Exercices de robotique mobile ENSI-3, 2007-08

Luc Jaulin

BE QT1 : Simulation graphique

Sous LINUX, lancer QTCREATOR

A) **Graphisme**

1. Créer un nouveau projet C++ avec GUI dont la classe de base est de type `QDialog`.
2. Mettre sur la fiche un *display widget* de la classe `QGraphicsView`. Par défaut, cet objet s'appelle `graphicsView`
3. Inclure dans le header de `dialog` les modules `QtCore` et `QtGui`.
4. Créer dans la classe `Dialog` un pointeur membre privé `S` sur `QGraphicsScene`. Déclarer un pointeur privé `E` sur `QGraphicsEllipseItem`.
5. Dans le constructeur de `Dialog`, créer la scène en faisant `S=new QGraphicsScene(this)`. Mettre cette scène dans le `graphicsView` en faisant `ui->graphicsView->setScene(S)`.
6. Ajouter une ellipse sur la scène en faisant `E=S->addEllipse(10,10,50,100);`
7. En déclarant un stylo bleu `QPen bluePen(Qt::blue)` et un pinceau rouge `QBrush redBrush(Qt::red)`, changer la couleur de l'ellipse.
8. Retracer cette ellipse en avec un stylo d'épaisseur de 5.
9. Tracer un rectangle `R` sur cette même figure. Rendre ce rectangle mobile à la souris en faisant `R->setFlag(QGraphicsItem::ItemIsMovable)`.
10. Tracer un polygone. On s'influencera de la suite d'instructions suivantes. `QPolygonF p1; p1 << QPointF(0,20)<<QPointF(20, -20)<<QPointF(10, 0); P=S->addPolygon(p1)`.

B) **Interface**. Mettre un `PushButton` sur la fiche. Lorsqu'on clique dessus, il tourne le polygone.

C) **Simulation**. Simuler un système de type char décrit par des équations d'état. Le robot sera représenté par un polygone.

Aide

1. Utiliser `setPos` et `setRotation` pour positionner un objet `Qgraphic` en absolu.
2. Pour régler les échelles d'un objet `QGraphicsView`, utiliser le *designer*.
3. Pour passer de radian en degrés (pour la rotation), multipliez par `180/M_PI`.

BE QT 2: Régulation

Sous LINUX, lancer QTCREATOR. On partira du projet

`www.ensta-bretagne.fr/jaulin/qt_be_control_start.zip`.

qui reprend en gros ce qui devait être fait au BE précédent (sauf que la classe de base est désormais `MainWindow` à la place de `QDialog`). Pour ce BE, Il est conseillé d'aller visionner les films `voidrealms` associés aux *timers* et aux *signaux/slots* (voir `www.ensta-bretagne.fr/jaulin/qt.html`).

A) **Timer**

1) Déclarer dans la classe `MainWindow` un pointeur `timer` sur la classe `QTimer` Créer ce timer dans le constructeur de `MainWindow` en faisant `timer = new QTimer(this)`.

2) Connecter ce timer au *slot* `Clock` de `MainWindow` par l'instruction

```
connect(timer,SIGNAL(timeout()),this,SLOT(Clock())).
```

Initialiser la période du timer à 10 ms par `timer->start(10)`. Lancer la simulation.

B) **Interactivité**

Déclarer la fonction `void keyPressEvent(QKeyEvent *event)` dans `MainWindow`. Programmer une téléopération du robot au clavier. On utilisera `switch (event->key())` et `case Qt::Key`.

C) **Régulation**

On veut maintenant réguler le robot autour d'une cible d'équation

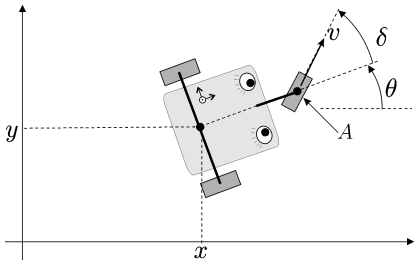
$$\mathbf{w}(t) = \begin{pmatrix} R_x \cos t \\ R_y \sin t \end{pmatrix}.$$

Créer une classe `controller` avec pour variables `u1` et `u2` et une méthode `Clock(x,y,theta,v)`. Avec le clavier, vous devez pouvoir agir sur les variables `Rx` et `Ry` qui seront considérées comme des consignes.

BE QT 3: OpenGL - le tricycle

Sous LINUX, lancer QTCREATOR. Reprennez le projet du BE QT sur `www.ensta-bretagne.fr/jaulin/qt_be3_start.zip`

On considère le tricycle représenté sur la figure ci-dessous.



Le conducteur du tricycle possède deux commandes : l'accélération de la roue avant et la vitesse de rotation du volant. Les variables d'état de notre système sont constituées par les coordonnées de position, (les coordonnées x, y du centre de l'essieu arrière, l'orientation θ du tricycle, et l'angle δ de la roue avant) et la vitesse v du centre de la roue avant. L'équation d'évolution du tricycle s'écrit

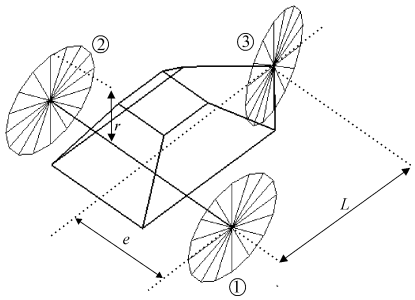
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v\cos\delta\cos\theta \\ v\cos\delta\sin\theta \\ \frac{v\sin\delta}{L} \\ u_1 \\ u_2 \end{pmatrix}$$

où $L = 3$ est l'écart entre l'essieu arrière et le centre de la roue avant. Le vecteur d'état est ici

$$\mathbf{x} = (x, y, \theta, v, \delta)^{\mathrm{T}}.$$

1) Modifier `robot3d::Clock` afin que le robot obéisse au modèle ci-dessus.

2) On souhaite un tricycle en 3D avec les roues, comme sur la figure.



On notera e la distance entre chaque roue arrière et l'axe du tricycle. On prendra $e = 2$m. Le rayon des roues sera noté r et on prendra $r = 1$m. On rajoute pour cela trois variables d'état α_1, α_2, α_3 correspondant à la rotation propre de chaque roue (numérotées de 1 à 3 comme sur la figure). En utilisant la règle de composition des vitesses, calculer les vitesses que doivent posséder chacune des roues.

3) Quelles sont les nouvelles équations d'état du tricycle.

4) Dans `WidgetSimu`, faites une méthode `DrawRoue(x,y,delta,alpha)` qui dessine une roue avec ses rayons. On utilisera la structure `GL_LINE_LOOP`.

5) Faite évoluer le tricycle avec ses trois roues.

BE QT 4: OpenGL - Régulation du tricycle

Sous LINUX, lancer QTCREATOR. Reprennez le projet du BE QT sur `www.ensta-bretagne.fr/jaulin/qt_be4_start.zip` pour le tricycle.

1) Passer quelques minutes à bien comprendre l'ensemble du programme.

2) Calculer un régulateur qui permette de décrire une cycloïde d'équation

$$\begin{array}{rcl} x_d(t) & = & R\sin(f_1 t) + R\sin(f_2 t) \\ y_d(t) & = & R\cos(f_1 t) + R\cos(f_2 t), \end{array}$$

avec $R = 15$, $f_1 = 0.02$ et $f_2 = 0.12$. Pour faire cette commande, on utilisera une méthode de linéarisation par bouclage et on prendra comme sortie le centre de la roue avant (afin d'éviter les singularités). Il faudra aussi dessiner la cible (x_d, y_d). On rappelle que les équations d'état du tricycle sont données par

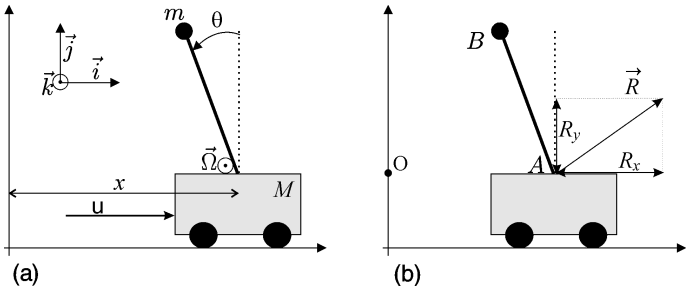
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v\cos\delta\cos\theta \\ v\cos\delta\sin\theta \\ \frac{v\sin\delta}{L} \\ u_1 \\ u_2 \end{pmatrix}.$$

3) Mettre n robots à suivre sur cette cycloïde. Ces n robots doivent être régulièrement répartis sur cette cycloïde. On s'arrangera pour que les n cibles n'entrent jamais en collision.

4) Pour la phase d'initialisation, rajouter un asservissement réflexe pour éviter les collisions entre les robots.

BE QT 5: Pendule inversé

Considérons le pendule inversé, formé d'un pendule posé en équilibre instable sur un chariot roulant, comme représenté sur la figure.



Modèle dynamique. La quantité u est la force exercée sur le chariot de masse M, x indique la position du chariot, θ est l'angle entre le pendule et la verticale. Les équations d'état s'écrivent sous la forme

$$\frac{d}{dt}\begin{pmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{\theta} \\ \frac{-m\sin\theta(\ell\dot{\theta}^2 - g\cos\theta)}{M+m\sin^2\theta} \\ \frac{\sin\theta((M+m)g - m\ell\dot{\theta}^2\cos\theta)}{\ell(M+m\sin^2\theta)} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{M+m\sin^2\theta} \\ \frac{\cos\theta}{\ell(M+m\sin^2\theta)} \end{pmatrix} u.$$

1) Linéariser ce pendule et proposer un retour d'état pour avoir une commande en position de ce système. On placera tous les poles en -2. Pour les coefficients, on prendra $m = 1; M = 5; \ell = 1; g = 9.81$;

2) Sous LINUX, lancer QTCREATOR. Vous partirez du projet du BE QT sur `www.ensta-bretagne.fr/jaulin/qt_be5_start.zip`. Simuler le système bouclé.

3) Reprendre les équations d'état du pendule inversé, mais prenons comme entrée, non plus la force, mais l'accélération $a = \ddot{x}$. Ecrire le modèle cinématique à partir du modèle dynamique.

4) En pratique, nous pouvons passer du modèle dynamique d'entrée u au modèle cinématique d'entrée a en calculant u par une *commande proportionnelle* de type *grand gain* de la forme

$$u = K\,(a - \ddot{x})$$

avec K très grand et où a est une nouvelle entrée. L'accélération $\ddot{x}$ peut être mesurée par un accéléromètre. Si K est suffisamment grand, nous aurons bien évidemment la commande u que nous donnera l'accélération a voulue, c'est-à-dire que l'on pourra admettre que $\ddot{x} = a$. Ainsi, le système pourra se décrire par le modèle cinématique qui ne font intervenir aucun des paramètres inertiels du système. Pour l'implémentation de la commande, il ne faut pas chercher à exprimer $\ddot{x}$, mais plûtot à mesurer $\ddot{x}$. C'est cette mesure qui nous permet d'avoir une commande indépendante des paramètres dynamiques. Implémenter cette commande et vérifier qu'on a bien $a = \ddot{x}$. On prendra $a = \sin t$ pour cette vérification.

5) *Commande linéarisante.* Cherchons à faire osciller le pendule de gauche à droite avec un angle désiré de la forme

$$w = \sin t.$$

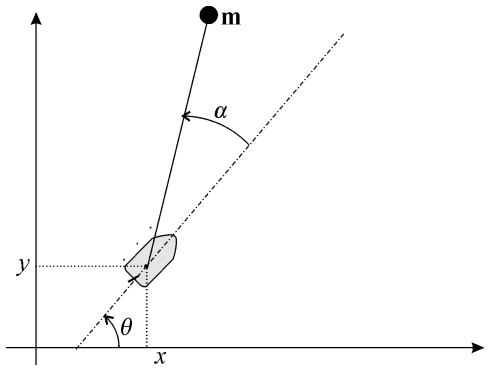
Appliquer pour cela une commande linéarisante. On se basera sur le modèle cinématique.

BE QT 6: Localisation goniométrique instantanée

On considère un robot $\mathcal{R}$ décrit par les équations d'état de type char suivantes.

$$\left\{\begin{array}{rcl} \dot{x} &=& v\cos\theta \\ \dot{y} &=& v\sin\theta \\ \dot{\theta} &=& u_1 \\ \dot{v} &=& u_2, \end{array}\right.$$

où v est sa vitesse du robot, θ son orientation et (x, y) les coordonnées de son centre. Son vecteur d'état est donné par $\mathbf{x} = (x, y, \theta, v)$. Dans l'environnement du robot se trouve un amer $\mathbf{m} = (x_m, y_m)^{\mathrm{T}}$ ponctuel dont on connaît la position. Le robot $\mathcal{R}$ est équipé de cinq capteurs : une caméra omnidirectionnelle (qui lui permet de mesurer l'angle α de la direction de l'amer), des odomètres pour mesurer sa vitesse, une boussole qui lui donne son cap θ, un accéléromètre pour u_2 et un gyromètre pour u_1.



1) Simuler ce système dans une situation où le robot bouge autour de $\mathbf{m}$ afin de générer les signaux $\alpha, \dot{\alpha}, \theta, v, u_1, u_2$. On supposera que l'amer suit l'équation suivante

$$\left\{\begin{array}{rcl} x_m(t) &=& 2\cdot\cos t \\ y_m(t) &=& 3\cdot\sin t \end{array}\right.$$

Pour la génération de α, on utilisera la fonction arc-tangente à deux arguments. La fonction atan2(b, a) renvoie l'argument du vecteur de coordonnées (a, b). Pour la génération de $\dot{\alpha}$ on pourra utiliser le fait que

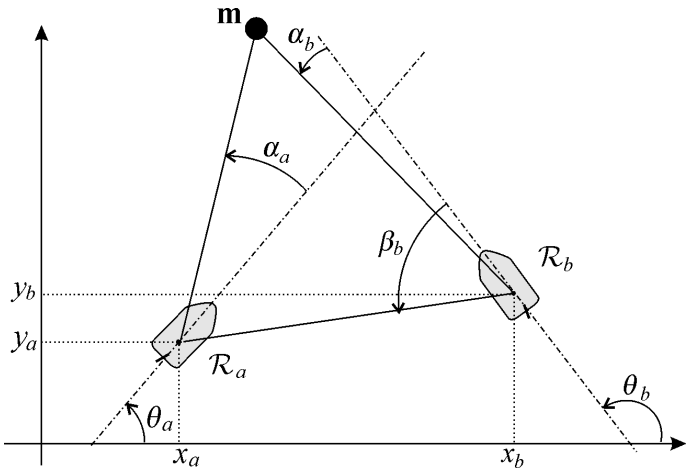
$$\frac{\partial \mathrm{atan}(b,a)}{\partial a} = -\frac{b}{a^2+b^2} \qquad \text{et} \qquad \frac{\partial \mathrm{atan}(b,a)}{\partial b} = \frac{a}{a^2+b^2}.$$

Pour la simulation, vous pouvez vous aider du programme suivant

`www.ensta-bretagne.fr/jaulin/qt_be6_start.zip`

2) Concevoir un système localisation instantanée pour $\mathcal{R}$. Vérifier votre système de localisation sur simulation.

3) On rajoute un deuxième robot $\mathcal{R}_b$ sur la scène qui est du même type que que le premier robot (que nous appelons maintenant $\mathcal{R}_a$) sauf qu'il ne possède pas d'odomètre contrairement à $\mathcal{R}_a$. Le robot $\mathcal{R}_b$ peut voir $\mathcal{R}_a$ (voir figure ci-dessous) ce qui lui donne un angle β_b. Les deux robots peuvent communiquer par WIFI. Concevoir un système de localisation pour le robot $\mathcal{R}_b$ qui lui permet aussi de retrouver sa vitesse.


m
α_b
α_a
β_b
$\mathcal{R}_b$
y_b
y_a
$\mathcal{R}_a$
θ_b
θ_a
x_a
x_b