Introduction au Middleware MOOS

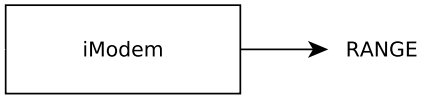
Robotique/UV5.7 - Session MOOS-IvP #1 - Décembre 2018

Supports de cours disponibles sur
`www.simon-rohou.fr/cours/moos-ivp`

1 Implémentation d'une application MOOS

On se place dans le cas d'un véhicule sous-marin cherchant à se localiser par rapport à une balise fixe émettant des signaux acoustiques. Les données perçues par le sous-marin sont des informations de distance (approche *range-only*).

Un modem acoustique est un capteur permettant, notamment, de calculer des distances sous l'eau. Dans la pratique, une application MOOS nommée `iModem` ferait l'interface entre ce capteur et des algorithmes de localisation embarqués. Une variable `RANGE` serait publiée dans la MOOSDB à chaque réception de signal.



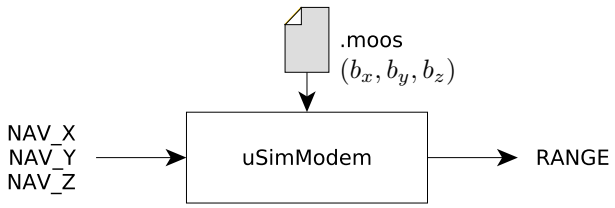
En simulation, il faut représenter cette acquisition de données.

Préparation d'une communauté MOOS :

1. Dans le répertoire `./moos-ivp-extend/missions`, créer un nouveau fichier `.moos` définissant une nouvelle communauté MOOS. Pour l'instant, seule la MOOSDB et l'application de visualisation `uMS` nous intéressent. Exécuter la mission avec la commande `pAntler`.
2. Utiliser le programme `uPokeDB` pour donner une valeur aux variables MOOS `NAV_X`, `NAV_Y`, `NAV_Z`. Ces trois variables représentent la position exacte du sous-marin et ne sont connues qu'en simulation.

Simulation du capteur :

À l'aide du programme `GenMOOSApp_AppCasting`, créer une nouvelle application MOOS nommée `uSimModem` et.. simulant un modem. Le capteur simulé est fixé à un véhicule sous-marin qui se déplace. On simulera ce déplacement en changeant les valeurs des variables `NAV_X`, `NAV_Y`, `NAV_Z` à l'aide de `uPokeDB`. On renseignera la position (b_x, b_y, b_z) de la balise émettrice en précisant de nouveaux paramètres dans le bloc de configuration de `uSimModem` dans le fichier `.moos` de ce TP.



Le paramètre `AppTick` permettra au robot de recevoir une nouvelle variable `RANGE` toutes les 2 secondes.

2 Intégration d'un observateur d'état par intervalles dans une communauté MOOS

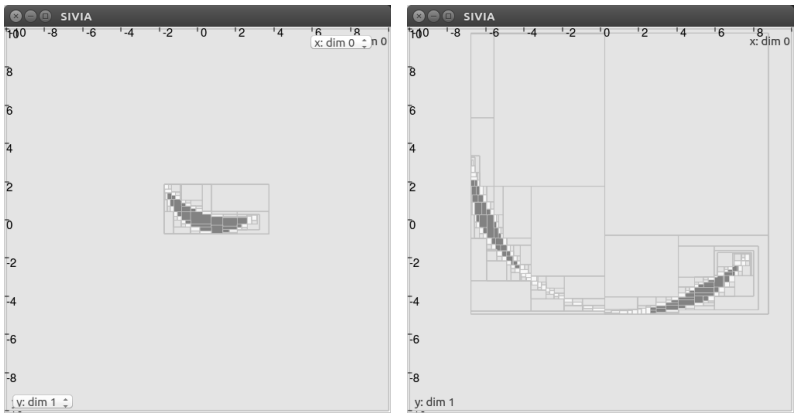
On se place dans un contexte 2D où le robot est initialement situé en $(0, 0)$. En pratique, cette position est inconnue. On se propose d'intégrer une localisation garantie du robot à partir de mesures range-only provenant de deux balises acoustiques dont les positions sont :

- balise A en $(1, 2)$;
- balise B en $(2, 5)$.

Ces données sont perçues par deux modems embarqués, avec une erreur e supposée bornée : $e \in [-0.5, 0.5]$.

1. Proposer une architecture séparant les différentes briques de simulation
   (la MOOSApp `uSimModem` ainsi que les MOOSVar `NAV_X`, `NAV_Y`, `RANGE` pouvant être réutilisées) ;
2. Écrire le nouveau fichier `.moos` correspondant dans le répertoire `missions/` ;
3. Implémenter l'application MOOS de localisation, que l'on appellera `pLocalization`.
   Dans un contexte garanti, un algorithme SIVIA permet de paver l'espace d'état pour définir l'ensemble des positions (x, y) possibles en fonction des deux observations bornées provenant des balises. L'application `pLocalization` pourrait retourner la boîte englobante contenant la position inconnue du robot, c'est à dire $(0, 0)$ dans cet exercice. Ici, on souhaite simplement dessiner le pavage résultant de l'algorithme SIVIA. Pour cela, on s'inspirera de l'exemple C++ `simple_sivia` de la bibliothèque graphique VIBes.
   - installer VIBes (voir notes d'installation ci-après) ;
   - générer un nouveau répertoire pour la MOOSApp de localisation dans `~/moos-ivp-extend/src/` ;
   - importer les fichiers `VIBES/client-api/C++/examples/interval` et `VIBES/client-api/C++/src/*` dans le répertoire de la nouvelle application ;
   - configurer le fichier `CMakeLists.txt` de `pLocalization` afin d'intégrer ces nouveaux fichiers dans la compilation de l'application ;
   - dans `Localization.cpp`, s'inspirer de l'exemple `VIBES/client-api/C++/examples/sivia_simple.cpp` proposé dans VIBes pour implémenter l'observateur d'état, basé sur un algorithme SIVIA.

     Note : les deux instances de `uSimModem` écrasent la même variable MOOS : `RANGE`. L'application `pLocalization` doit pouvoir connaître la balise émettrice de la donnée `RANGE` perçue, et intégrer simultanément les deux observations dans le même algorithme SIVIA.
4. Tester la mission. Déplacer le robot à l'aide de `uPokeDB`. Résultats attendus :



(a) Position du robot en $(0, 0)$ (b) Position du robot en $(-6, 0)$

Figure 1: Estimateur d'état garanti par mesures de distances.

Notes pour les utilisateurs de Linux

Installation de MOOS-IvP et du répertoire projet

Pour installer MOOS-IvP, vous aurez besoin de SVN :

```
> sudo apt-get install subversion
```

En cas de proxy, editer le fichier `/etc/subversion/servers` :

```
[Global]
http-proxy-host=192.168.1.10
http-proxy-port=3128
```

Installation de `moos-ivp` :

```
> sudo apt-get install g++ subversion xterm cmake \
                       libfltk1.3-dev freeglut3-dev libpng-dev \
                       libjpeg-dev libxft-dev libxinerama-dev libtiff5-dev
> svn co https://oceanai.mit.edu/svn/moos-ivp-aro/releases/moos-ivp-17.7.2 ~/moos-ivp
> cd ~/moos-ivp
> ./build.sh
```

Configuration des chemins d'accès : les applications MOOS doivent être accessibles depuis le `PATH` du système. Pour cela, ajouter à la fin de `~/.bashrc` :

```
export PATH=$PATH:~/moos-ivp/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/moos-ivp/lib
export PATH=$PATH:~/moos-ivp-extend/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/moos-ivp-extend/lib
```

Préparation du répertoire de travail `moos-ivp-extend` :

```
> source ~/.bashrc
> svn co https://oceanai.mit.edu/svn/moos-ivp-extend/trunk ~/moos-ivp-extend
> cd ~/moos-ivp-extend
> ./build.sh
```

Création d'une nouvelle MOOSApp

Les nouvelles MOOSApp sont à créer dans `~/moos-ivp-extend/src/` à l'aide de `GenMOOSApp_AppCasting`
Une ligne doit alors être ajoutée dans `~/moos-ivp-extend/src/CMakeLists.txt`

Configuration d'une mission

Par convention, l'exécution des applications MOOS se fait dans le répertoire `mission`. Un fichier `test_modem.moos` est à créer dans `~/moos-ivp-extend/missions/test_modem/`. Son exécution avec `pAntler` exécutera les MOOSApp renseignées dans cette configuration.

Installation de VIBes

VIBes est une bibliothèque graphique qui se veut simple et légère. Elle est disponible sur Github :

```
> sudo apt-get install qt5-default libqt5svg5-dev cmake git
> cd
> git clone https://github.com/ENSTABretagneRobotics/VIBES
> cd VIBES/viewer
> mkdir build ; cd build ; cmake .. ; make
> ./VIBes-viewer        # lancement de VIBes
```