Robot - TP2 - Commande - Suiveur de ligne

Installation

Téléchargez et décompressez le fichier robot-simulator-bin.tar.gz contenant une version compilé du de la solution du TP simu.

Depuis https://gitlab.cri.epita.fr/jeremie.graulle/ssie-robot-command créez un "fork" personnel privé.

Ajoutez votre binôme et "jeremie.graulle" en "maintainer".

Clonez ce projet sur votre ordinateur et compilez le ensuite lancez le robot-simu suivi du robot-command, vous devrez voir le robot du simu se déplacer en suivant l'algorithme naïf du robot-command.

Etape 1

Créez une branche `tp2` pour commiter les modifications demandées pour ce TP.

Modifiez le contenu de la boucle `while (true)` de `main.cpp` du projet robot-command avec éventuellement une initialisation avant la boucle pour faire un robot suiveur de ligne.

N'hésitez pas à baisser la vitesse de déplacement ou de rotation du robot pour permettre à l'algo d'être plus stable.

Le but est d'avoir un algorithme très simple contenant le minimum de code pour que le robot fasse le tour de la tache jaune en considérant qu'au début le robot est très proche du bord comme sur le simu après avoir appuyé sur la touche espace.

Commitez et poussez cette première étape dans la branche `tp2` sur votre fork personnel, pour que vous puissiez rapidement la récupérer pour faire un test avec le vrai robot au TP4.

Etape 2

0. Ajoutez le template suivant autour du code de l'étape 1 (en remplaçant `// Step 1 code` par votre code):

```
// #define STEP_1
#ifdef STEP_1
  // Step 1 code
#else
  // Step 2 code
#endif
```

1. En vous inspirant de la branche `motorKeyboard`, en plus de faire le suivi de ligne de l'étape 1, ajoutez une fenêtre SFML pour pouvoir dessiner.

   Attention :

   - Il ne faudra pas prendre la partie déplacement du robot au clavier ou jostick.
   - La fonction `sf::Window::display` fait une attente pour atteindre une certaine valeur de FPS et donc la boucle principale du programme, ne devra pas contenir de fonction bloquante (`Robot::waitChanged` ou `sleep`). Vous pouvez afficher le nombre de FPS et vous devez avoir la même valeur que sur le simu.
   - Si vous avez des erreurs de link pensez à vérifier que vous avez bien récupéré toutes les modifications en regardant le dernier commit de la branche `tp2-motorKeyboard`.

2. En vous inspirant du `main.cpp` du simulateur, ajoutez l'affichage d'un carré bleu de 20 pixels représentant le robot.

3. Mettez à jour la position de ce rectangle bleu à l'aide des capteurs de roues codeuses en vous inspirant des formules utilisées pour mettre à jour la position du robot dans le simu :

- Il faut récupérer les données depuis les capteurs de roue codeuse (en utilisant la fonction `getEncoderWheelValue()` avec comme indice 0 et 1) donnant la distance parcourue de chaque roue en nombre "fente" de la roue codeuse.
- Vous devrez ensuite convertir ce nombre de pas en pixel en créant des constantes avec la même valeur que dans le simu (vous avez déjà eu besoin de ces valeurs pour le TP1) :
  - Nombre de "fente" de la roue codeuse par tour

  - Diamètre de la roue
- Appliquez une formule très proche de la formule utilisée dans le TP1 pour l'update du robot en créant une constante avec la même valeur que dans le simu (que vous avez également déjà eu besoin pour le TP1) :
  - Distance entre les 2 roues
- Attention les roues codeuses ne font pas la différence entre aller en avant ou en arrière donc vous supposerez que le robot se déplace toujours vers l'avant et donc dans vos tests (controle au clavier ou par le code) penser à ne jamais donner de puissance négative à aucune roue.

4. Utilisez un `sf::VertexArray` avec des primitives de type `sf::Points` pour afficher l'ensemble des anciennes positions du robot (limité à 1000 pixels visibles à l'écran, si le robot ne se déplace pas, tous les anciens pixels doivent rester affiché, attention à l'arrondi) permettant d'afficher une forme ressemblant à celle dessinée au sol du simulateur.

   Attention : Il y a une limitation dans la classe `sf::VertexArray` qui fait que l'on ne peut pas coder la solution idéale, mais écrivez un code qui s'en rapproche le plus possible en rajoutant en commentaire ce qui manquerait pour faire la solution idéale.

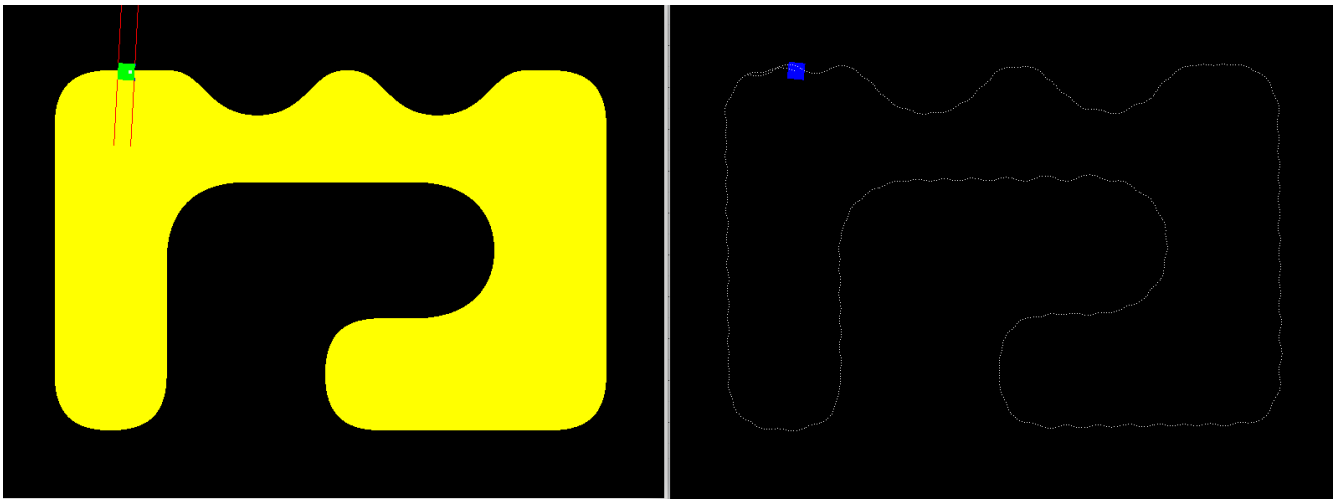

Figure 1: Exemple de résultat attendu