Travaux pratiques sous HPP

Steve Tonneau et Florent Lamiraux
CNRS-LAAS, Toulouse, France

Introduction

L'objectif de ce TP est d'implémenter un algorithme de planification de mouvement en utilisant le framework HPP (Humanoid Path Planner). HPP regroupe un ensemble d'algorithmes utilisés de manière classique pour des robots complexes, tel que l'humanoïde HRP-2. Il propose aussi un outil, le gepetto-viewer, utilisé pour visualiser les trajectoires calculées pour le robot. Grâce à une interface en python, l'outil est assez facile d'utilisation.

Dans une première (courte) partie du TP, vous allez utiliser l'interface python en tant qu'utilisateur pour résoudre un problème de planification pour le robot PR-2.

Dans une deuxième partie, vous allez devoir implémenter, puis tester, votre propre algorithme de planification de mouvement en utilisant les interfaces de HPP.

1 Mise en place

HPP et les tutorials sont installés dans une machine virtuelle, sur laquelle vous allez devoir travailler. La machine virtuelle émule un système d'exploitation Ubuntu 14.04, sur lequel est installé HPP.

Vous allez travailler sur un dossier partagé entre votre machine physique et la machine virtuelle, pour plus de sécurité.

Pour ce faire, créez un dossier local sur votre machine portant votre nom :

```
$ mkdir ~/tp_hpp_xxx_share
```

Ensuite, lancez la machine virtuelle. Ouvrez un terminal et tapez la commande :

```
$ virtualbox
```

Dans l'interface virtualbox sélectionnez la machine virtuelle u1404x32_1 et cliquez sur "configuration", puis sur "shared folders". Ajouter le dossier crée, et cochez la case "automount".

Ensuite, lancer la machine virtuelle u1404x32_1 en la sélectionnant puis en cliquant sur "Start".

Une fois la machine lancée, ouvrez y un terminal et montez votre dossier dans le répertoire *home/student/dev/hpp/src*

```
$ cd ~/dev/hpp/src
$ cd mkdir tp_hpp_xxx
$ sudo mount -t vboxsf tp_hpp_xxx_share ~/dev/hpp/src/tp_hpp_xxx
```

le mot de passe du compte student est student.

2 Prise en main de l'interface python HPP

2.1 Tutorial 1 HPP

Dans un navigateur, ouvrez la page /dev/hpp/install/share/doc/hpp-doc/index.html et lisez la présentation du logiciel. Ensuite, cliquez à gauche de la page sur "Tutorial", puis "Tutorial 1". Suivez les instructions du tutorial. Le mouvement final peut se jouer de manière assez lente en raison de la machine virtuelle.

2.2 Définition d'un nouveau problème en python

A partir du script du tutoriel 1, nous allons définir un nouveau problème. Pour cela il faut d'abord récupérer les données du nouveau tp.

```
$ cd ~/dev/hpp/src/tp_hpp_xxx
$ git clone --recursive https://github.com/laastp/hpp_aip.git
```

Ensuite, compiler le projet avec cmake

```
$ cd ~/dev/hpp/src/tp_hpp_xxx/build
$ cmake ..
$ make
```

A ce stade, appeler l'encadrant pour modifier la variable d'installation du paquet. Une fois compilé et installé, un nouveau serveur est disponible dans la console : hpp-tp-rrt. Dans tout le reste du tp, on lancera ce serveur à la place de hppcorbaserver pour tester les réponses aux questions.

2.3 Définition d'un nouveau problème en python

Question 1 : En s'inspirant du tutoriel 1, compléter le script situé dans tp_hpp_xxx/script pour charger le robot "buggy" et la scene "scene". Le robot buggy est un robot qui peut se déplacer en 2 dimensions, et possède en plus un angle d'orientation, donné par son cosinus et son sinus. On ne considère pas les autres degrés de liberté pour l'instant.

Définir une position de départ en [-3.7, -4] et la position d'arrivée en [15,2], calculer et jouer la solution.

2.4 Implémentation d'un nouveau plannificateur de mouvement en c++

Dans la suite du TP, nous allons maintenant utiliser l'API HPP pour définir un nouveau planificateur de mouvement. Vous allez implémenter l'algorithme RRT vu en cours. Tout au long de l'implémentation, les classes à utiliser vous sont données. C'est à vous d'utiliser la documentation HPP pour parcourir les méthodes et appeler celles qui vous intéressent. Un squelette vous est fourni, qu'il vous faut remplir.

Environnement de programmation Bien que cela ne soit pas obligatoire, il est conseillé d'utiliser l'IDE QTcreator pour travailler sur ce tp. QTcreator se lance avec la commande

```
$ qtcreator&
```

Pour ouvrir le projet, choisissez File...Open file or Project, et choisissez le fichier "CMakeLists.txt" qui se trouve à la racine. Dans l'invite qui apparaît, choisissez le répertoire **build** créé précédemment et non pas le répertoire par défaut. La touche F4 permet de naviguer entre .hh et .cc, et la touche F2 vous permet de vous rendre à l'endroit où est définie une variable ou un type sous le curseur de la souris.

Question 3 : Commencez par implémenter la méthode *shoot* de la classe ShooterTp (shooter-tp.cc). Il s'agit d'échantilloner uniformément chaque articulation (joint) du robot.

Question 4 : Implémenter la méthode *oneStep* de la classe PlannerTp (planner-tp.cc). *oneStep* correspond à une itération de l'algorithme RRT, c'est à dire à l'échantillonage d'une configuration, et, si possible, à sa connexion avec des éléments de la roadmap. *oneStep* appelle la méthode *extend* que vous devez également implémenter. Attention, dans HPP, chaque noeud créé est associé automatiquement á une composante connexe. Lorsque des noeuds de composantes connexes différentes sont connect'es, ces composantes connexes sont automatiquement fusionnées.

Pour vérifier qu'un chemin est (partiellement) valide, vous pouvez utiliser directement l'objet "pathValidation" défini dans la méthode.

Question 5 : Tester votre planner. Pour cela modifier le script que vous avez choisi pour lancer votre planner. Le nom du planner que vous avez défini est renseigné dans le fichier main.cc.

Question 6 : Tester l'algorithme d'optimisation *RandomShortcut* avec votre planner, et vérifier qu'il fonctionne.

Question 7 : Biaiser la méthode d'échantillonnage pour qu'une fois sur dix, elle retourne la configuration d'arrivée. Cela a t'il une influence sur le temps de calcul ? Le chemin trouvé ? Pourquoi ?

Question 8 : Implémenter votre propre méthode de validation de chemin, en utilisant une méthode itérative vue en cours. Pour simplifier le travail, une free function *validate* a été définie dans planner-tp.cc. Il vous faut la compléter, puis modifier votre planner pour appeler cette méthode.

Question 9 : Modifier votre planner pour implémenter cette fois l'algorithme PRM. Quelles sont les différences entre les deux algorithmes ?