

Tutorial pour la prise en main d'un modèle Simulink de véhicules autonomes connectés

Introduction

Ce document permet la prise en main du modèle Simulink qui vous est fourni en début d'enseignement d'intégration pour le volet simulation à savoir suivi_ligne_huit_2022_eleve_2022b.slx (version Matlab 2022b) ou suivi_ligne_huit_2022_eleve_2020a.slx (version Matlab 2020a si vous avez une version de Matlab 2020 ou 2021). Ce modèle est volontairement incomplet. Vous aurez à le compléter et l'enrichir afin de mettre au point vos différents algorithmes à implémenter dans la plateforme physique.

Vous aurez également besoin d'installer la « Mobile Robotics Simulation Toolbox ». Le modèle Simulink utilise un terrain qui décrit un circuit en huit. Le fichier huit.jpg décrit un huit uniquement sans obstacle. Le fichier huit_obstacle.jpg décrit un circuit en huit avec un obstacle (modélisé par un point rouge) placé sur la trajectoire.

Description du modèle bloc par bloc

1) Bloc « Caméra ».

Son rôle est de modéliser la photo brute prise par la caméra. Le modèle est extrêmement simplifié et fait comme si la photo était prise à la verticale et à l'avant du robot (par d'angle de vue).

- Entrées : pos_mon_robot (position du robot), image_sol (« terrain » soit la vue du ciel du circuit où doit évoluer le robot)
- Sorties : image_camera (vue du ciel du champ de vision du robot, rectangle rouge sur Video Display2)

2) Bloc « Télémètre ».

Son rôle est de modéliser le signal de sortie du télémètre à l'avant du robot.

- Entrées : pos_mon_robot (position du robot), image_sol (« terrain » soit la vue du ciel du circuit où doit évoluer le robot)
- Sorties : distance_obstacle (distance de l'obstacle)

3) Bloc « Perception ».

Ce bloc doit analyser les données venant des capteurs pour en extraire les informations d'erreur d'orientation du robot, détection d'un obstacle et détection d'une intersection. Ce bloc permet aussi la génération d'une animation video montrant l'évolution du robot sur le terrain et la vision du robot (figure 1).

- Entrées : pos_mon_robot, image_camera, distance_obstacle, image_sol
- Sorties : erreur_orientation (scalaire modélisant l'erreur signée de l'axe du robot par rapport à celui de la ligne), detect_obstacle (à 1 si obstacle) , detect_inter (à 1 si un croisement est détecté), image_capteurs, image_sol

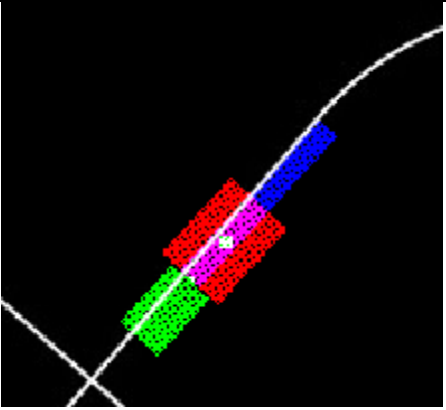
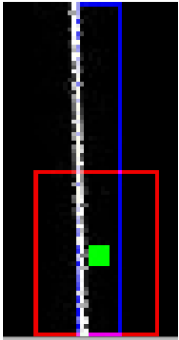
	<p>Dessin du robot ainsi que des zones de vision de la caméra et du télémètre sur le terrain :</p> <ul style="list-style-type: none"> • Robot : rectangle vert • Vision caméra : rectangle rouge • Vision télémètre : rectangle bleu
	<p>Dessin du terrain vu par la caméra et le télémètre :</p> <ul style="list-style-type: none"> • Caméra : à l'intérieur du cadre rouge • Télémètre : à l'intérieur du rectangle bleu <p>Le carré vert modélise graphiquement le suivi de ligne estimé par traitement image ; si la détection de ligne se passe bien ce point doit rester sur la ligne. Ici ce point est représenté à mi-profondeur de la vision de la caméra. Cette modélisation du suivi de ligne peut être modifiée suivant l'algorithme de suivi utilisé.</p>

Figure 1 : Evolution du robot sur le terrain et la vision du robot

4) Bloc « Cerveau_Robot ».

Son rôle est de décrire le comportement souhaité du robot, sous forme d'une machine d'état. Le cerveau peut aussi gérer la communication avec une infrastructure. Les entrées/sorties du cerveau du robot ont été choisies pour permettre un comportement d'arrêt à une intersection, envoi d'un message à l'infrastructure puis redémarrage après réception d'un message de l'infrastructure.

- Entrées : erreur_orientation, detect_obs, detect_inter.
- Sorties : consigne ([vitesse linéaire ; vitesse angulaire])

5) Bloc « Differential Drive Inverse Kinematics » (de la « Mobile Robotics Simulation Toolbox »). Son rôle est de modéliser le mouvement des roues. Il modélise le mouvement des roues à partir d'une consigne double {vitesse linéaire, vitesse angulaire} et en tenant compte de l'état précédent.

- Entrées : vitesse linéaire, vitesse angulaire ([v; w])
- Sorties : vitesse des roues droites et gauche

6) Bloc « Differential Drive Simulation ». Son rôle est de modéliser la position du robot. Il calcule la nouvelle position du robot en fonction du mouvement effectif des roues et de la position et vitesse précédentes des roues.

- Entrées : vitesse des roues droites et gauche ([wL; wR])
- Sorties : position du robot

Remarque : la position initiale (x, y, angle) du robot est à indiquer dans ce bloc « Differential Drive Simulation ».

Informations supplémentaires

Comme pour tout système, maîtriser l'état initial est important. Ici, les positions initiales du robot sont renseignées dans les blocs « Differential Drive Simulation ». Les machines d'état ont un état initial indiqué par un signe significatif.

Il convient également de comprendre comment va fonctionner la simulation à partir de l'état initial. Ici, c'est dans le bloc « Differential Drive Simulation » que nous avons renseigné un « Sample Time » qui cadence la fréquence de mise à jour de la position du robot (ici 0.1 secondes). Ainsi, les autres blocs vont être déclenchés séquentiellement à partir d'un déclenchement régulier basé sur ce « Sample Time ». Il suffit alors de positionner la mise à jour de l'état du cerveau du modèle en « Inherited » ce qui signifie que sa mise à jour dépend de celle du bloc précédent.