étudié est le suivi d'une ligne sur le sol avec évitement d'obstacles. Dans cette application, le robot doit suivre la ligne et contourner tout obstacle rencontré sur son chemin. Pour cela, le robot utilise un capteur de couleur placé sous le robot, un capteur de proximité ultrason à l'avant et des capteurs de proximité laser sur les côtés, comme dans l'application précédente.

Ce mémoire est organisé en trois chapitres. Dans le premier chapitre nous présentons le robot unicycle et les capteurs utilisés dans les trois applications étudiées, et le simulateur de robotique CoppeliaSim. Dans le deuxième chapitre nous présentons les principes de contrôle de ces trois algorithmes. Et enfin, dans le troisième chapitre nous expliquons la construction de la scène dans chacune des trois applications étudiées, l'implémentation de ces algorithmes dans le simulateur CoppeliaSim et les résultats des simulations.

Chapitre I

CoppeliaSim

et

Robot unicycle

I.1 Introduction

On présente dans chapitre le robot unicycle, le simulateur de robotique CoppeliaSim et les capteurs utilisés dans les trois applications étudiées.

I.2 Robot unicycle

Le robot unicycle est une plate-forme de robot mobile actionné par deux roues indépendantes situées sur le même axe et possédant une roues libre pour assurer sa stabilité. Comme tous les robots, il peut être équipé de certains types de capteurs suivant la tâche qu'il aura à accomplir.

Figure I.1 Robot mobile industriel (en bas) et robot mobile pour la recherche scientifique (les deux robots en haut)

I.2.1 Modèle cinématique d'un robot unicycle

Le mouvement d'un robot unicycle est défini par deux composantes de vitesse : **V** vitesse linéaire suivant la direction longitudinal du robot et **ω** une vitesse angulaire de rotation autour d'un centre instantanée de rotation (**CIR**) de rayon de courbure **R [5]**.

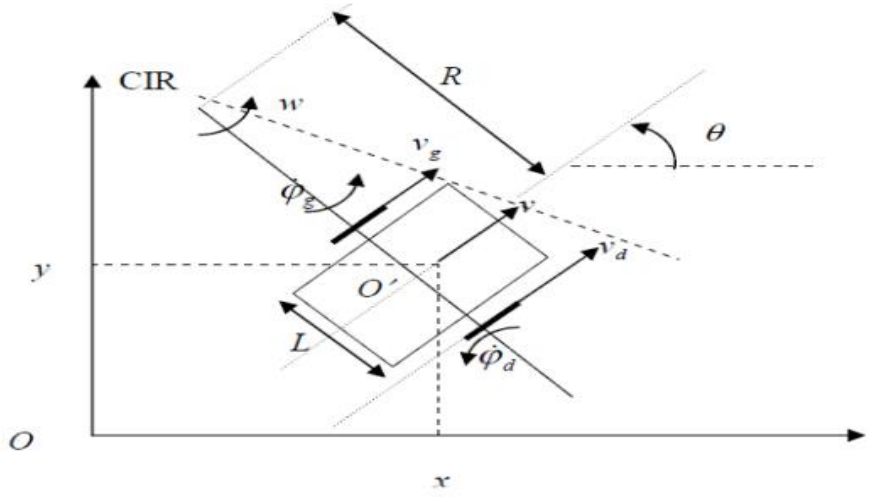


Figure I.2 Paramètres de configuration d'un robot unicycle

Pour un robot unicycle dont les vitesses V et ω sont connues les vitesses linéaires des roues droite et gauche Vd et Vg sont calculées avec la relation suivante :

$$V_d = V + \left(\frac{L}{2}\omega\right) \qquad V_g = V - \left(\frac{L}{2}\omega\right) \tag{I.1}$$

où L est la distance entre les deux roues motrices

Les vitesses angulaires des roues ωd et ωg sont déterminées avec les relations suivantes :

$$\omega_d = \frac{V_d}{r} \qquad \omega_g = \frac{V_g}{r} \tag{I.2}$$

où **r** est le rayon des deux roues motrices

Inversement, les relations suivantes permettent de déterminer V et **ω** à partir de **ωd** et **ωg.**

$$V_g = V\left(\cos\alpha + \frac{L}{2e}\sin\alpha\right) \qquad V_r = V\left(\cos\alpha - \frac{L}{2e}\sin\alpha\right) \tag{I.3}$$

Les relations suivantes donnent **Vr et Vg** en fonction de **α**l'angle d'orientation de la roue libre par rapport au robot**.**

$$\boxed{V = \frac{r(\omega_d + \omega_g)}{2}} \quad \boxed{\omega = \frac{r(\omega_d - \omega_g)}{L}} \tag{I.4}$$

Où **e** représente la distance entre les roues motrices et la roue libre.

I.2.2 Présentation du robot Pioneer

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. **[16]**

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.3 Robot Pioneer P3 DX

Spécifications techniques du robot P3-DX **[16]**:

- 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 ultrason 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 Les capteurs utilisés

I.2.3.1 Capteur de couleur

Un capteur de couleur 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.

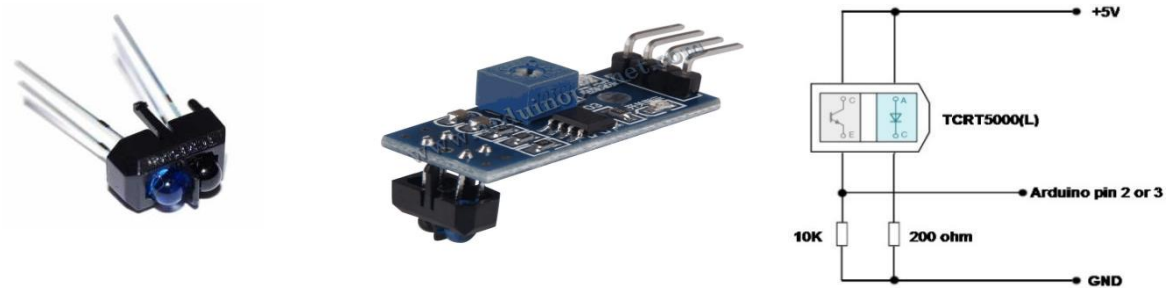


Figure I.4 Capteur de couleur IR (à gauche) – Carte et circuit électronique du capteur (à droite)

I.2.3.2 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 entre le capteur et 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. **[6][7]**

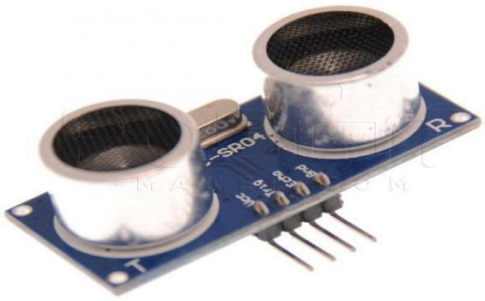


Figure I.5 Carte électronique d'un capteur de proximité ultrason

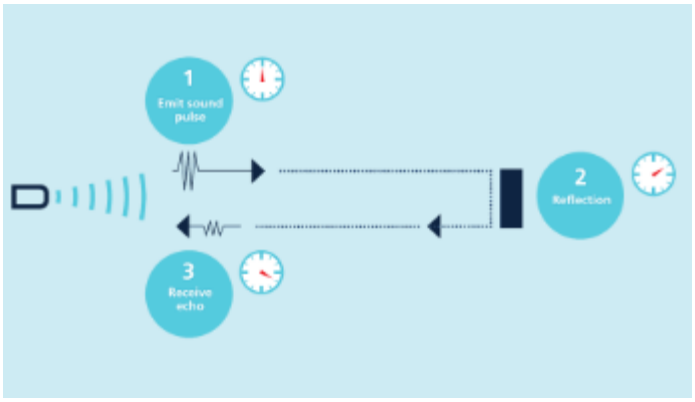


Figure I.6 Principe de fonctionnement d'un capteur de proximité ultrason

I.2.3.3 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. **[8]**

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 :

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

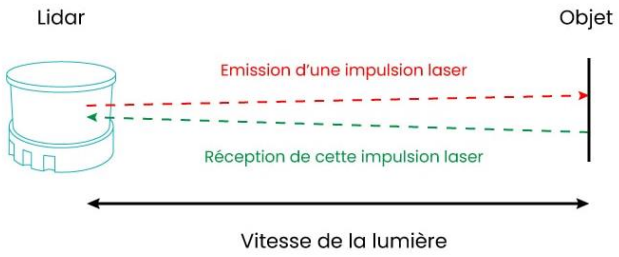


Figure I.7 Principe de fonctionnement d'un Lidar

I.3 Simulateur CoppeliaSim

I.3.1 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 :

Figure I.8 CoppeliaSim logo

- 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
- On peut 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.**[5]**

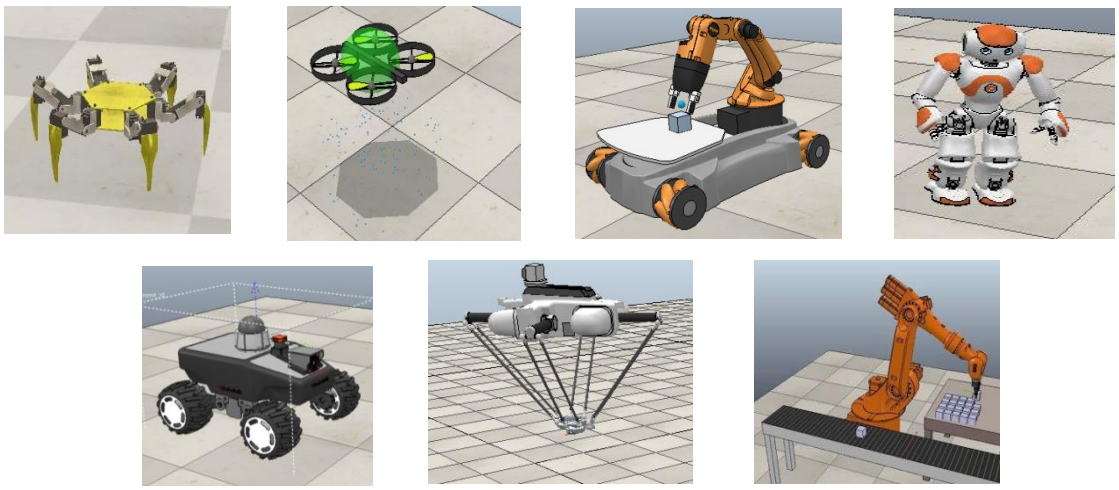


Figure I.9 Quelques modèles de robots existants dans CoppeliaSim

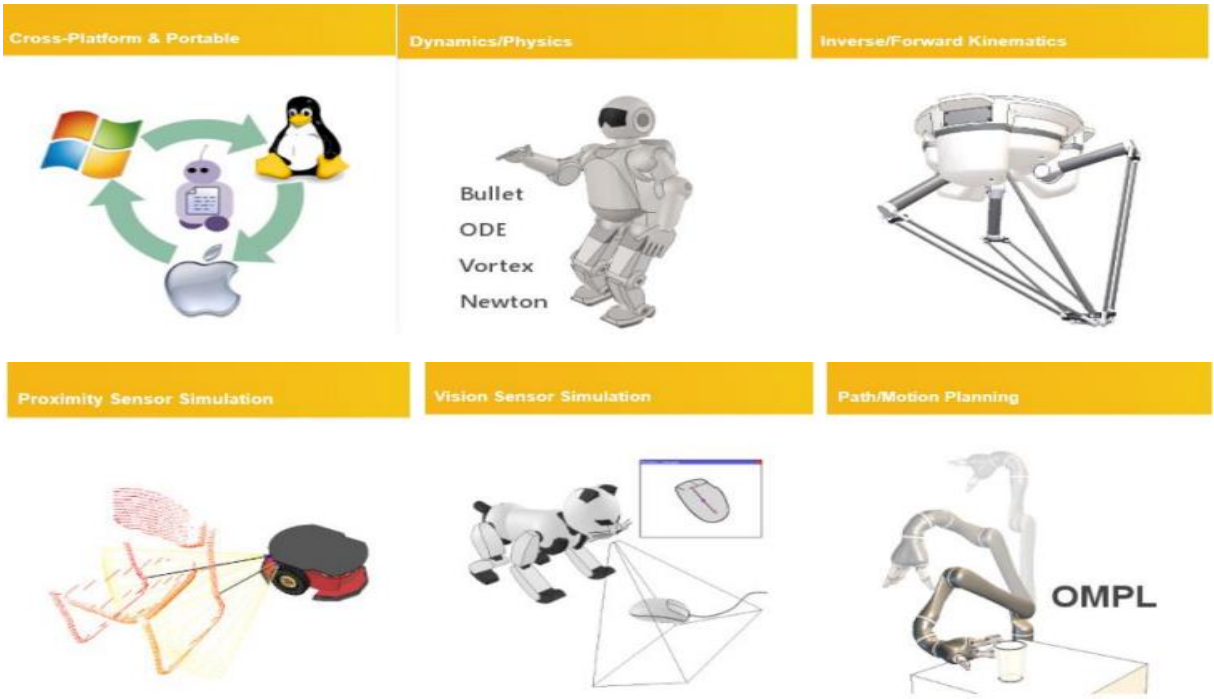


Figure I.10 Illustration de quelques caractéristiques du simulateur CoppeliaSim

I.3.2 Interface utilisateur graphique de CoppeliaSim

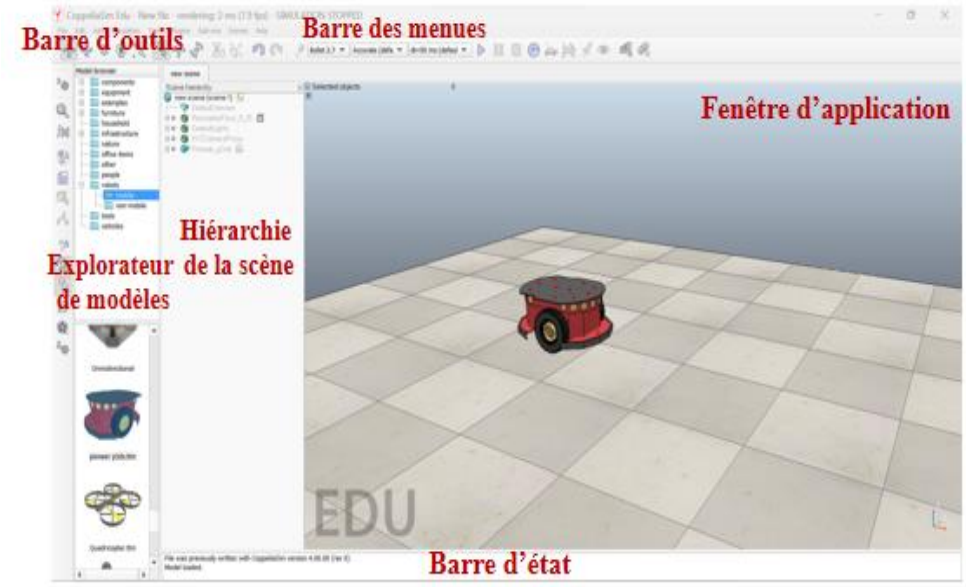


Figure I.11 Interface utilisateur CoppeliaSim

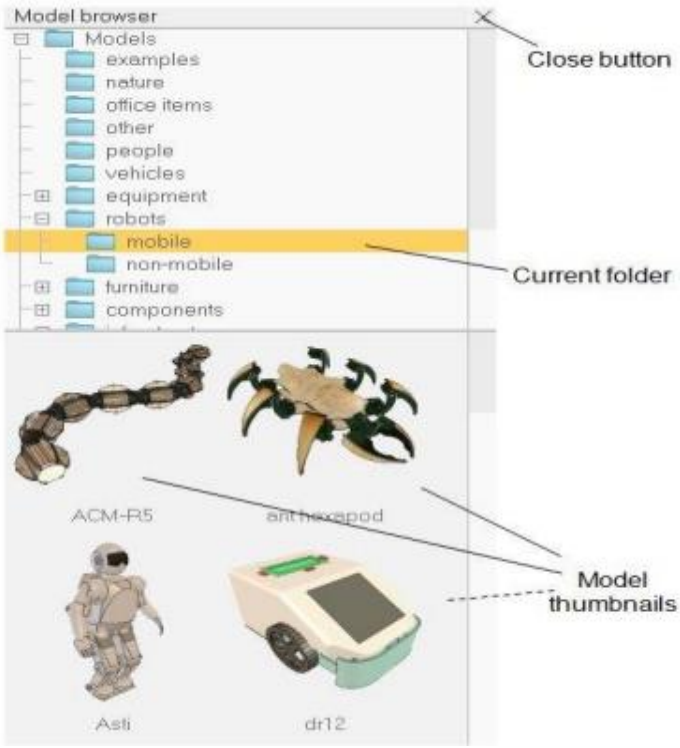


Figure I.12 Explorateur de modèles

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 lumières 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

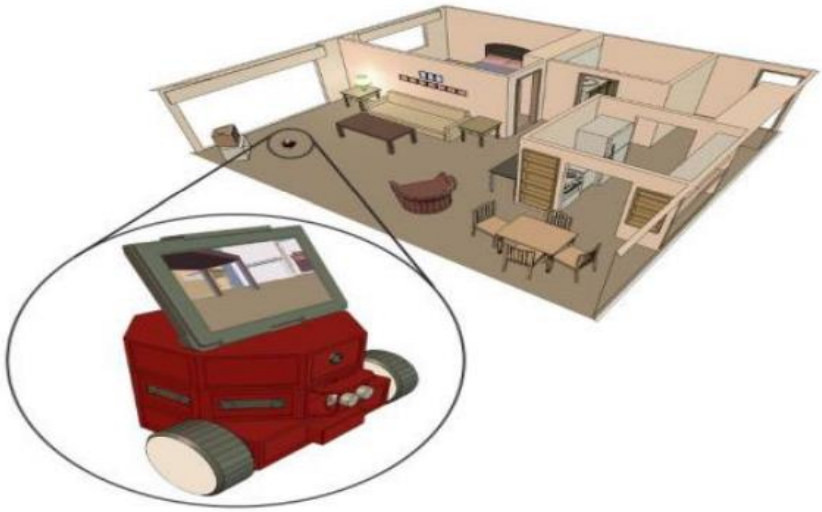


Figure I.13 Exemple de scène dans CoppeliaSim

La fenêtre hiérarchie de la scène (figure II.6) affiche une structure arborescente de la scène dans laquelle on peut voir les objets constituants la scène et leur hiérarchie. Un objet dans la hiérarchie de la scène peut être glissée et déposée sur un autre objet, afin de créer une relation parent-enfant

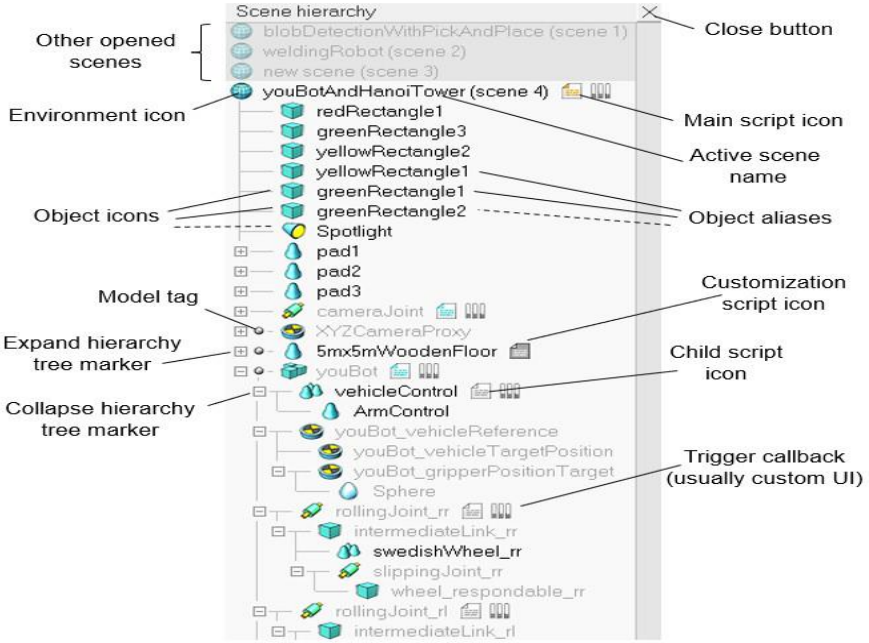
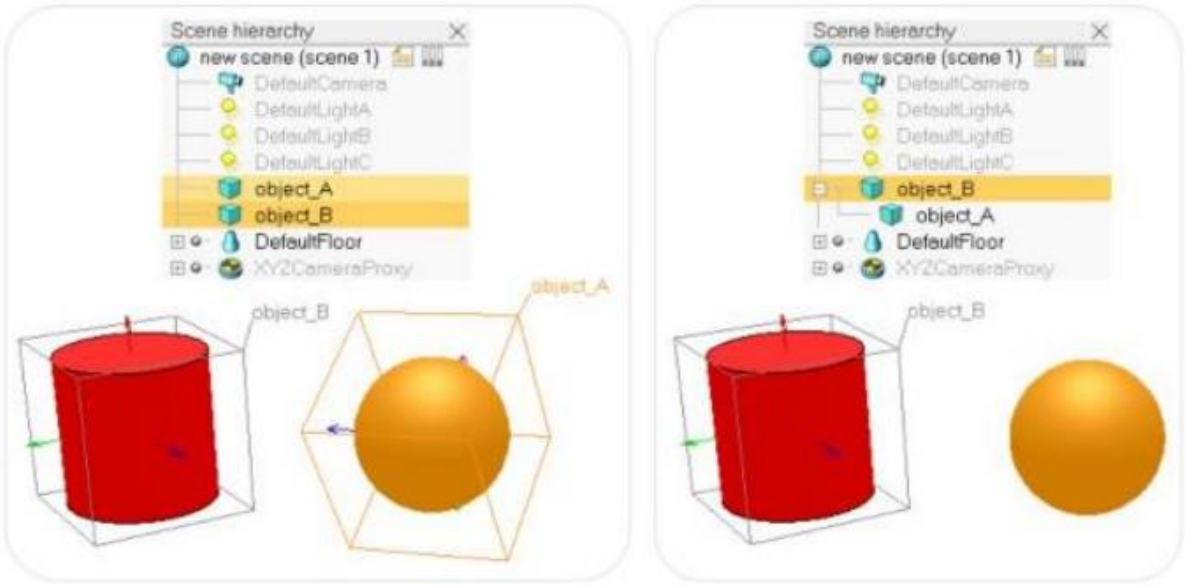


Figure I.14 Fenêtre hiérarchie d'une scène

Si l'objet A dans la hiérarchie de la scène est construit sur l'objet B (figure I.15), l'objet B est le parent de l'objet A. Réciproquement, l'objet A est l'enfant de l'objet B. Ainsi, lorsque l'objet A se déplace, l'objet B le suivra automatiquement puisqu'il est attaché à l'objet B.



[(1) Before attaching object A to object B, (2) after attaching object A to object B]

Figure II.15 Relation Parent/Enfant entre objets de la scène

I.3.2.1 Les objets de la scène

Les objets de scène (figure I.16) 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.

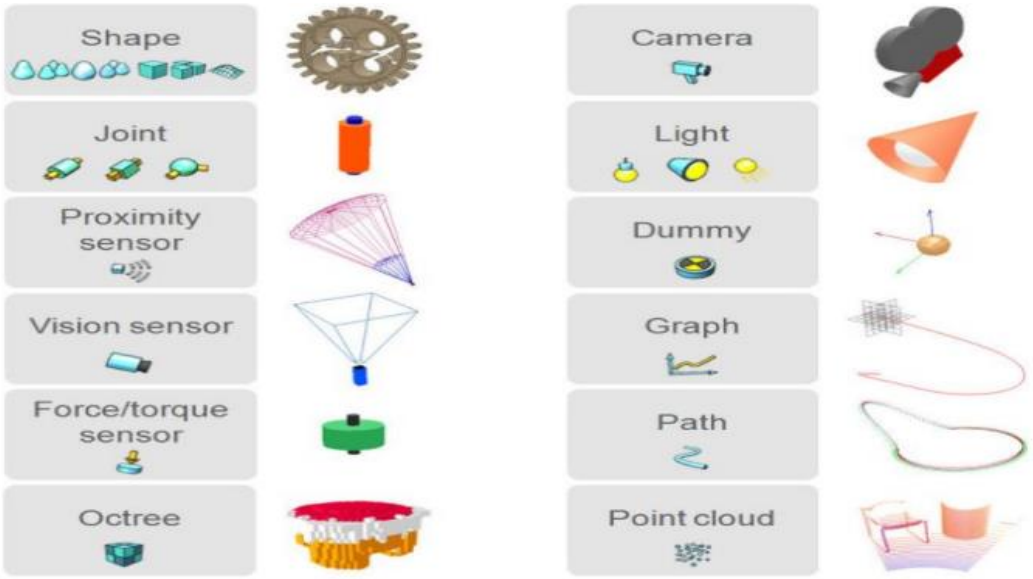


Figure I.16 Les différents types d'objets que peut contenir une scène

La figure I.17 affiche les 5 formes primitives (plan, disque, cube, sphère et cylindre) qu'on peut directement ajouter dans une scène.

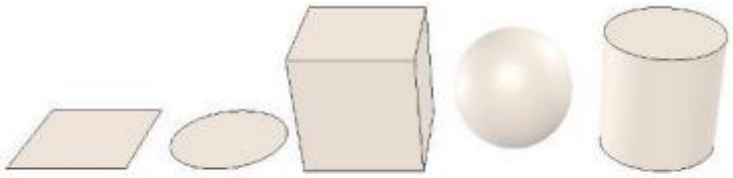


Figure I.17 Formes primitives dans CoppeliaSim

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 prises en charge dans CoppeliaSim sont de quatre types : rotoïde, prismatique, vis et sphérique.

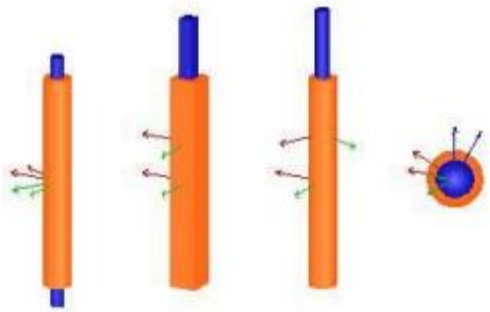


Figure I.18 Articulations rotoïdes, prismatiques, vis et sphériques (de gauche à droite)

Il existe dans CoppeliaSim deux types de **capteurs de vision** qui peuvent être ajustés à différentes fins :

- Type projection orthographique
- Type projection en perspective

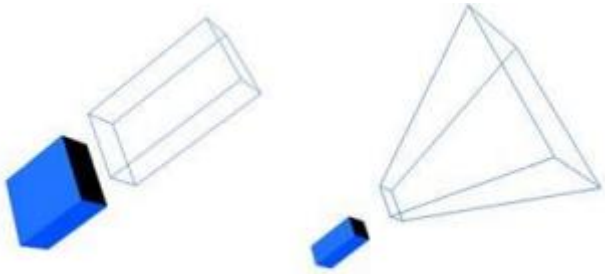


Figure I.19 Capteurs de vision orthographique (à gauche) et perspective (à droite)

La boîte de dialogue de la figure I.20 affiche les réglages et les paramètres du capteur de vision. Parmi ces paramètres on, a le type de caméra (perspective ou orthographique), l'angle de perspective et la résolution de l'image

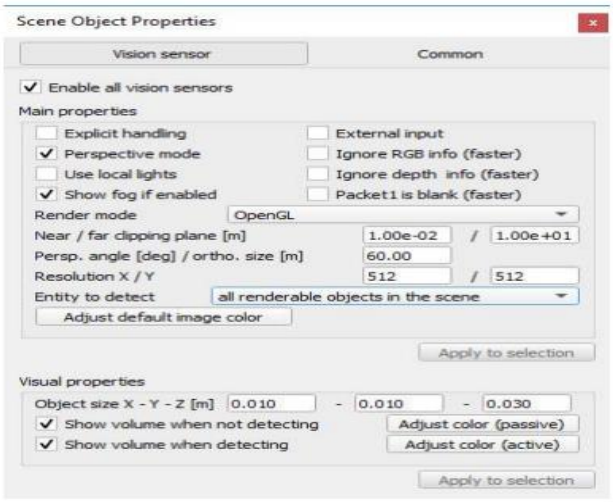


Figure I.20 Paramètres de réglage d'un capteur de vision

Dans CoppeliaSim les **capteurs de proximité** peuvent réaliser une simple détection de type rayon ou des détections de type volume pour modéliser presque tous les types de capteurs de proximité existants réellement (ultrason, infrarouge, laser)

Six types de capteurs de proximité sont disponibles et peuvent être personnalisés dans une large mesure :

- Type rayon.
- Type pyramidal
- Type cylindre
- Type disque
- Type conique
- Type aléatoire

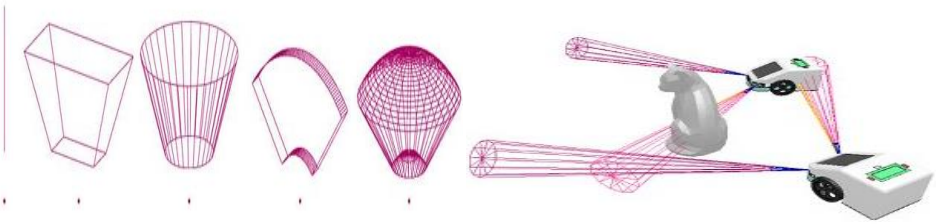


Figure I.21 De gauche à droite, capteurs de proximité de type rayon, pyramide, cylindre, disque, conique et aléatoire

I.4 Programmation avec CoppeliaSim

CoppeliaSim fournit une bibliothèque de plus de 100 fonctions. Ces fonctions permettent entre autre de lire les valeurs des capteurs ou d'actionner les moteurs d'un robot. Les programmes de simulation peuvent être écrits directement dans le simulateur avec le langage **LUA** ou utiliser une API distante qui 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.

On présente dans le tableau ci-dessous les quatre fonctions qu'on a utilisées dans notre code LUA.

Fonctions	Utilisation
Sim.getObjectHandle	Cette fonction retourne un descripteur d'objet (Handle) pour les objets de la scène repérés par leurs noms dans la hiérarchie
Sim.getVisionSensorImage	Cette fonction retourne l'image RGB (ou une partie de celle-ci) d'un capteur de vision
Sim.setJointTargetVelocity	Cette fonction permet de définir la vitesse de tous les types d'articulation à l'exception de l'articulation sphérique.
Sim.readProximitySensor	Cette fonction permet de lire l'état d'un capteur de proximité (c.à.d s'il y a détection ou non) et donne la distance mesurée entre le capteur et l'obstacle dans le cas d'une détection.

I.5 Conclusion

Dans ce chapitre, nous avons donné un aperçu sur le robot unicycle et son modèle cinématique, un aperçu sur les capteurs de proximité ultrason et laser et le capteur de couleur utilisés dans la simulation, et un aperçu sur le simulateur CoppeliaSim (ses caractéristiques, son interface graphique, ses modèles de robots, ses différents capteurs et actionneurs, et quelques fonctions de programmation)

Chapitre II

Présentation des algorithmes étudiés

II.1 Introduction

Dans cet chapitre on présente trois algorithmes : « Suiveur de ligne proportionnel » , «déplacement entre deux murs » et « Suivi de ligne avec évitement d'obstacles ». Ces algorithmes sont appliqués à un robot mobile unicycle.

II.2 Algorithme 1 « Suiveur de ligne proportionnel»

II.2.1 Problématique

Dans cette application, le robot a pour objectif de suivre une ligne noir sur le sol de forme arbitraire et fermée en utilisant un capteur de couleur. Ce robot suiveur de ligne à base d'un contrôleur proportionnel est ainsi appelé pour le différencier des autres types de robots suiveurs de lignes. **[14]**

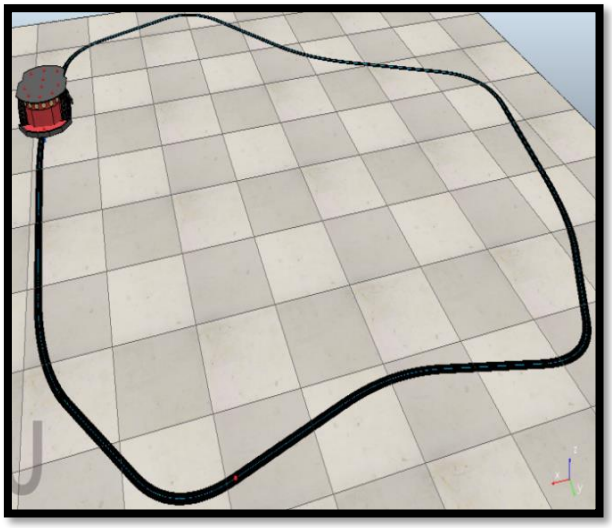


Figure II.1 La scène **(**path et robot pionner)

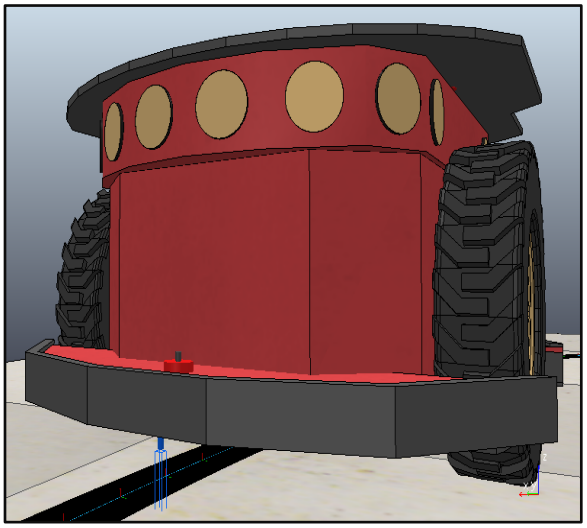


Figure II.2 Capteur de couleur placé sous le robot

II.2.2 Le principe du contrôle

Pour que le robot suive la ligne sur le sol avec une vitesse V fixe, il doit à chaque instant changer son orientation (en réglant convenablement sa vitesse de rotation ω) en fonction de la variation de la direction de la ligne.

$$\omega = -k(couleur-0.5) \qquad \textbf{(II.1)}$$

Pour cela, le capteur de couleur doit être exactement sur le bord de ligne, et donc mesurer un niveau de gris de 0.5 (la couleur blanche ayant un niveau de gris égal à 1 et une couleur noir un niveau de gris égal à 0).

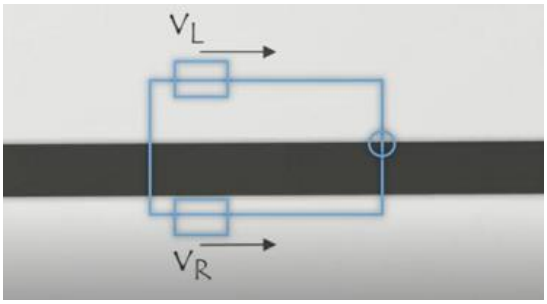


Figure II.3 Robot avec capteur de couleur placé au dessus du bord de la ligne

Pour que le robot initialement placé sur la ligne noir avance avec une vitesse V tout en en restant sur le bord de la ligne, il doit constamment changer son orientation (ω≠ 0) en fonction de l'écart entre le niveau de gris mesuré et le niveau de gris du bord de la ligne (0.5).

Les équations (I.2) du chapitre I permettent de déterminer les vitesses Vg et Vd des roues gauche et droite en fonction de ω.

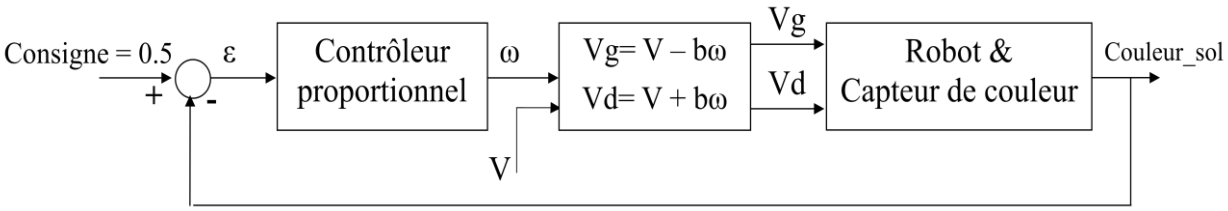


Schéma de contrôle pour le suivi d'une ligne sur le sol

Figure II.4 Schéma de contrôle d'un robot suiveur de ligne proportionnel

II.3 Algorithme 2 « Navigation entre deux murs» (Couloir)

II.3.1 Problématique

Dans cette application le robot doit avancer dans un couloir (entre deux murs) de forme quelconque et largeur variable de façon à rester bien au centre entre les deux murs. Pour cela le robot est équipé de quatre capteurs de proximité de type Lidar placées sur les cotés du robot, deux de chaque coté et dirigés face aux murs. Arrivé à la fin du couloir qui est fermé, le robot doit progressivement diminuer sa vitesse jusqu'à s'arrêter complètement grâce au capteur de proximité ultrason situé à l'avant du robot en laissant une distance de sécurité entre lui et le mur de la fin. **[12][13]**

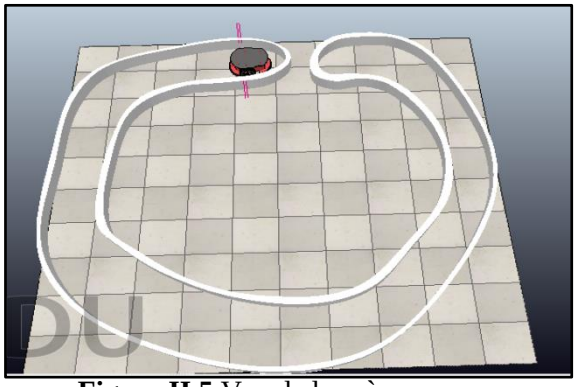


Figure II.5 Vue de la scène

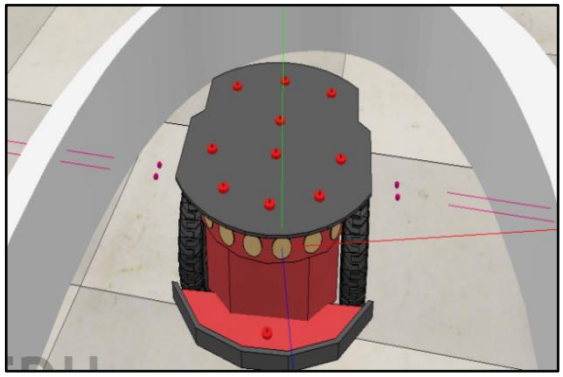


Figure II.6 Capteur US et capteurs laser

II.3.2 Le principe de contrôle

Ce schéma montre les différentes étapes de fonctionnement du robot et les conditions pour le passage d'une étape à l'autre.

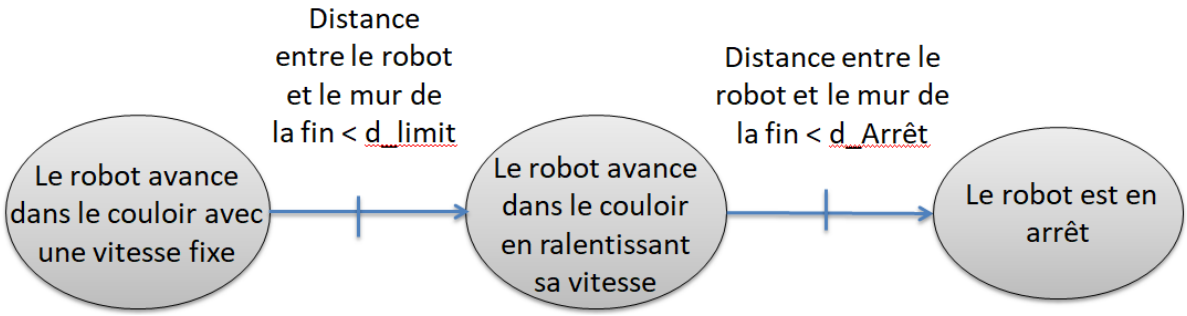


Figure II.7 Schéma de fonctionnement

II.3.2.1 Contrôle pour la navigation dans le couloir

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.

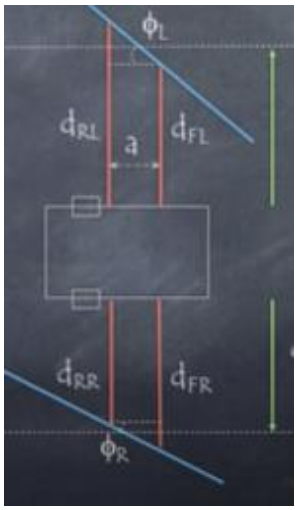


Figure II.8 Paramètres géométriques de la scène pour le suivi de murs

φ_l est l'angle d'orientation du mur gauche par rapport à la direction du robot

$$\varphi_l = Arctan\left(\frac{d_{rl}-d_{fl}}{a}\right) \quad \textbf{(II.2)}$$

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

$$d_l = \frac{d_{fl}-d_{rl}}{2} \quad \textbf{(II.3)}$$

φ_r est l'angle d'orientation du mur droit par rapport à la direction du robot

$$\varphi_r = Arctan\left(\frac{d_{fr}-d_{rr}}{a}\right) \quad \textbf{(II.4)}$$

a est le distance entre les capteurs laser situés sur le même coté.

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

$$d_r = \frac{d_{fr}+d_{rr}}{2} \quad \textbf{(II.5)}$$

L'angle d'orientation du couloir ϕ est la moyenne de ϕ_r et ϕ_l

$$\varphi = \frac{\varphi_l+\varphi_r}{2} \quad \textbf{(II.6)}$$

$\boldsymbol{\gamma}$ est l'angle de la roue libre par rapport à la direction du couloir. C'est la différence entre la l'angle de la roue libre par rapport au robot $\boldsymbol{\alpha}$ et l'angle du couloir par rapport au robot $\boldsymbol{\phi}$.

$$\gamma = \alpha - \varphi \quad \textbf{(II.7)}$$

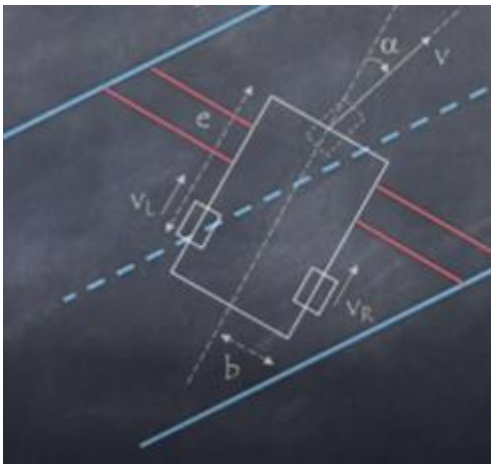


Figure II.9 Paramètres géométriques du robot pour le suivi de murs

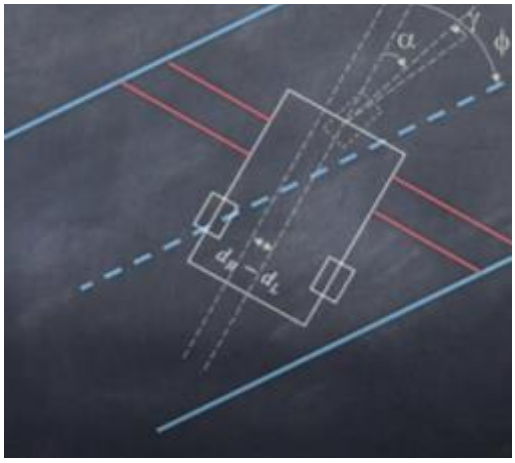


Figure II.10 Représentation des angles α, φ et γ

Lorsque dr = dl, le robot est au milieu du couloir et aussi a la même orientation que le couloir. Autrement dit, dr – dl = 0 → γ=0 et plus la différence dr-dl est grande plus l'angle γ est grand.

On peut donc utiliser un contrôleur proportionnel pour contrôler l'angle γ par rapport à la différence (dr – dl) mesurée.

$$\gamma = k\,(d_r - d_l) \qquad \textbf{(II.8)}$$

D'après les équations (I.1 et I.2) données au chapitre I on peut déterminer les vitesses des roues motrices gauche et droite correspondants à l'angle α.

Le schéma suivant résume le principe de contrôle pour que le robot puisse avancer dans le couloir avec une vitesse V fixe en restant au milieu du couloir.

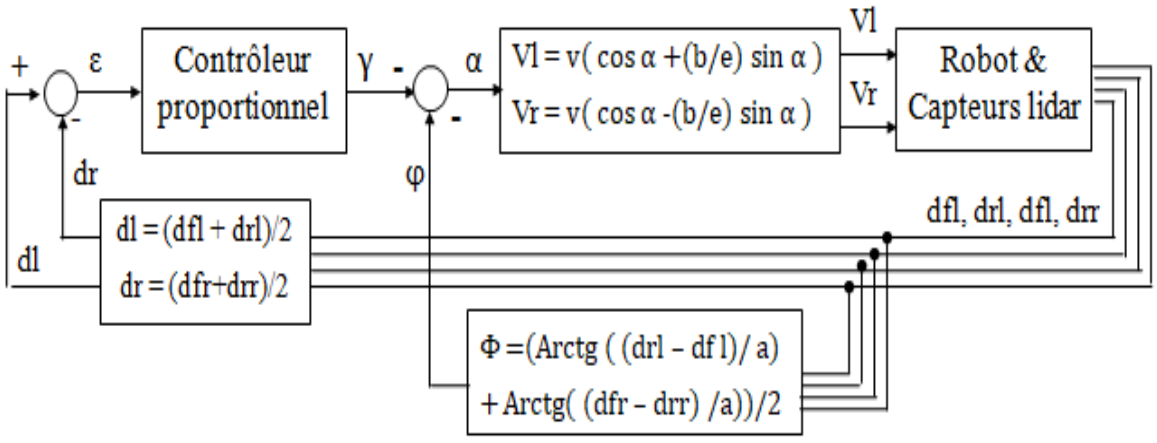


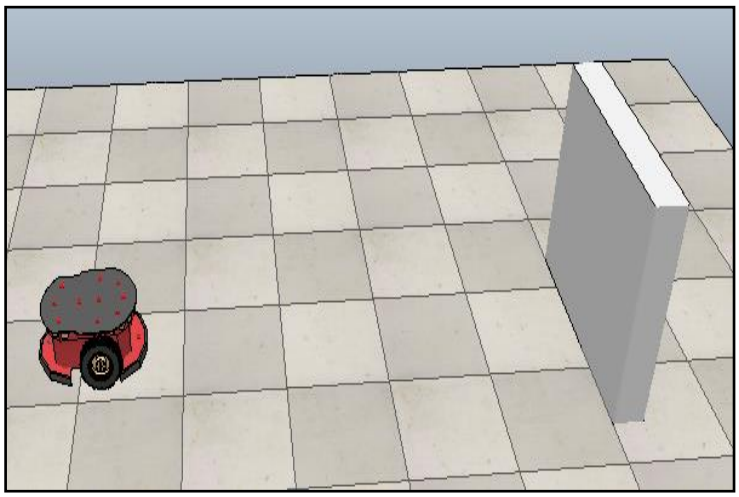
Figure II.11 Schéma de contrôle pour la navigation dans un couloir

II.3.2.2 Contrôle pour la décélération et l'arrêt du robot

Arrivé à la fin du couloir (qui est fermé) le robot doit s'apprêter à s'arrêter en laissant une distance de sécurité d_Arrêt entre lui et le mur de la fin. Pour cela, le robot doit d'abord

diminuer sa vitesse de façon linéaire suivant la relation II.9 à partir d'une distance d égale à d_limit. Lorsque la distance d est égale à d_Arrêt le robot doit s'arrêter complètement (v=0).

$$v = \begin{cases} Vmax\left(\frac{d}{d\_limit}\right) & si\ d > d\_Arrêt \\ 0 & si\ \leq\ d\_Arrêt \end{cases} \quad \textbf{(II.9)}$$



FigureII.12 Robot en face d'un obstacle

II.4 Algorithme 3 «Suivi de ligne avec évitement d'obstacles»

II.4.1 Problématique

Dans cette application, le robot a pour objectif de suivre une ligne noir sur le sol tout en contournant les obstacles se trouvant sur son chemin. Pour se faire, il utilise trois types de capteurs : Capteurs de proximité US et laser et un capteur de couleur. **[15]**

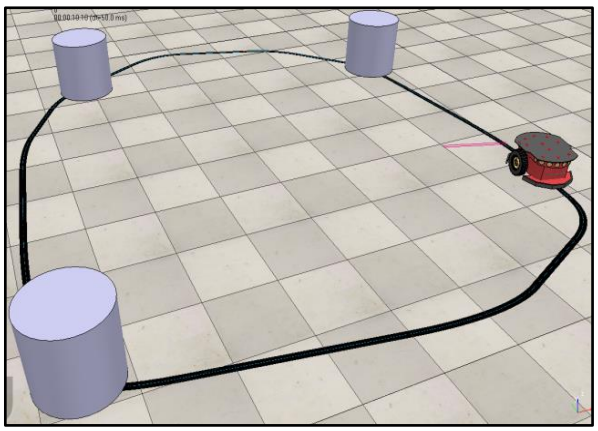


Figure II.13 Vue de la Scène

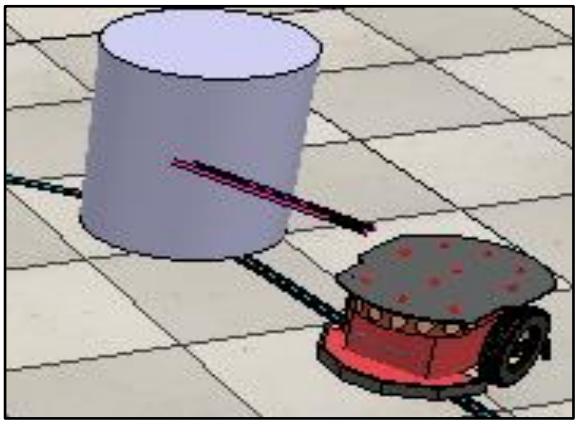


Figure II.14 Capteurs utilisés dans le cas du suivi de ligne avec évitement d'obstacles

II.4.2 Le principe de contrôle

Ce schéma montre le cycle des étapes à suivre par le robot et les conditions pour le passage d'une étape à l'autre.

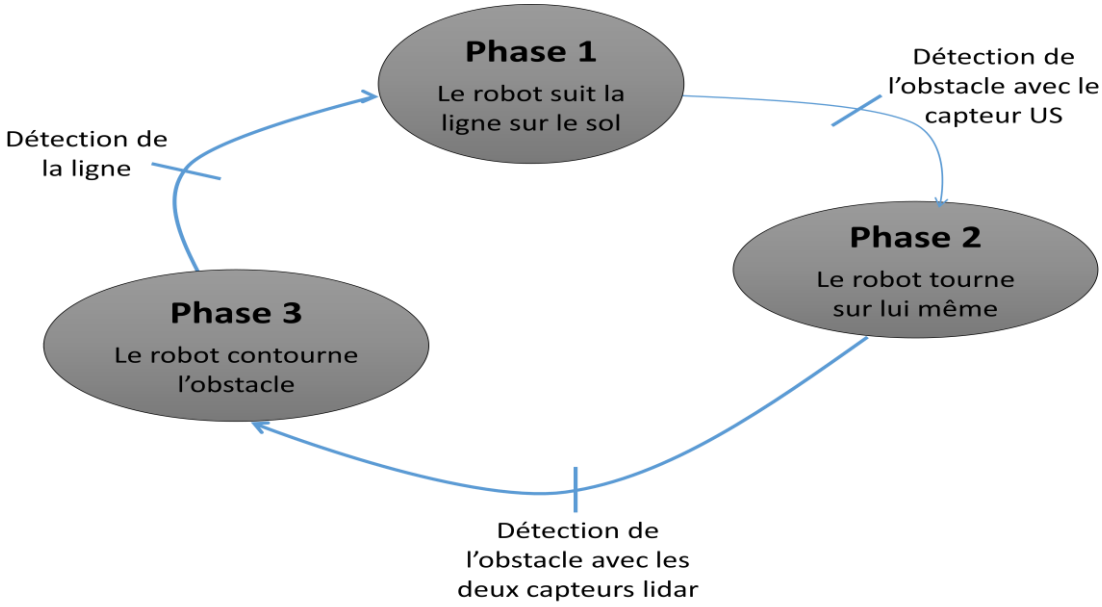


Figure II.15 Schéma globale de fonctionnement

Phase 1 : Suivi de la ligne sur le sol (le même que l'application 1)

- Le robot doit suivre le bord de la ligne noir.
- ω la vitesse angulaire du robot doit être proportionnelle à l'écart entre la couleur du sol mesurée et la couleur du bord de la ligne (Niveau de gris = 0.5).

$$\omega = k(couleur\_sol - 0.5) \qquad \textbf{(II.10)}$$

Phase 2 : Rotation du robot autour de lui même

Cette phase commence quand le robot détecte un obstacle devant lui avec le capteur US. Pour commencer à contourner l'obstacle le robot doit détecter l'obstacle avec ses deux capteurs laser situés sur le coté, ce qui n'est pas le cas au départ. Pour cela, le robot doit faire une rotation sur lui-même jusqu'à ce qu'il détecte l'obstacle avec les capteurs laser. La rotation du robot est réalisée quand les vitesses des roues motrices du robot sont égales en module et de sens inverse (équation II.11).

$$V_l = -V_r \qquad \textbf{(II.11)}$$

Où **Vl** et **Vr sont** les vitesses linéaires des roues gauche et droite.

Phase 3 : Contournement de l'obstacle (le même que l'application 2)

Pour réaliser le contournement et l'évitement des obstacles en maintenant une distance d fixe entre le robot et l'obstacle, c'est le principe de l'algorithme de navigation dans un couloir qui est exploité.

Rappel des équations :

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

$$\varphi = \text{Arctan}\left(\frac{d_{fr} - d_{rr}}{a}\right) \quad \textbf{(II.12)}$$

Relation qui donne α l'angle de la roue libre par rapport au robot à partir de son angle par rapport au mur γ, et de l'angle d'orientation du mur par rapport au robot ϕ

$$\alpha = (\gamma + \varphi) \quad \textbf{(II.13)}$$

On utilise un contrôleur proportionnel pour contrôler l'angle γ suivant la distance dr mesurée et la consigne d.

$$\gamma = k\,(d_r - d) \quad \textbf{(II.14)}$$

II.5 Conclusion

On a présenté dans ce chapitre la problématique et le principe du contrôle pour les trois applications : « Suiveur de ligne proportionnel », « navigation entre deux murs » et « Suivi de ligne sur avec évitement d'obstacles » qui sont appliqués dans les simulations présentées au chapitre III.

Chapitre III

Simulations et résultats

III.1 Introduction

Dans ce chapitre on montre comment sont construites les scènes et comment implémenter les programmes dans le simulateur CoppeliaSim pour les trois algorithmes dont les principes ont été décrits au chapitre II.On présente ensuite les résultats des simulations accompagnés de nos observations et conclusions.

III.2 Simulation de l'algorithme 1« Suiveur de ligne proportionnel»

III.2.1 Construction de la scène

On peut voir sur la figure III.1 représentant la fenêtre de la hiérarchie de la scène, les éléments qui composent la scène construite :

- La ligne tracée 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 en avant et au centre pour mesurer le niveau de gris du sol sous le capteur.

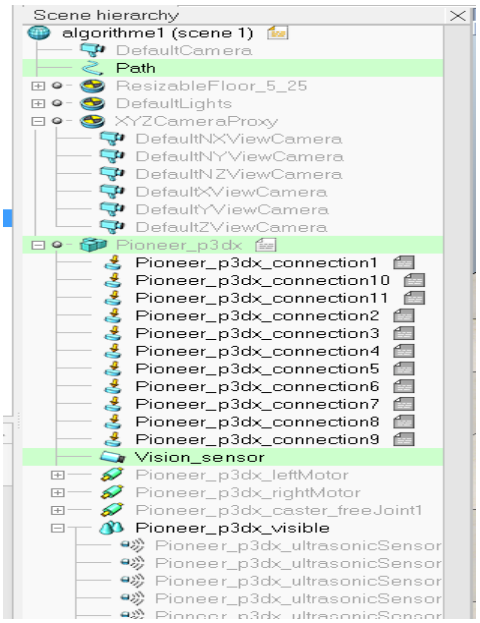


Figure III.1 Fenêtre hiérarchie de la scène construite

Élément de la scène « Path »

L'élément path (la ligne) utilisé dans cette simulation doit être de type segment, plat, et fermé de couleur noir et assez large. **[11]**

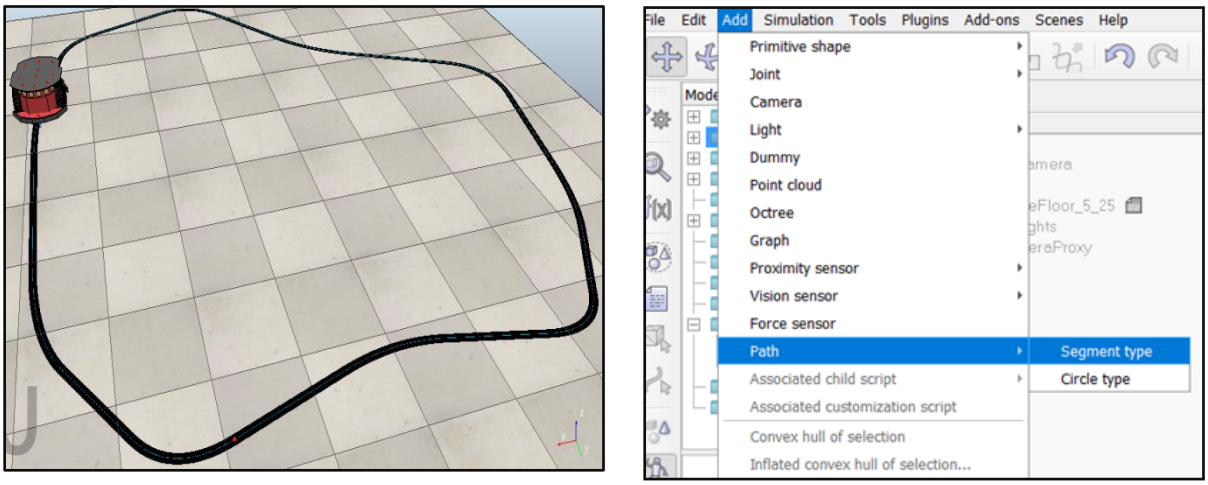


Figure III.2 Ajout d'un élément path de type segment dans la scène (à droite) – Le chemin (path) créé (à gauche)

Élément 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. On doit l'ajouter dans la scène comme child du robot Pioneer. Dans CoppeliaSim le capteur de couleur est un capteur de vision (Vision sensor) avec un seul pixel de quelques mm. **[14]**

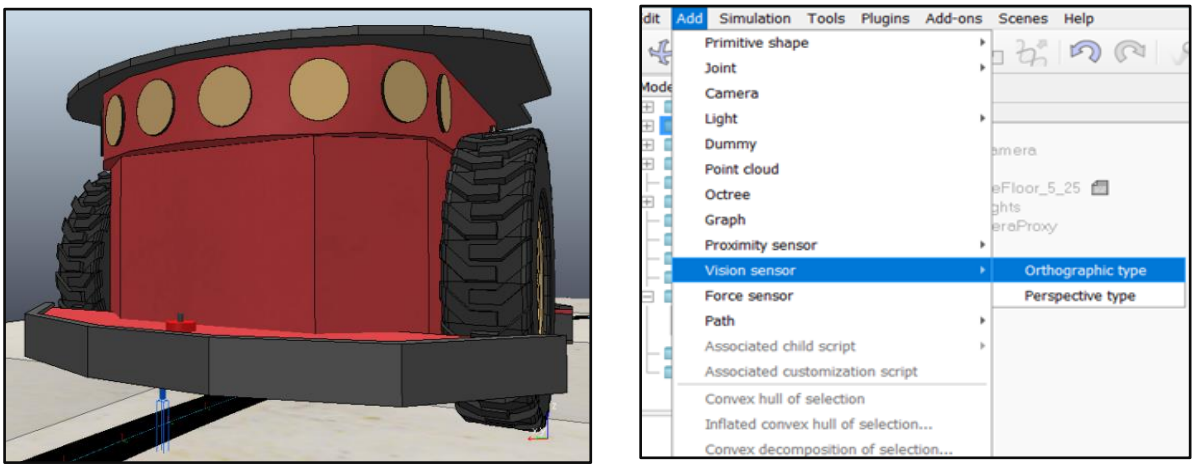


Figure III.3 Ajout d'un capteur de vision orthographique dans la scène (à droite) – Capteur de vision placé sous le robot (à gauche)

III.2.2 Organigramme et programme de simulation

On déduit du schéma de contrôle donné au chapitre II, l'organigramme et le programme suivants :

L'organigramme

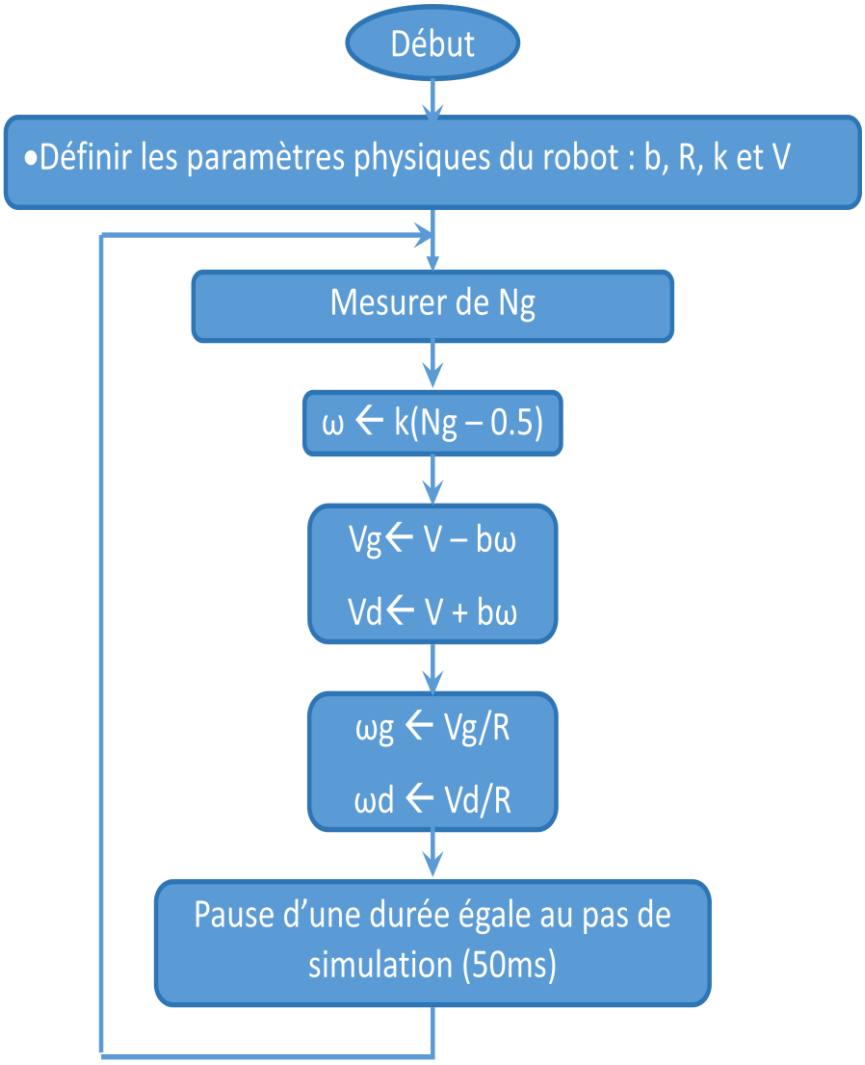


Figure III.4 Organigramme d'un suiveur de ligne proportionnel

- **Ng : Couleur** du sol mesurée en niveau de gris
- **ω :** Vitesse de rotation du robot
- **Vg, Vd, ωg et ωd : Vitesses** linéaires et angulaires des roues gauche et droite
- **R :** Rayon des roues du robot
- **K :** gain
- **2 x b :** Distance entre les deux roues motrices du robot

Programme

```
function sysCall_init()

    motorLeft=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
    motorRight=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
    ir=sim.getObjectHandle("Vision_sensor")
    v=0.15      --la vitesse linéaire du robot
    k=1         --le facteur de proportionalité du controleur
    b=0.33/2    --la moitier de la distance entre les deux roues
    r=0.1       --rayon des roues

end
function sysCall_actuation()

      --mesure du niveau de gris
      ng=sim.getVisionSensorImage(ir+sim.handleflag_greyscale)
      --la vitesse de rotation du robot
      w=-k*(ng[1]-0.4)
      --les vitesse linéaire des roues gauche et droite
      vg=v-(b*w)
      vd=v+(b*w)
      --les vitesse angulaire des roues gauche et droite
      wg=vg/r
      wd=vd/r
      --appliquer les vitesse angulaire wg ,wd des roues gauche et droite
      sim.setJointTargetVelocity(motorLeft,wg)
      sim.setJointTargetVelocity(motorRight,wd)

    end
```

Figure III.5 Programme en langage LUA pour un suiveur de ligne proportionnel

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, 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 Observations et interprétations des résultats de simulation

On remarque que le robot suit la ligne avec des oscillations dont l'ampleur dépend de la valeur du facteur K du contrôleur. K doit être choisie de façon à ce que le robot suive la ligne tout en minimisant les oscillations autour de la ligne. On remarque aussi que la valeur dépend du choix de la vitesse V du robot. Plus V est grand, plus on doit augmenter K.

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

III.3.1 Construction de la scène

Les éléments constituants la scène construite sont donnés sur la figure III.6 représentant lafenêtre « Hiérarchie de la scène » :

- 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)

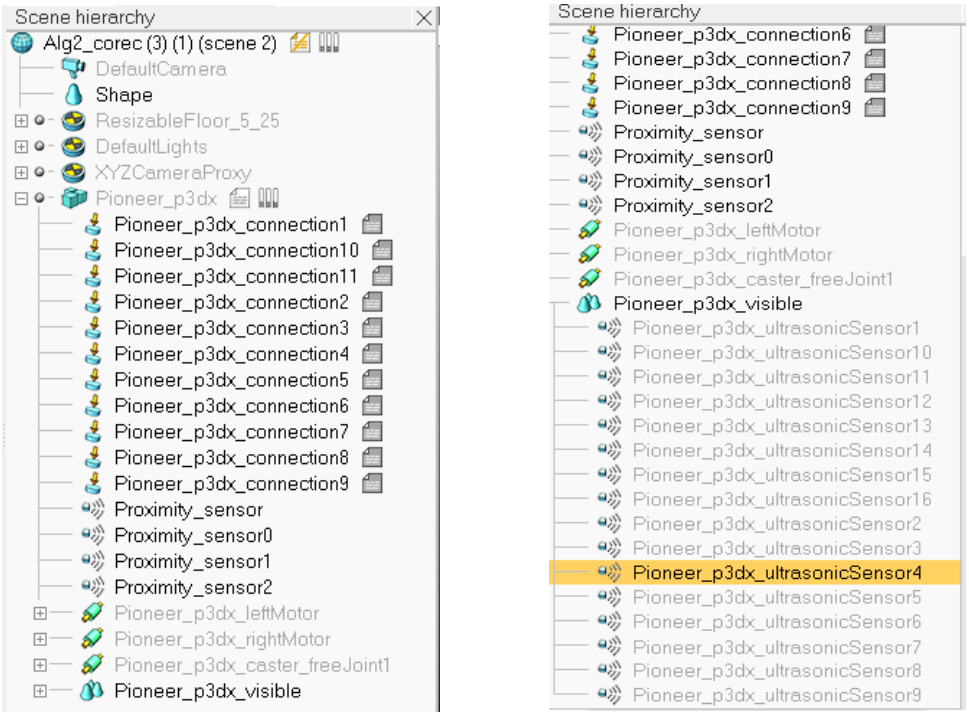


Figure III.6 Fenêtre hiérarchie de la scène construite

Élément de la scène « Shape » (couloir)

« Shape » (figure III.7) représente un couloir qui fait le tour. 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**. [12][13]**

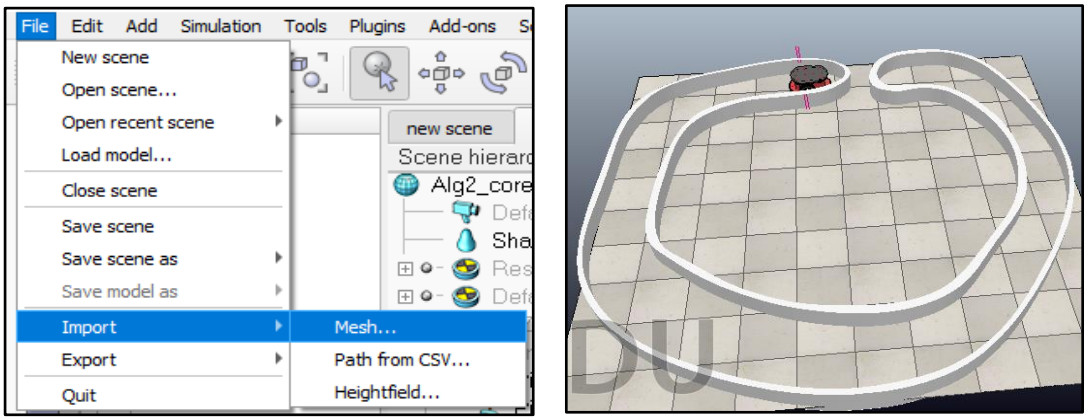


Figure III.7 Importation du modèle du couloir (à gauche) – Le couloir (redimensionné) (à droite)

Élément 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 entre 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.

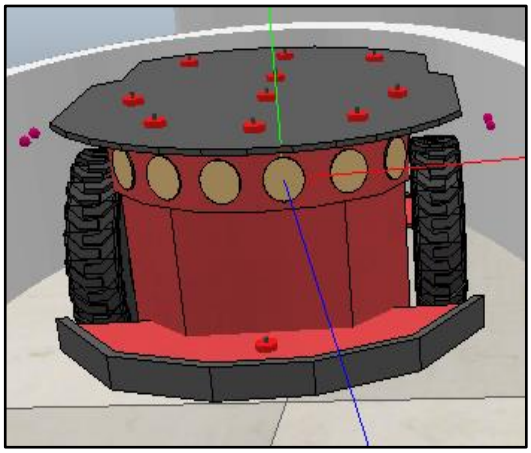


Figure III.8 Le capteur ultrason du robot Pioneer utilisé pour détecter les obstacles se trouvant sur son chemin

Élément 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.

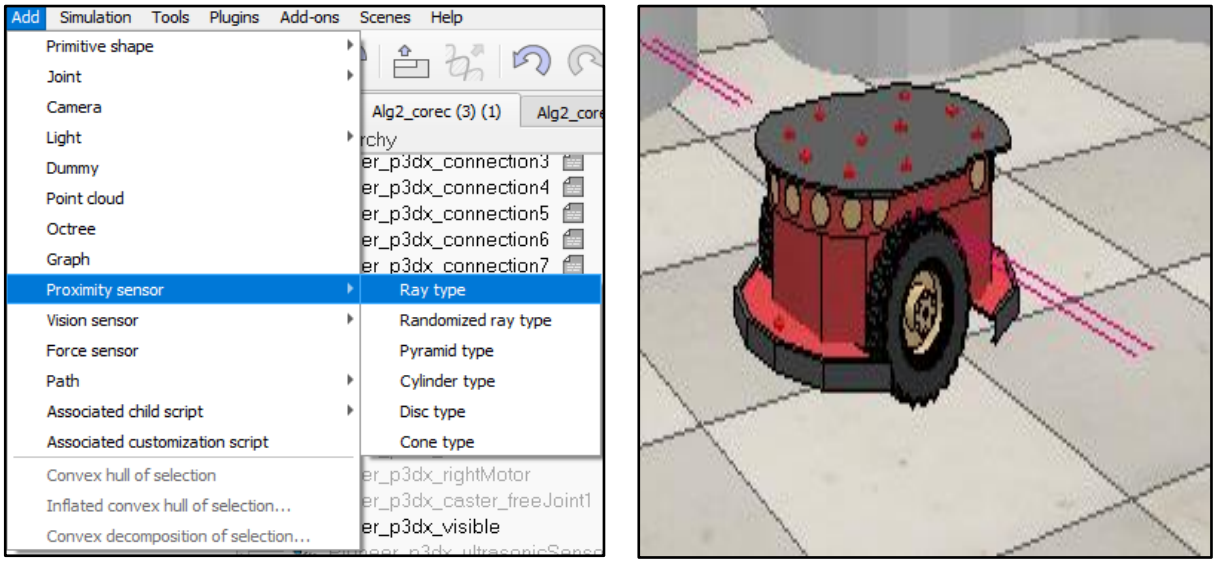


Figure III.9 Capteurs de proximité de type laser (Lidar) ajoutés au robot Pioneer

III.3.2 Organigramme et programme de simulation

On déduit du schéma de contrôle donné au chapitre II, les organigrammes et le programme suivants :

L'organigramme

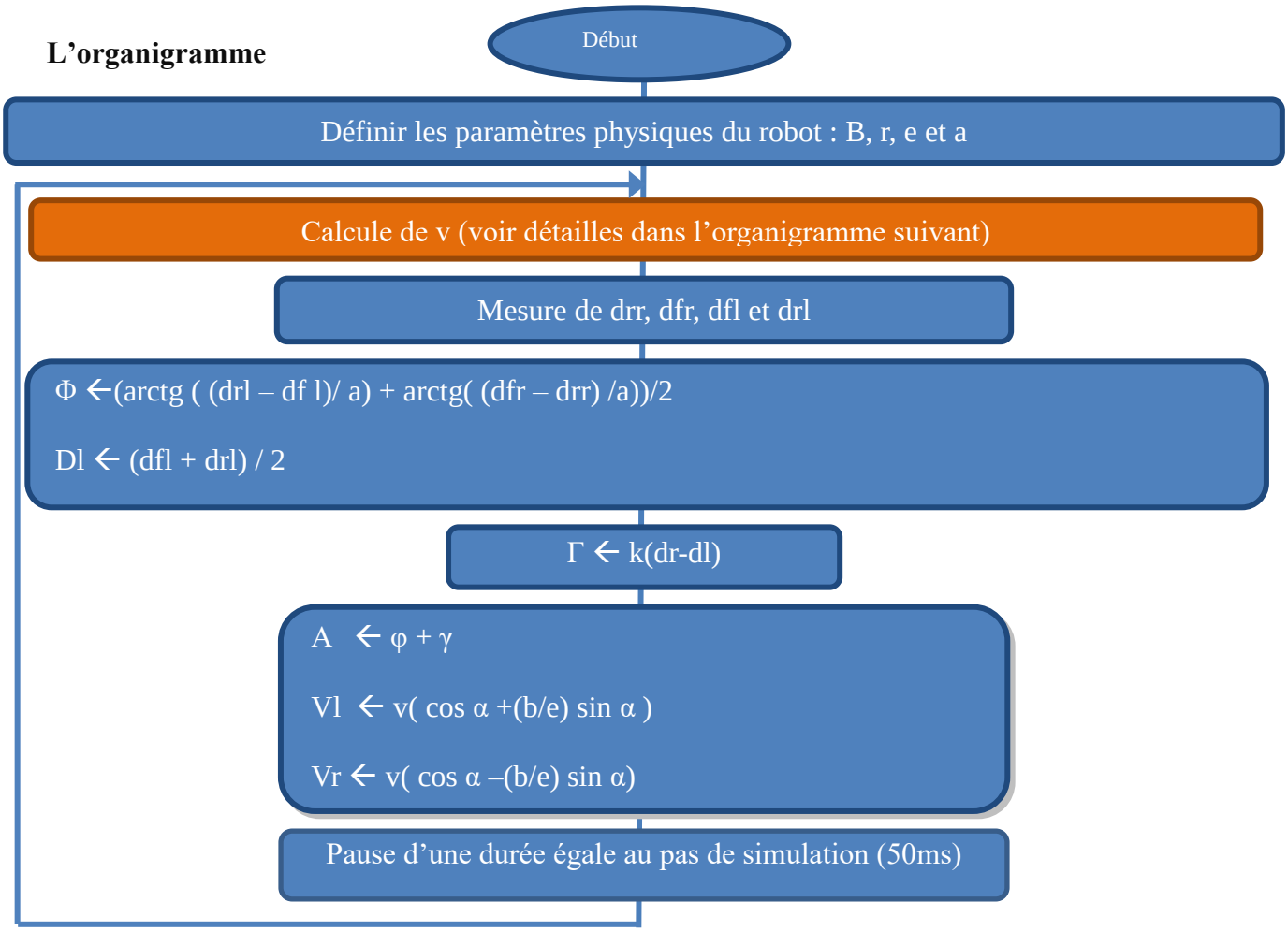


Figure III.10 Organigramme de la navigation d'un robot mobile entre deux murs

- **Dfr, drr, dfl, drl :** Distances entres les capteurs lidar et les murs droit et gauche
- **2xb** : Distance entre les deux roues motrices
- **R** : Rayons des roues motrices
- **E** : Distance entre les roues avants et la roue libre,
- **A** : Distance entre le lidar avant et le lidar arrière de chaque coté
- **Γ** : Angle de la roue libre par rapport au couloir
- **A** : Angle de la roue par rapport au robot
- **ϕ** : Orientation du couloir par rapport au robot
- **V** : Vitesse linéaire du robot
- **Vl**, **vr**, **ωl**, et **ωr** : Vitesses linéaires et angulaires des roues motrices gauche et droites

L'organigramme

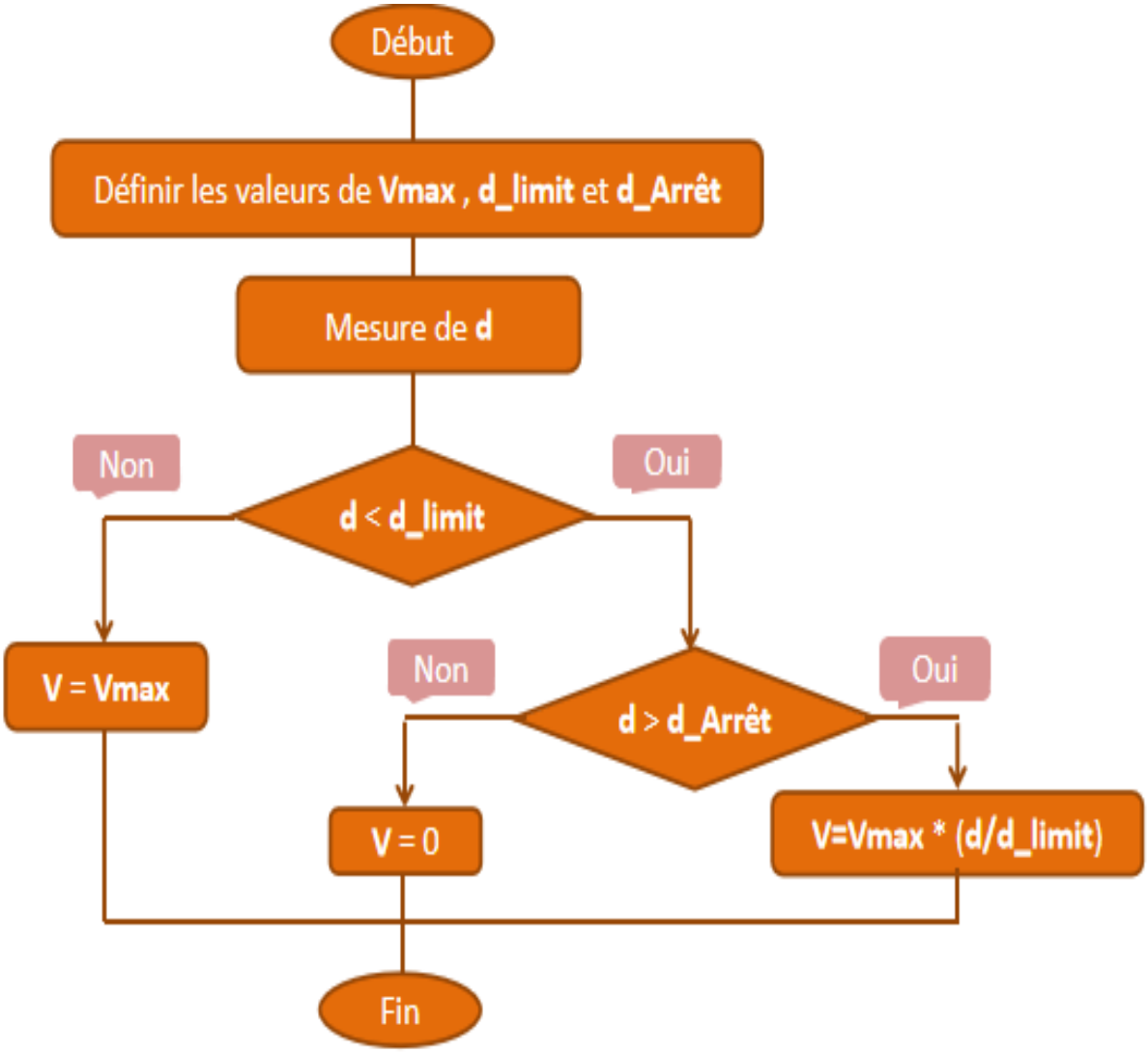


Figure III.11 Organigramme du calcul de la vitesse V du robot

Programme

```
function sysCall_init()
    Lefet=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
   Right=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
    sensor=sim.getObjectHandle("Pioneer_p3dx_ultrasonicSensor4")
    sensor0=sim.getObjectHandle("Proximity_sensor0")
    sensor1=sim.getObjectHandle("Proximity_sensor1")
    sensor2=sim.getObjectHandle("Proximity_sensor2")
    sensor3=sim.getObjectHandle("Proximity_sensor3")
    max_dist=2 --la distance maximale que peut détecter le capteur
    dmin=0.2   --la distance minimale que peut détecter
    vmax=0.2   --la vitesse linéaire du robot
    R=0.0975   --Rayon des roues
    b=0.33/2   --la moitié de la distance entre les deux roues
    e=0.1934   --la distance entre la troisième roue et les deux roues
    a=0.03     --la distance entre deux capteurs situés sur le même coté du robot
    k=2
end

function sysCall_actuation()
    --mesure de distance entre le robot et l'obstacle
    detected,dist=sim.readProximitySensor(sensor)
   if detected <1
    then v=vmax
    else
    if dist >dmin
    then
    v=vmax*(dist/max_dist)--calcul de vitesse linéaire du robot
    else
    v=0
    end
    end
```

```
 --mesure des distance entre le robot et le mur gauche
 det1,dfl=sim.readProximitySensor(sensor0)
 det2,drl=sim.readProximitySensor(sensor1)
 --mesure des distance entre le robot et le mur droite
 det3,dfr=sim.readProximitySensor(sensor2)
 det4,drr=sim.readProximitySensor(sensor3)

    --La distance moyenne entre les capteurs à gauche et le mur gauche
    dl=(dfl+drl)/2
    --La distance moyenne entre les capteurs à droit et le mur droit
    dr=(dfr+drr)/2

    -- calcul d'angle d orientation du chemin
    phi=((math.atan((drl-dfl)/a))+(math.atan((dfr-drr)/a)))/2
    --calcul d'angle de rotation de la troisième roue
    gama=k*(dr-dl)
    alpha=gama+phi

    -- les vitesses linéaire des roues gauche et droite
    vl=v*(math.cos(alpha)+(b/e)*math.sin(alpha))
    vr=v*(math.cos(alpha)-(b/e)*math.sin(alpha))

    -- les vitesses angulaires des roues gauche et droite
    wl=vl/R
    wr=vr/R

    --appliquer les vitesses angulaires
    sim.setJointTargetVelocity(Lefet,wl)
    sim.setJointTargetVelocity(Right,wr)
end
```

Figure III.12 Programme de la navigation entre deux murs

III.3.3 Observations et interprétations des résultats de simulation

On remarque dans cette simulation que le robot parcourt le couloir du début jusqu'à la fin en maintenant une position centrale entre les murs du couloir bien que le couloir a une forme quelconque et sa largeur varie d'un endroit à l'autre.

III.4 Simulation de l'algorithme 3 «Suivi d'une ligne avec évitement d'obstacles»

III.4.1 Construction de la scène

Les éléments constituants la scène construite sont donnés sur la figure III.13représentant « Hiérarchie de la scène»

- Le chemin tracé sur le sol (path), obstacles
- 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 du sol sous le capteur.
- Deux capteurs de Proximité (Lidar) qu'on a ajouté au robot Pioneer comme child (Proximity_sensor1, Proximity_sensor2) situés sur le coté droit du robot Pioneer. **[15]**

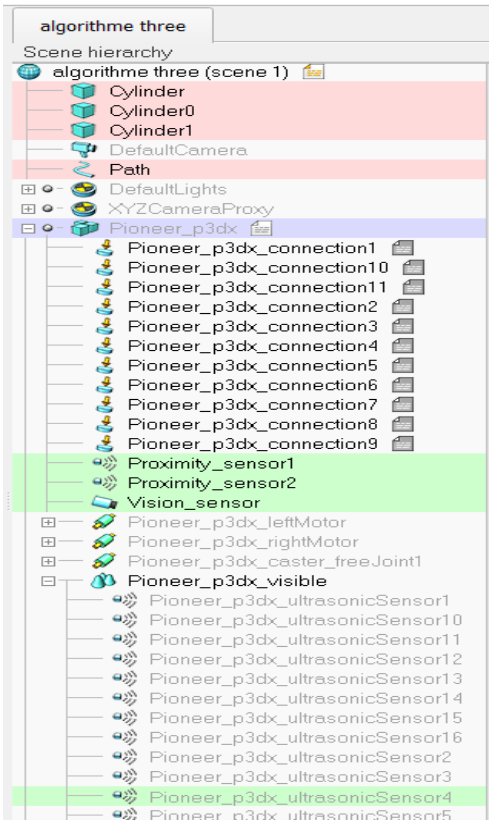
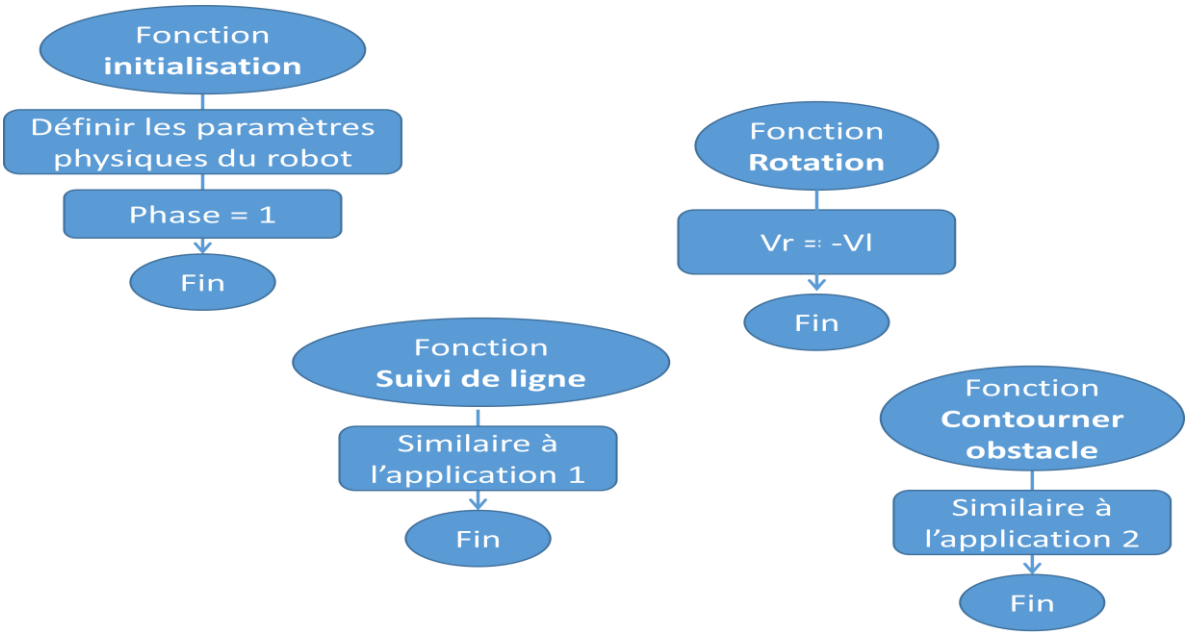


Figure III.13 Fenêtre hiérarchie de la scène construite

III.4.2 Organigramme et programme de simulation

On déduit des schémas de contrôle donné au chapitre II,les organigrammes et le programme suivants.Ce programme est structuré en 5 fonctions. La fonction « actuation » tourne en boucle et fait appel aux quatre autres fonctions. Les détaille des fonctions « suivi de ligne » et « contourner obstacle » ne sont pas donnés dans les organigrammes car ils sont similaires à ceux des applications 1 et 2.

L'organigramme



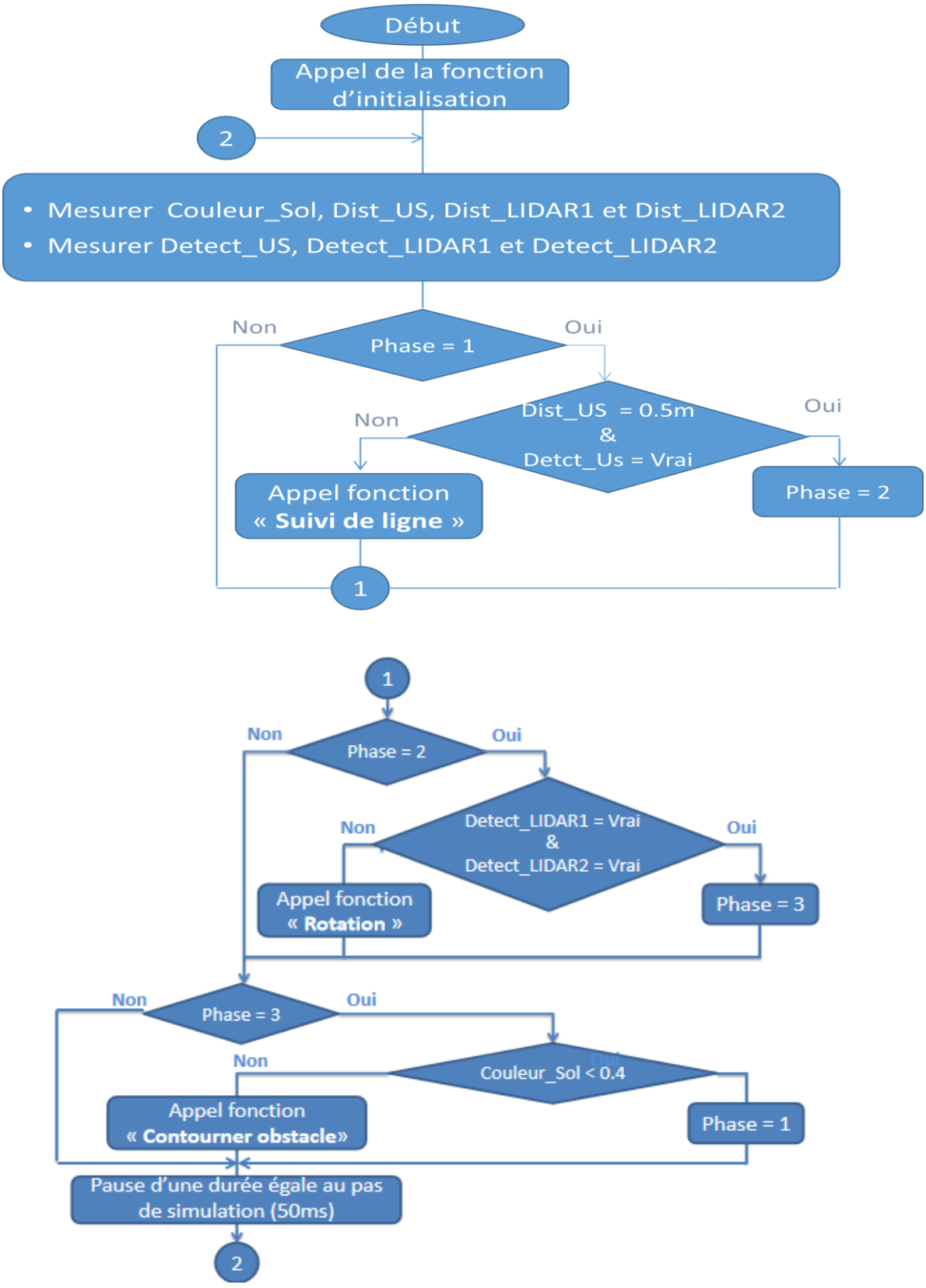


Figure III.14 Organigrammes pour le suivi de ligne avec évitement d'obstacles

Programme

```
function sysCall_init()
    motorLeft=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
    motorRight=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
    ir=sim.getObjectHandle("Vision_sensor")
    sensor=sim.getObjectHandle("Pioneer_p3dx_ultrasonicSensor4")
    sensor0=sim.getObjectHandle("Proximity_sensor1")
    sensor1=sim.getObjectHandle("Proximity_sensor2")
     vmax=0.35 --Vitesse de déplacement du robot
     R=0.0975  --Rayon des roues motrices du robot
     b=0.33/2  --Moitier de la distance entre les deux roues motrices
     e=0.1934  --Distance entre les roues motrices et la roue libre
     a=0.03    --Distance entre les capteurs laser se trouvant sur le même coté
     k1=1.5    --Coefficient du contrôleur suiveur de ligne
     k2=2.5    --Coefficient du contrôleur de contournement d'obstacle
     phase=1   --valeur initiale de phase
end
```

```
function path()
--Vitesse de rotation du robot
 w=-k1*(ng[1]-0.4)
 v=vmax
 --Calcul des vitesse linéaire des roues gauche et droite
 vl=v-(b*w)
 vr=v+(b*w)
end

function rot()
 vl=0.5
 vr=-0.5
end

function obst()
 dr=(dfr+drr)/2   --Distance moyenne entre le robot l'obstacle
 phi=(math.atan((dfr-drr)/a))
 gama=k2*(dr-0.1)
 alpha=-(gama+phi)
 --Calcul des vitesse linéaire des roues gauche et droite
 vl=vmax*(math.cos(alpha)+(b/e)*math.sin(alpha))
 vr=vmax*(math.cos(alpha)-(b/e)*math.sin(alpha))
end
```

```
function sysCall_actuation()
   --Mesure du niveau de gris du sol
   ng=sim.getVisionSensorImage(ir+sim.handleflag_greyscale)
   --Mesure de la distance entre l'obstacle et le capteur US
   detected,dist=sim.readProximitySensor(sensor)
   --Mesure des distances entre l'obstacle et les capteurs laser
   det1,dfr=sim.readProximitySensor(sensor0)
   det2,drr =sim.readProximitySensor(sensor1)

  if phase== 1 --Phase 1 : Suivi de la ligne
  then
    if (detected==1 and dist<0.5) --Condition pour passer de la phase 1 à la phase 2
    then phase=2
    else path()
    end
  end

  if phase==2  --Phase 2 : Le robot tourne sur lui même
  then
    if (det1==1 and det2==1)--Condition pour passer de la phase 2 à la phase 3
    then phase=3
    else rot()
    end
  end

  if phase==3  --Phase 3 : Le robot contourne l'obstacle
  then
    if (ng[1]<0.4) --Condition pour passer de la phase 3 à la phase 1
    then phase=1
    else obst()
    end
  end

  --Appliquer les vitesses linéaires des roues gauche droite
  sim.setJointTargetVelocity(motorLeft,vl)
  sim.setJointTargetVelocity(motorRight,vr)
  end
```

Figure III.15 Programme pour le suivi de ligne avec évitement d'obstacles

III.4.3 Observations et interprétations des résultats de simulation

Dans cette simulation on remarque la présence d'une phase transitoire qui apparait au moment du passage de la phase « contournement d'obstacle » à la phase « suivi de la ligne ». Dans cette phase transitoire le robot ne rejoint pas tout de suite la ligne et dévie un peu pendant un court moment avant de suivre correctement la ligne.

III.5 Conclusion

Les simulations et programmes que nous avons proposé dans ce chapitre ont fonctionné correctement et les résultats obtenues sont satisfaisants.

Conclusion général

En conclusion, ce travail nous a permis d'étudier et d'appliquer trois algorithmes de contrôle de robot mobile : Suiveur de ligne proportionnel, navigation entre deux murs et suivi de chemin avec évitement d'obstacle.

Pour réaliser ces tâches, nous avons utilisé le langage de programmation LUA et le simulateur CoppeliaSim. Ces outils ont grandement simplifié notre tâche.

La réalisation de ce projet a été pour nous l'occasion de découvrir le domaine de la robotique mobile. Cette expérience a été très enrichissante pour notre formation académique car elle nous a permis d'appliquer plusieurs concepts fondamentaux en rapport avec notre spécialité, tels que le contrôle, les capteurs et la programmation.

Comme perspective il serait intéressant que ce travail serve de base pour une application de ces algorithmes sur un robot réel.

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

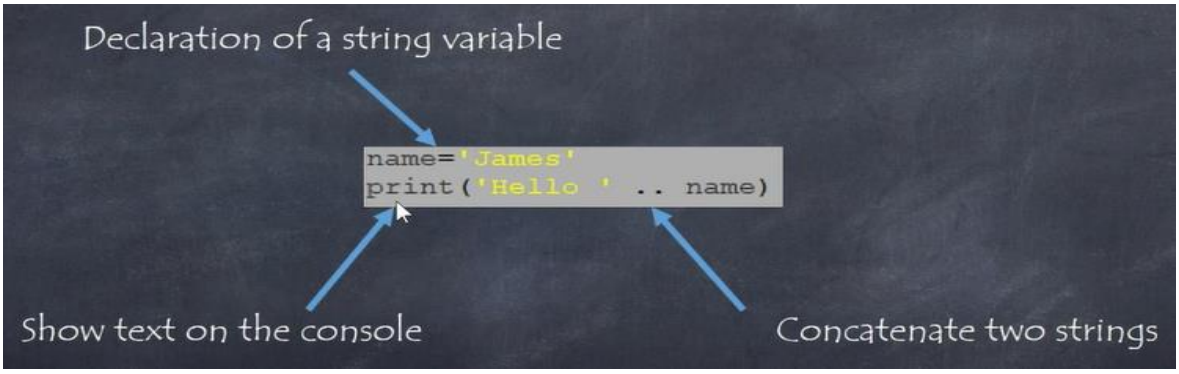
`true` et `false`

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



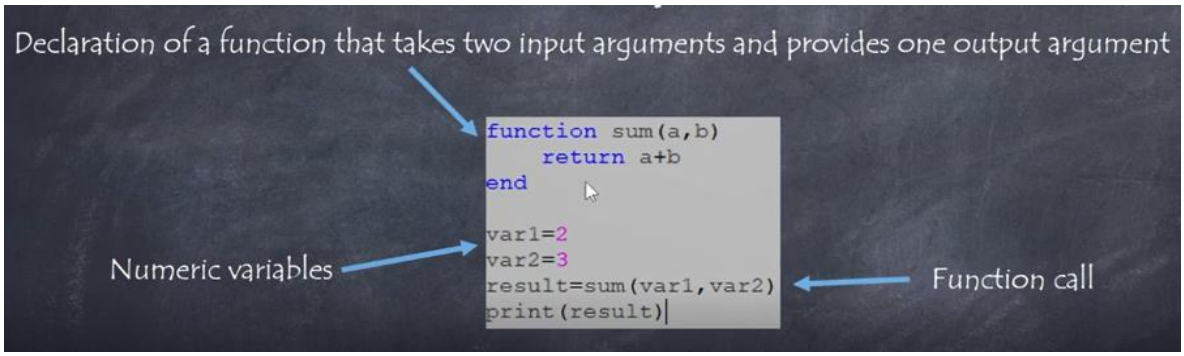
Remarques

- 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.



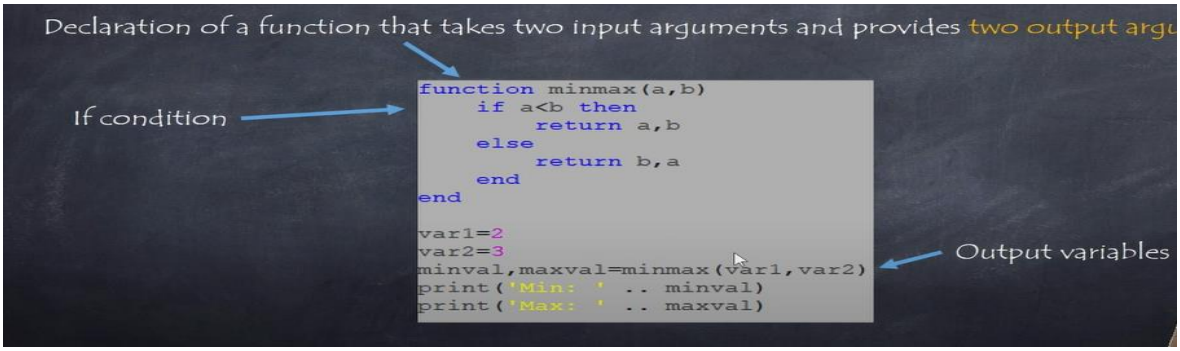
Remarque

- 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
```

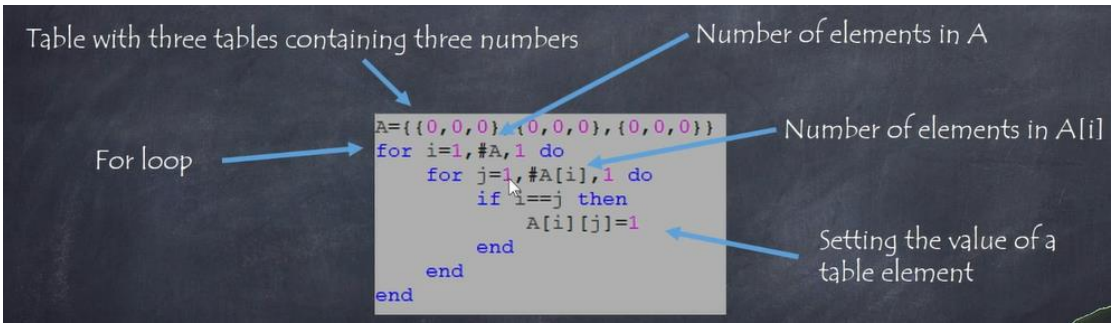
- Syntaxe de l'appel de la fonction :

```
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] Bernard BAYLE « Robotique mobile » Polycopie de cours Université de Strasbourg **2011**

[2] Illah Reza Nourbakhsh, Roland Siegwart « Introduction to Autonomous mobile robots. MIT Press 2011

[3] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, **« Robotics Modelling, Planning and Control »** Springer 2009

[4] Mark W. Spong, Seth Hutchinson, M. Vidyasagar « Robot Modeling and Control » Wiley 2006

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

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

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

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

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

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

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

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

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

[14] 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

[15] Demonstration of Line Tracking with Obstacle Avoidance (Advanced) | CoppeliaSim (V-REP)

https://www.youtube.com/watch?v=G7n35V1fZIk&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=32

[16] Adept mobilerobotics « Pioneer 3 Operations Manual » Revision 6.4 - 2013