SY31
TP3 - Localisation par odométrie

7 novembre 2012

1 Introduction

L'objectif de ce TP est de reconstruire la trajectoire du Wifibot en utilisant les informations de vitesse issues des 2 codeurs de roues situés sur les moteurs avant du robot. Pour cela, vous allez développer un module RTMaps prenant comme entrée les vitesses de roues et donnant en sortie l'odométrie : la position x, y et l'orientation θ relatives du robot.

1.1 Disposition des roues et centre instantané de rotation

C'est la combinaison du choix des roues et de leur disposition qui confère à un robot son mode de locomotion propre. Sur les robots mobiles, on rencontre principalement trois types de roues :

– les roues fixes dont l'axe de rotation, de direction constante, passe par le centre de la roue ;

– les roues centrées orientables, dont l'axe d'orientation passe par le centre de la roue ;

– les roues décentrées orientables, souvent appelées roues folles, pour lesquelles l'axe d'orientation ne passe pas par le centre de la roue.

Pour qu'une disposition de roues soit viable et n'entraîne pas de glissement des roues sur le sol, il faut qu'il existe pour toutes ces roues un unique point de vitesse nulle autour duquel tourne le robot de façon instantanée. Ce point, lorsqu'il existe, est appelé centre instantané de rotation (CIR). Les points de vitesse nulle liés aux roues se trouvant sur leur axe de rotation, il est donc nécessaire que le point d'intersection des axes de rotation des différentes roues soit unique. Pour cette raison, il existe en pratique certaines catégories de robots mobiles à roues. Dans notre cas, nous avons choisi pour le Wifibot, une configuration de type unicycle (Fig. 1).

1.2 Modélisation

Centre instantané de rotation Les roues motrices ayant même axe de rotation, le CIR du robot est un point de cet axe. Soit ρ le rayon de courbure de la trajectoire du robot, c'est-à-dire la distance du CIR au point O' (voir Fig. 1). Soit L l'entre-axe et ω la vitesse de

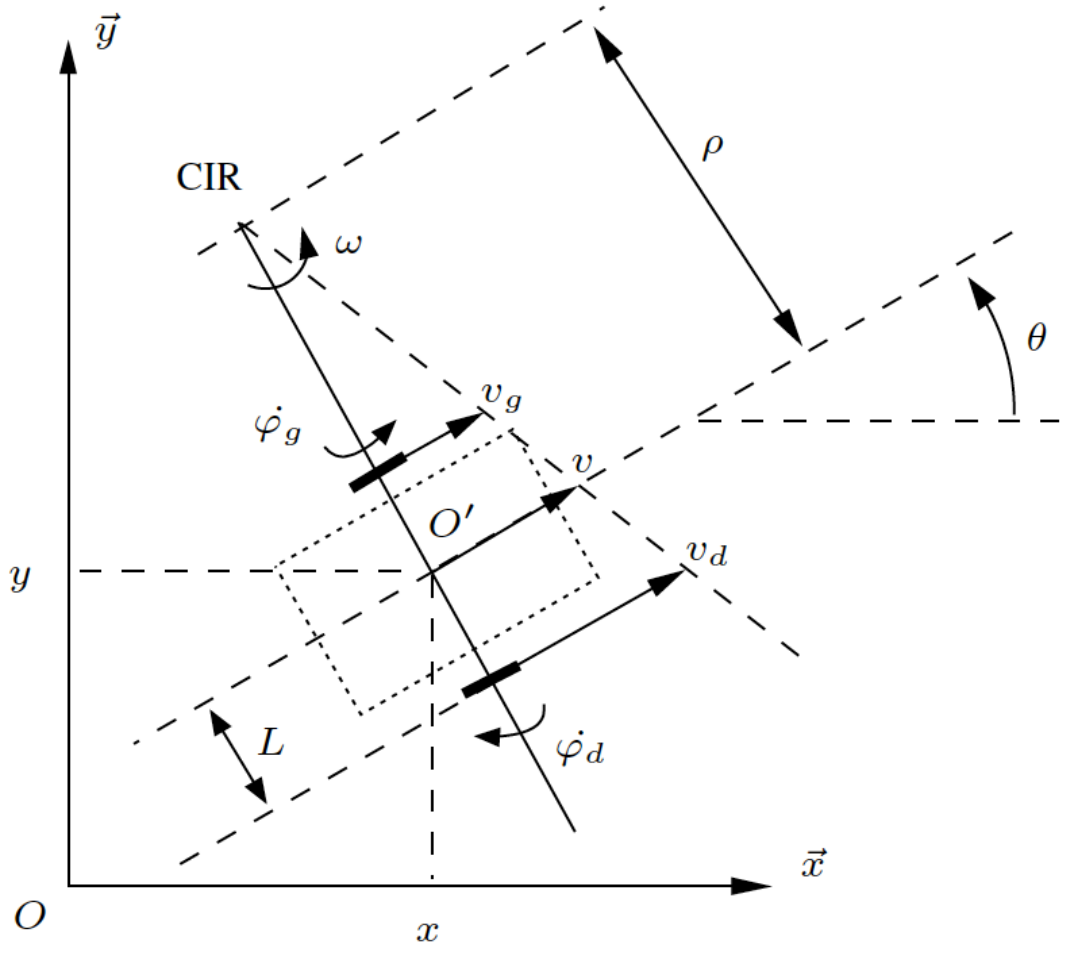

FIGURE 1 – Centre instantané de rotation d'un robot de type unicycle

rotation du robot autour du CIR. Alors les vitesses des roues droite et gauche, respectivement notées v_d et v_g et définies à la Fig. 1, vérifient :

$$v_d = -r.\dot{\varphi}_d = (\rho + L)\,\omega \tag{1}$$

$$v_g = r.\dot{\varphi}_g = (\rho - L)\,\omega \tag{2}$$

ce qui permet de déterminer ρ, v et ω à partir des vitesses des roues :

$$\rho = L\frac{\dot{\varphi}_d - \dot{\varphi}_g}{\dot{\varphi}_d + \dot{\varphi}_g} \tag{3}$$

$$v = \frac{v_d + v_g}{2} = \frac{r\,(\dot{\varphi}_g - \dot{\varphi}_d)}{2} \tag{4}$$

$$\omega = \dot{\theta} = -\frac{r\,(\dot{\varphi}_d + \dot{\varphi}_g)}{2L} \tag{5}$$

L'équation 3 permet de situer le CIR sur l'axe des roues. Par ailleurs ces équations expliquent deux propriétés particulières du mouvement des robots de type unicycle : si $\dot{\varphi}_d = -\dot{\varphi}_g$, le robot se déplace en ligne droite ; si $\dot{\varphi}_d = \dot{\varphi}_g$, alors le robot effectue un rotation sur lui-même. L'utilisation de ces deux seuls modes de locomotion, bien que limitée, permet de découpler les mouvements et de fournir une solution simple pour amener le robot d'une posture à une autre. C'est sans doute là une des raisons du succès de ce type de robots.

L'intégration numérique de ω permet d'obtenir l'orientation θ du robot :

$$\theta = \frac{-r\,(\varphi_d + \varphi_g)}{2L} + \theta_0 \tag{6}$$

où θ_0 est la valeur de θ à l'initialisation (les angles des roues, généralement mesurés par des codeurs incrémentaux, étant alors choisis nuls).

2 Travail à réaliser

2.1 Vitesses des roues $\dot{\varphi}_d$ et $\dot{\varphi}_d$

1. Développer un diagramme RTMaps permettant d'afficher les vitesses des roues, utilisez votre ancien diagramme qui permettait le déplacement au clavier.

2.2 Vitesse linéaire v et de rotation ω

1. Développer un module RTMaps permettant de calculer la vitesse linéaire v (en m/s) et la vitesse angulaire ω (en rad/s) à partir des vitesses de roues. Pour cela vous aurez besoin de mesurer le demi-entraxe L et le rayon des roues r.

2.3 Odométrie : position x, y et orientation θ du robot

La reconstruction de la trajectoire du robot se fera par intégration des positions élémentaires au cours du temps suivant les équations suivantes :

$$\begin{array}{rcl} x(k+1) & = & x(k) + v(k).dt.cos\theta(k) \\ y(k+1) & = & y(k) + v(k).dt.sin\theta(k) \\ \theta(k+1) & = & \theta(k) + \omega(k).dt \end{array} \quad (7)$$

où dt est la période d'échantillonnage non constante.

1. Proposer un code C permettant de calculer θ par intégration numérique de ω.
2. Calculer x et y par intégration numérique, à l'aide de v et θ précédemment calculés.
3. Afficher la position et l'orientation du robot.
4. Tester l'odométrie en faisant déplacer le robot au sol et valider le résultat.
5. Faire plusieurs essais et essayez de quantifier la précision de l'approche.

3 Développement

3.1 Création du paquet RTMaps

La création d'un composant RTMaps se fait via le logiciel Visual Studio 2010. La procédure de compilation est simple, dans un premier temps Visual Studio compile votre code source afin de créer une dll. Un utilitaire va ensuite convertir cette dll en paquet RTMaps. Vous trouverez la solution à remplir dans le répertoire du TP : *C :\data\tp-sy31\odometry*. Une fois la solution ouverte, vous pouvez compiler le projet en choisissant Génerer > Générer la solution. Un moyen plus rapide consiste à tout simplement utiliser le raccourcis clavier F7. Le paquet généré se trouve dans le répertoire package, situé à la racine de la solution.

3.2 RTMaps

Au cours de ce TP vous allez devoir remplir du code C++, vous ne devriez pas avoir à modificer le code RTMaps. Le seul objet RTMaps que vous aurez à manipuler est celui permettant de récupérer le timestamp en cours. Ce dernier vous permettra de calculer le temps écoulé depuis la dernière frame. Voici un exemple d'utilisation de cet objet, faites bien attention à la base de temps !

```
1 // MAPS::CurrentTime(): returns the RTMaps date in microsec.
2 double t = MAPS::CurrentTime();
```

3.3 Organisation

Les fichiers à remplir sont :
- odometry-component.cpp : récupère les vitesses et demande la mise-à-jours de l'odométrie
- wifibot.cpp : calcul l'odométrie du robot

Les autres fichiers sont pour le viewer OpenGL qui vous permettra de vérifier l'implémentation de votre algorithme. Avant de commencer à remplir le code, assurez-vous bien que votre solution compile.