Présentation du Middleware MOOS-IvP

Simon Rohou

ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, France
simon.rohou@ensta-bretagne.org

SHARC – 1er juillet 2016
Software and Hardware Architectures for Robots Control

Lab-STICC

ENSTA Bretagne

ENSTA Bretagne

Section 1

Introduction

Introduction

Middleware : définition

De l'architecture logicielle en robotique :

Un **Middleware** (intergiciel) est un logiciel tiers qui créé un réseau d'échanges d'informations entre différentes applications informatiques.

Il permet de diviser la partie logicielle du robot en une suite de processus pouvant être démarrés simultanément ou séquentiellement au lancement d'une mission.

Introduction

Utilité d'un middleware

L'intérêt d'un middleware ? C'est pouvoir :

- **séparer** les applications
  et donc clarifier la répartition du travail au sein d'une équipe
- **paralléliser** facilement les processus
  et tirer profit des n cœurs du processeur utilisé
- **distribuer** les applications sur différentes machines
  car elles sont toutes liées à un serveur
- **rejouer** des missions
  car il est facile de logger tous les changements de variables
- **réutiliser** les applications déjà implémentées
  qui répondent à nos problématiques

Introduction

MOOS : présentation

MOOS en quelques mots...

- « *a Mission Oriented Operating Suite* »
- architecture *publish-subscribe*
- initialement développé par Paul Newman (Université d'Oxford)
- implémentation en C++
- désormais interfaçable avec Python

Introduction

MOOS-IvP : une extension de MOOS

Quelle différence entre **MOOS** et **MOOS-IvP** ?

- **IvP** est une extension de **MOOS**
- des chercheurs du MIT [1] sont actifs dans son développement
- abréviation de *Interval Programming* ($\neq$ Interval Analysis)
- introduit de nombreux outils de prise de décision fondamentaux en robotique autonome
- **IvP** fourni aussi de nouveaux outils de contrôle (*ex :* PID), de simulation (*ex :* modélisation de courants marins), de communications (*ex :* interactions inter-robots), etc.

1. Michael R. Benjamin, Henrik Schmidt, John J. Leonard

Introduction

MOOS-IvP : les pour et les contre

- utilisation facile
- documentation très complète
- bon support technique
- économe en ressources $\Longrightarrow$ adapté à de l'embarqué
- utilisé par de grands organismes
  (MIT, Oxford, Bluefin Robotics, US Navy, CMRE, CGG, RTSys)
- contributions possibles sur Github
  (MOOS déjà présent, MOOS-IvP en migration)

- communauté souvent restreinte à la recherche
- absence de forums de discussions
- absence d'outils de tests unitaires ou d'intégration

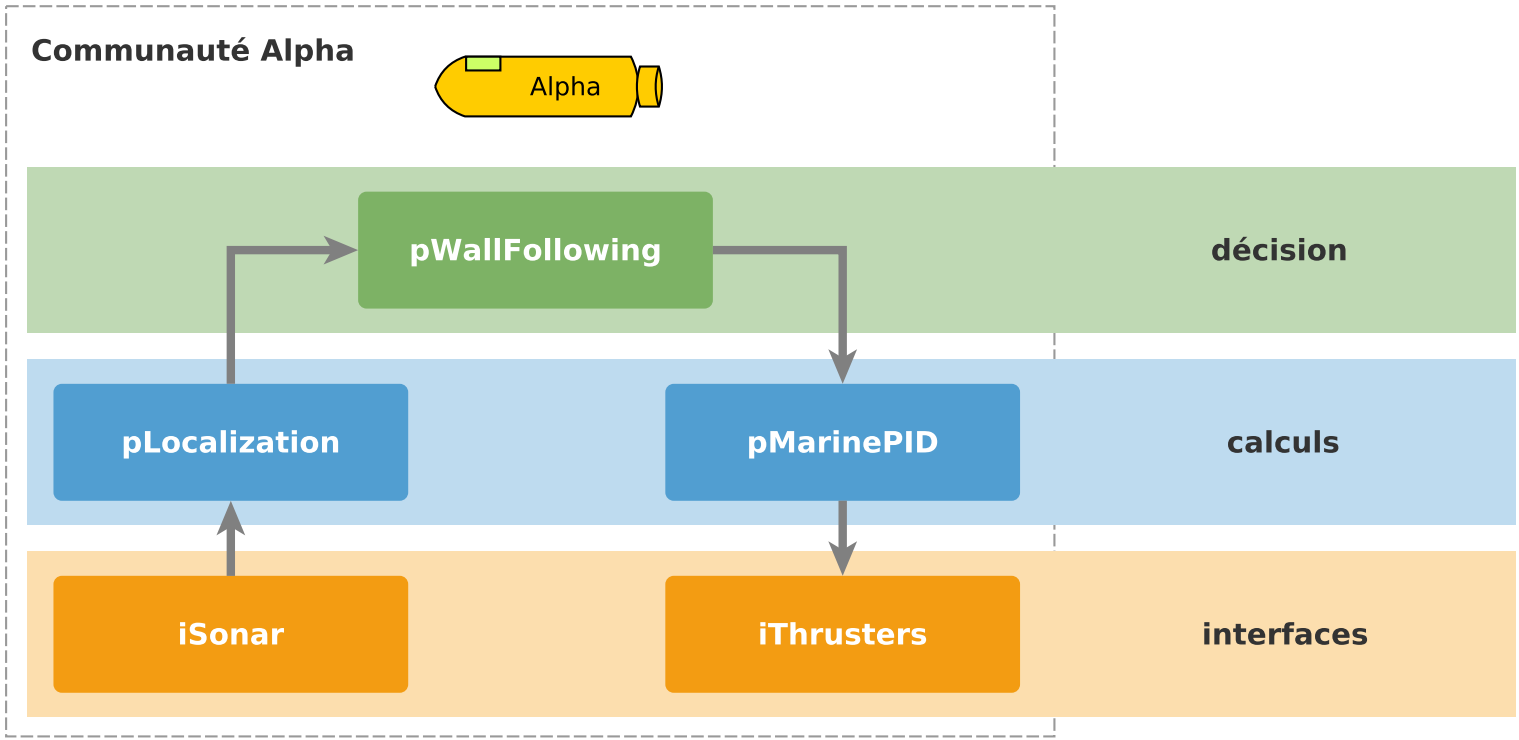
Section 2

Architecture publish-subscribe

Architecture publish-subscribe

Notion de communauté

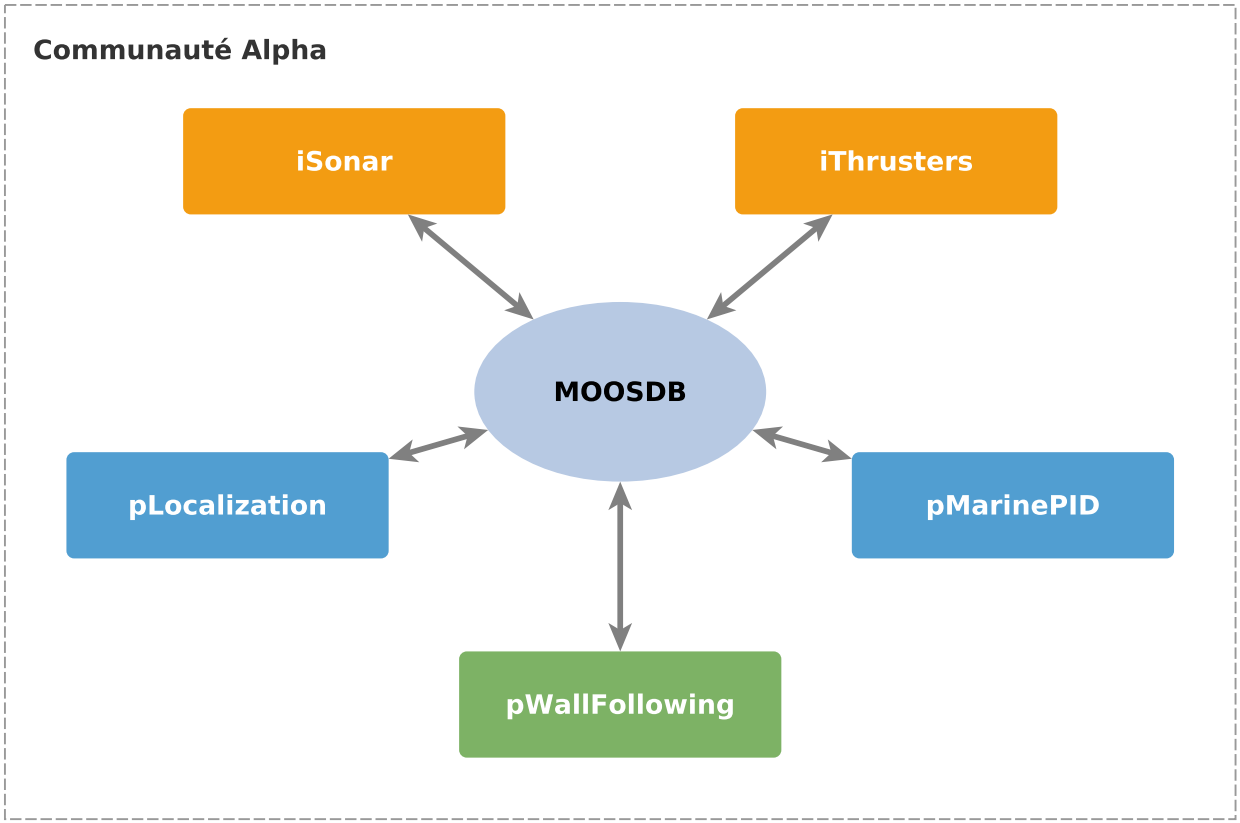
En général on associe un robot à une communauté de programmes. L'exemple ci-dessous rassemble 5 applications permettant à un robot sous-marin de faire un suivi de mur.



Architecture publish-subscribe

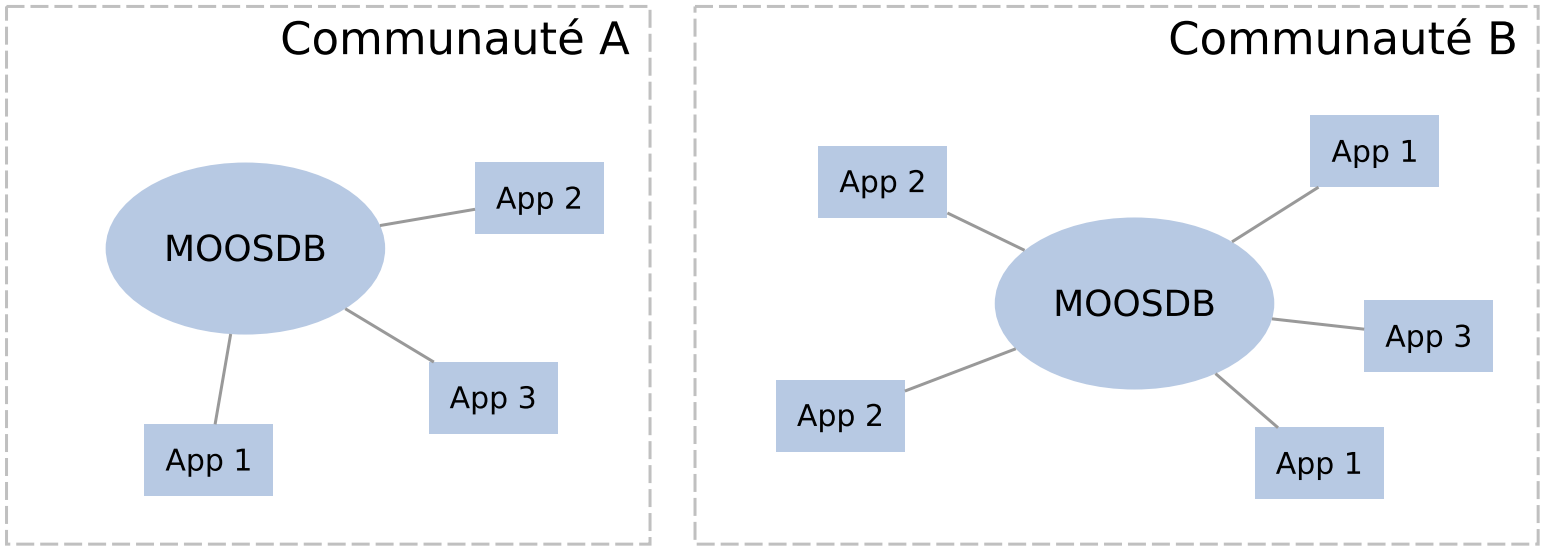
Notion de communauté

Dans MOOS, les programmes communiquent à travers une base de données centrale appelée MOOSDB :



Architecture publish-subscribe

Notion de communauté



Plusieurs communautés peuvent cohabiter sur une même machine.

Un programme peut avoir plusieurs instances dans une communauté.

Section 3

Quelques outils

Quelques outils

La toolbox de MOOS-IvP

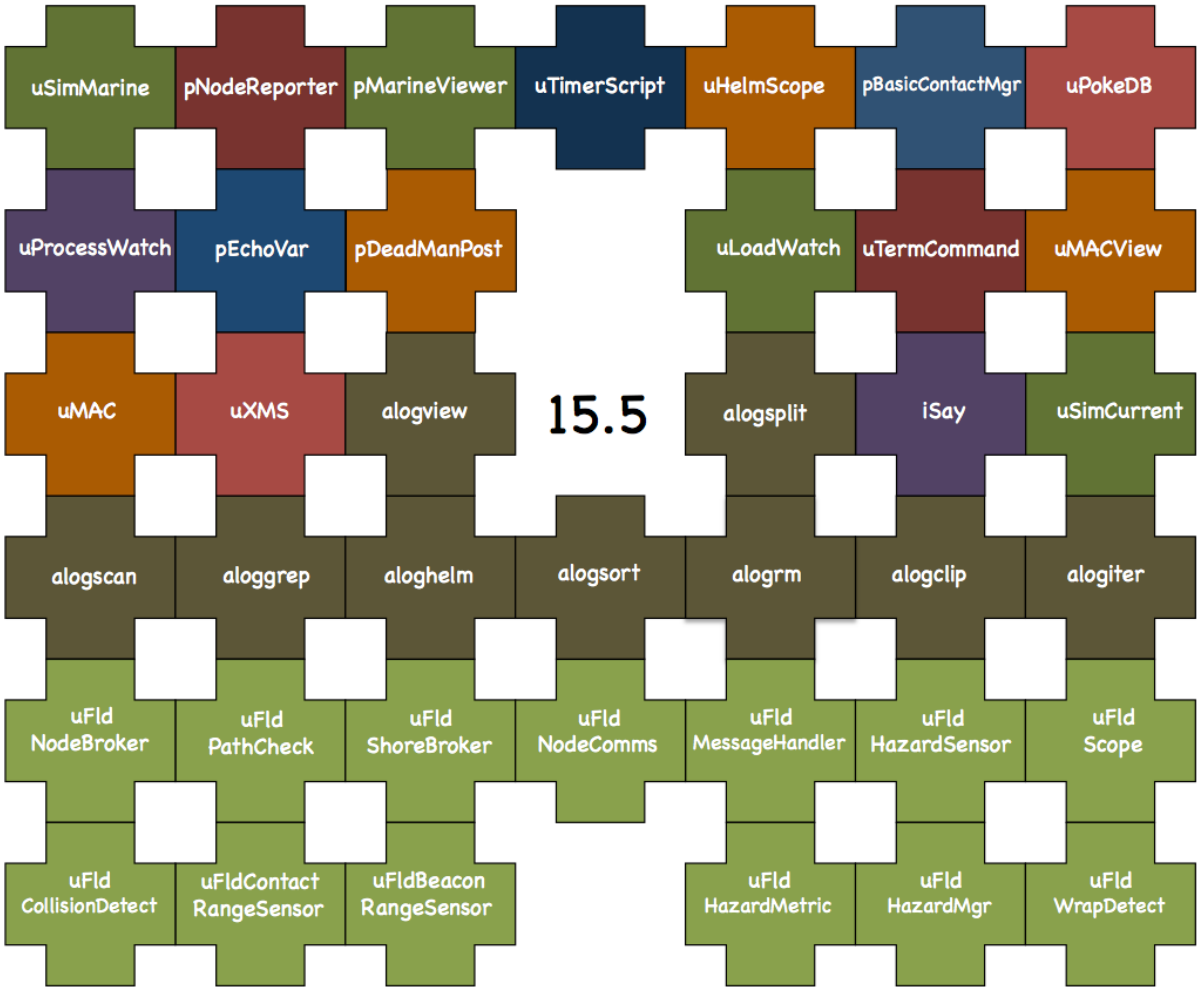


Figure – L'ensemble des `MOOSApp` proposées par IvP 15.5

Quelques outils

pMarineViewer : contrôle et visualisation

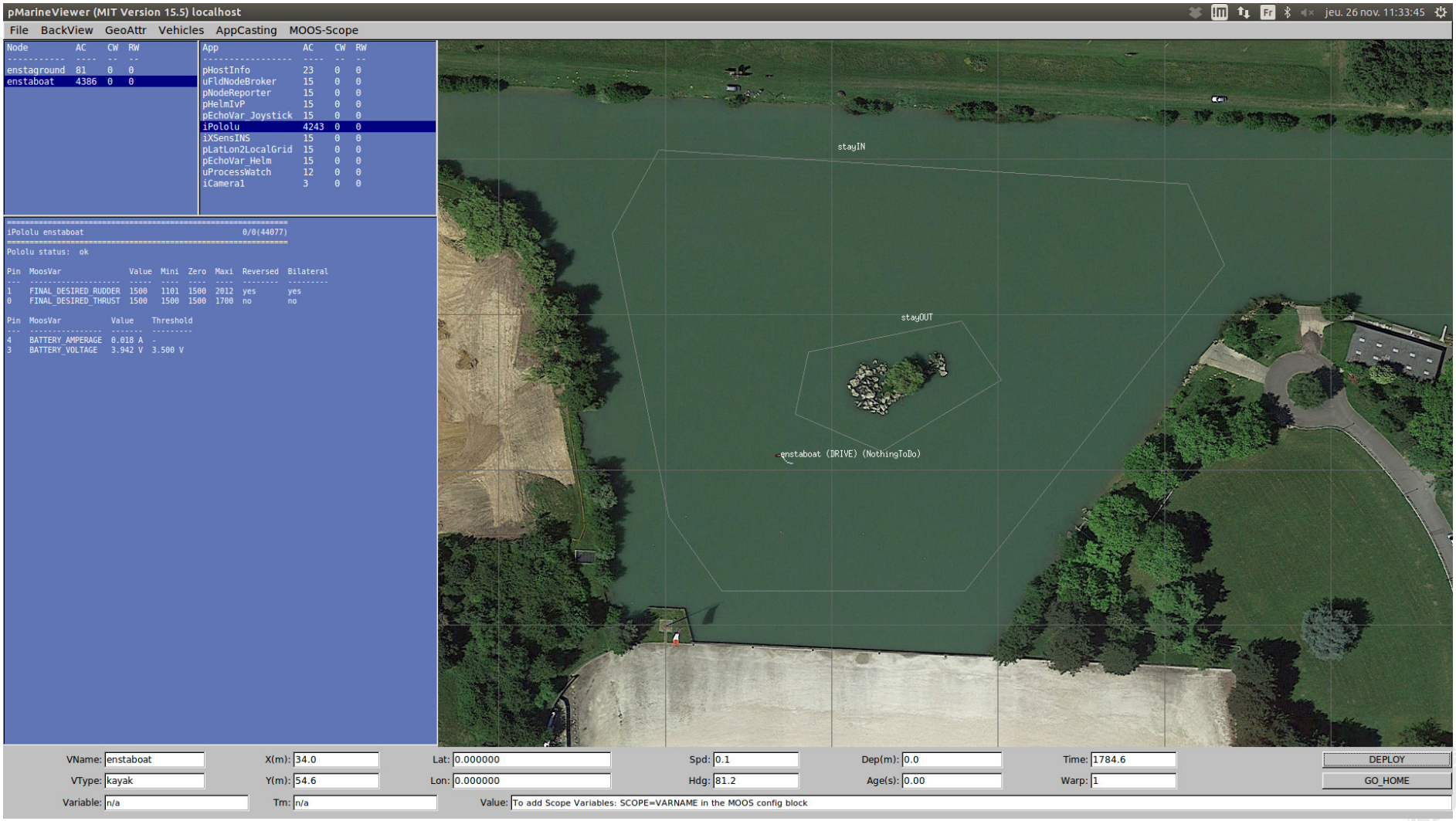


Figure – un bateau autonome sur le plan d'eau de Polytechnique

Quelques outils

pMarineViewer : contrôle et visualisation

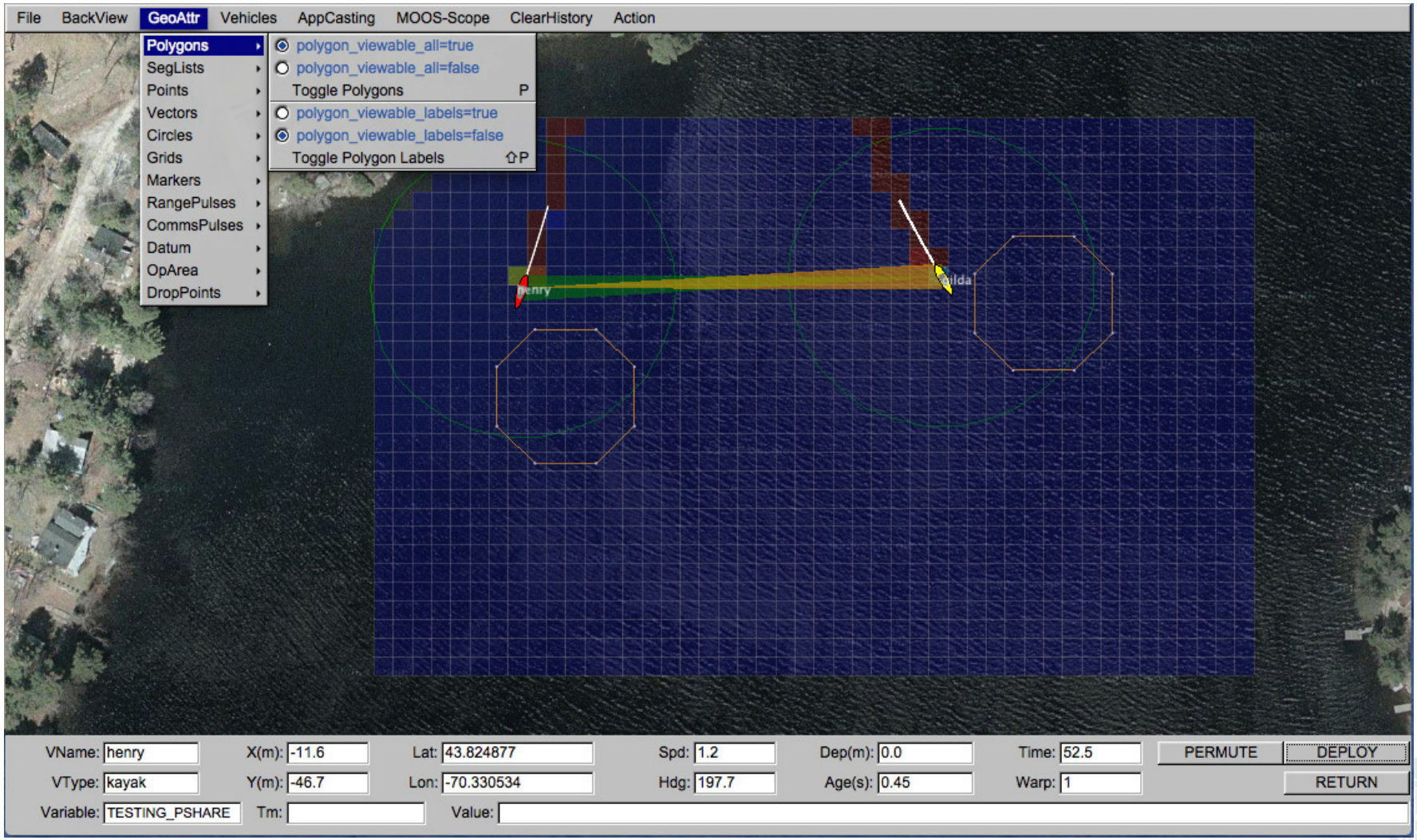
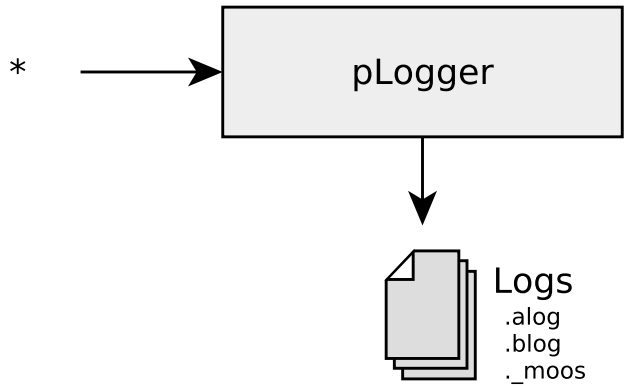


Figure – différents objets affichables dans `pMarineViewer`

Quelques outils

pLogger : enregistrer les données d'une mission

Les changements des `MOOSVar` sont enregistrés dans des fichiers :



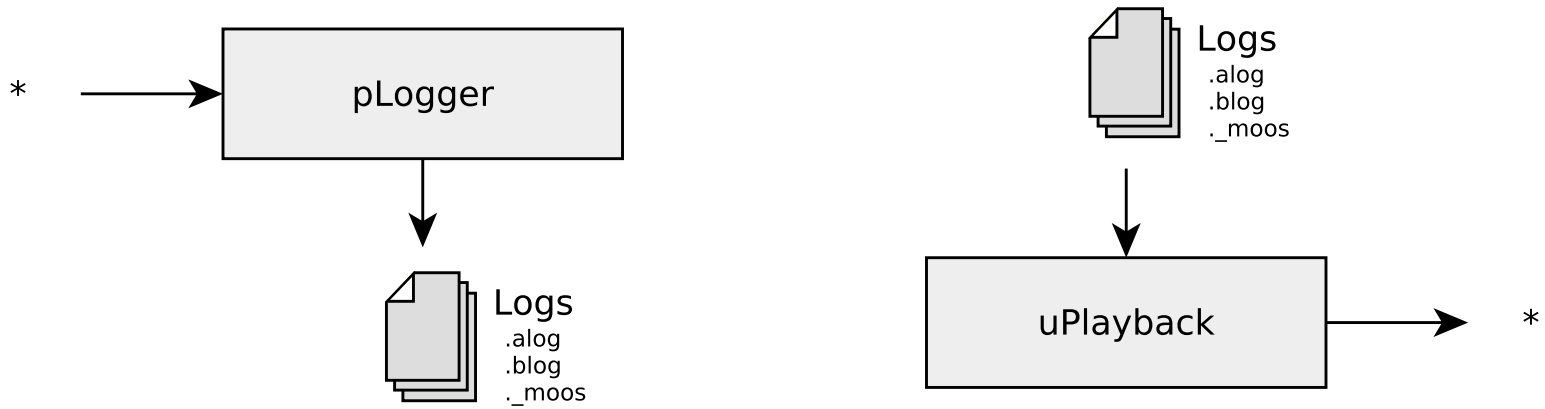
Des fichiers de logs sont créés pour chaque mission :

- `file.alog` : valeurs datées de chaque `MOOSVar`
- `file.blog` : logs de `MOOSVar` binaires
- `file._moos` : sauvegarde du fichier de mission `.moos` utilisé

Quelques outils

uPlayback : rejouer une mission

`uPlayback` utilise les logs de `pLogger`.
Les changements des `MOOSVar` sont reproduits dans la MOOSDB :



`uPlayback` relance la mission telle que configurée dans le `._moos`

Plus d'information :

`http://www.robots.ox.ac.uk/~pnewman/MOOSDocumentation/Essentials/Logging/latex/pLogger.pdf`

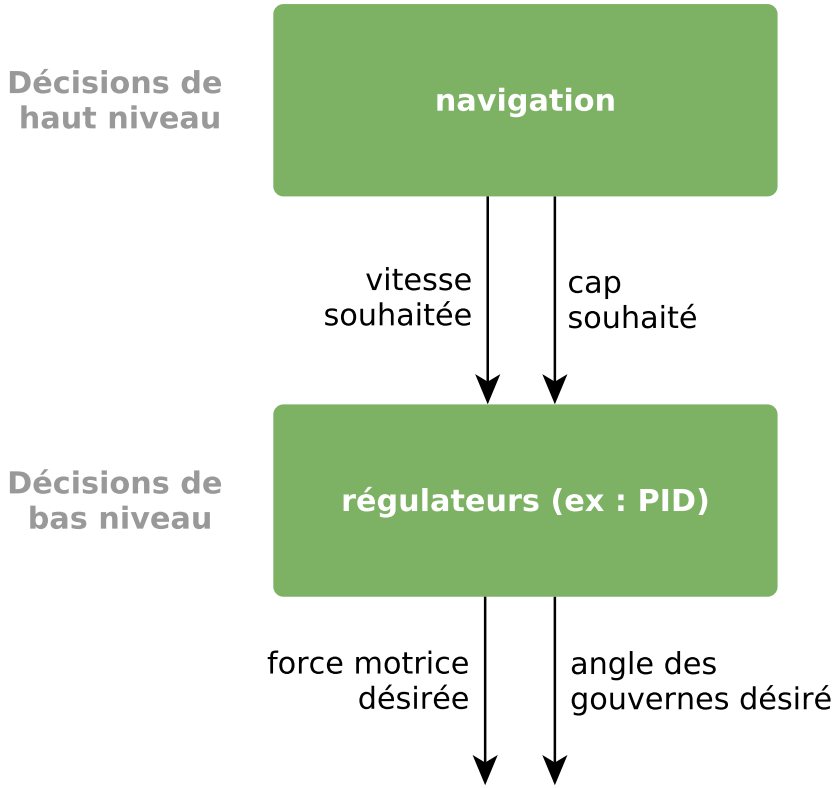
Section 4

Prises de décisions avec IvP

Prises de décisions avec IvP

De l'autonomie à différents niveaux

Plusieurs niveaux d'abstraction :

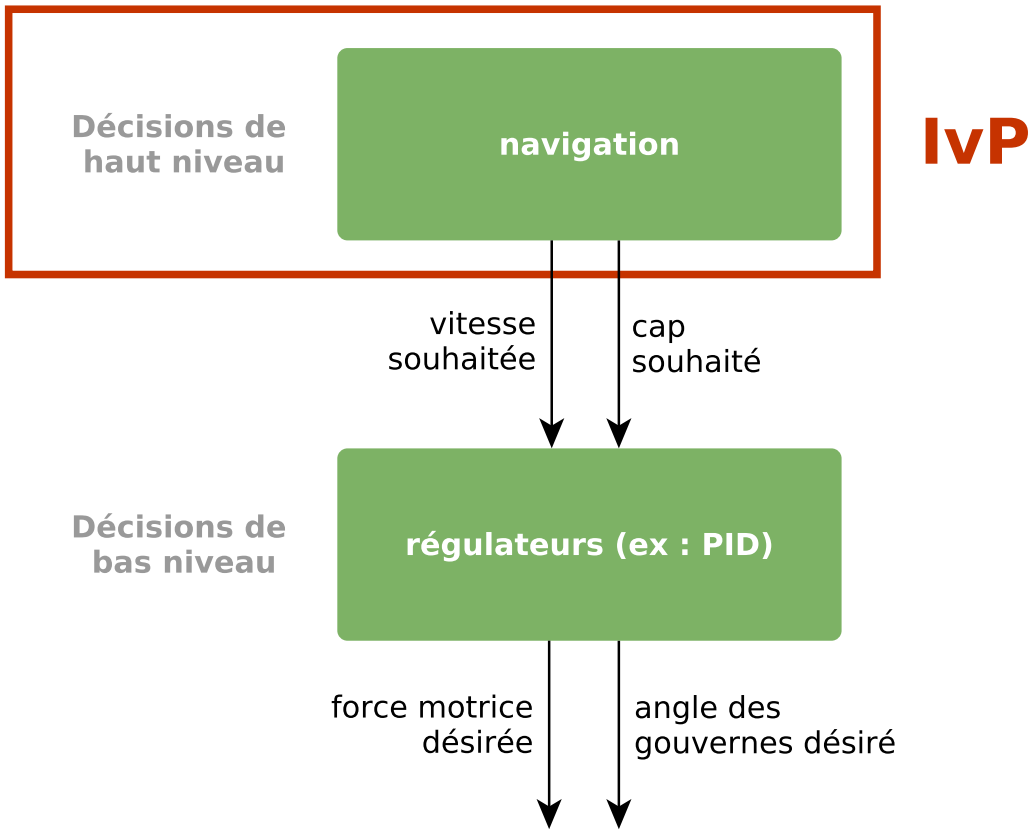


On s'intéresse à présent aux décisions de haut niveau.

Prises de décisions avec IvP

De l'autonomie à différents niveaux

Plusieurs niveaux d'abstraction :



On s'intéresse à présent aux décisions de haut niveau.

Prises de décisions avec IvP

De la modularité dans l'autonomie

Le besoin, **gérer des missions autonomes** :

- pouvant être complexes et difficiles à organiser
- évoluant en fonction de l'environnement, ou d'autres robots
- ayant déjà été implémentées par d'autres organismes

Un middleware répond à ce besoin.

Pour IvP, on parle de « *behavior based architecture* ».

Prises de décisions avec IvP

Qu'est-ce qu'une behavior ?

Une *behavior* est une **bibliothèque** se chargeant de proposer une **décision** selon un **contexte** donné.

Figure – Une behavior quelconque

Exemple de décisions :

- tourner à droite
- ne pas revenir en arrière
- maintenir une vitesse

Prises de décisions avec IvP

Behavior : domaines de décision

Chaque décision est proposée sur un domaine donné.

Figure – Une behavior proposant une décision sur ses 3 domaines

Exemple de domaines :

- vitesse : de $-2m/s$ à $3m/s$
- cap : de $0°$ à $360°$
- profondeur : de $0m$ à $50m$

Généralement, un domaine correspond à une composante du vecteur d'état $\mathbf{x}$ du robot.

Prises de décisions avec IvP

Behavior : fonctions IvP

La force d'IvP est que la décision proposée n'est pas un scalaire mais une **fonction IvP**. Elle permet une plus grande modularité dans le processus de prise de décision.

On associe une valeur à chaque portion d'un domaine de décision.

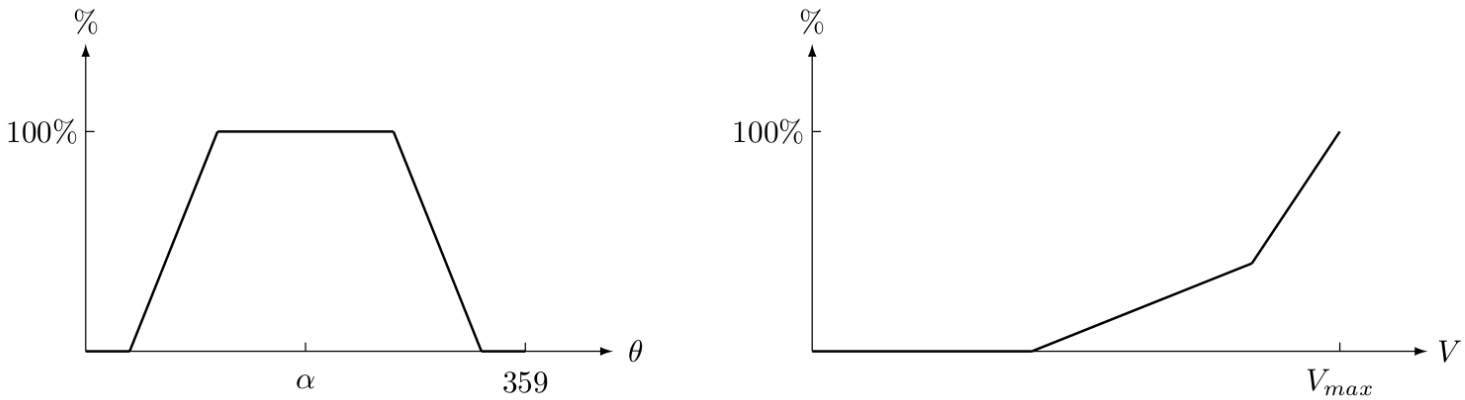
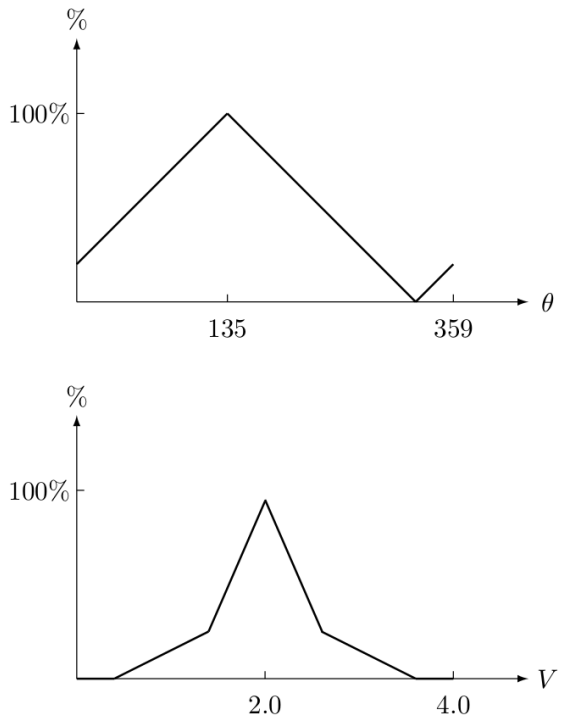


Figure – Des fonctions de prise de décision — ici, la décision proposée est d'aller *globalement* dans la direction de α en favorisant *franchement* la vitesse V_{max}.

Prises de décisions avec IvP

Behavior : produit cartésien parmi les variables de décision



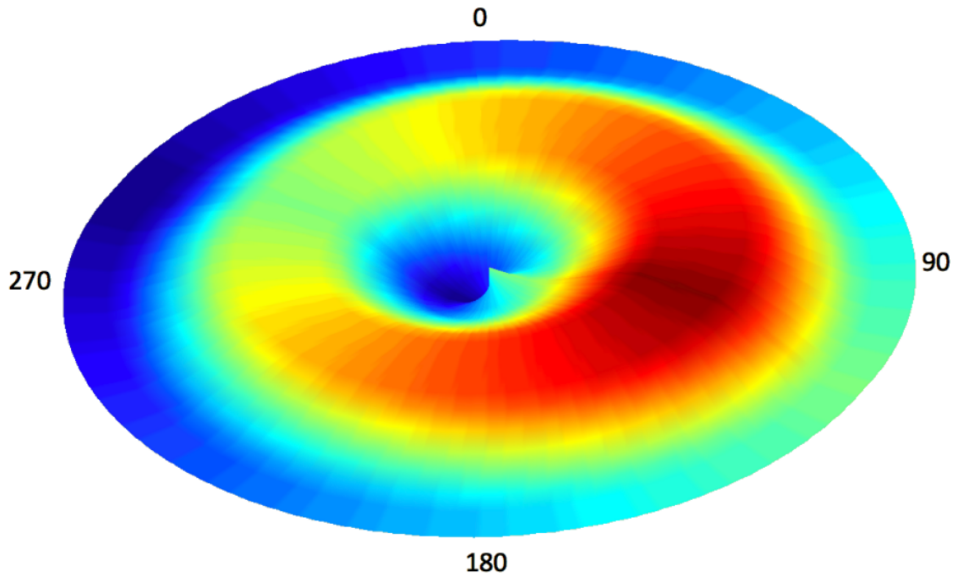


Figure – Représentation de fonctions IvP selon 2 domaines : le cap du robot de 0 à 360° et sa vitesse de 0 à 4m/s. Ici, le point le plus chaud donne la décision finale : $\theta_D = 135°$ et $V_D = 2.0$m/s.

Prises de décisions avec IvP

Fusion de behaviors : le solveur IvP

Plusieurs lois de comportement peuvent être intégrées dans un robot. Selon le contexte, le robot doit pouvoir prendre une décision en favorisant l'une de ces lois. Son comportement est défini comme la **synthèse des propositions** des behaviors en jeu.

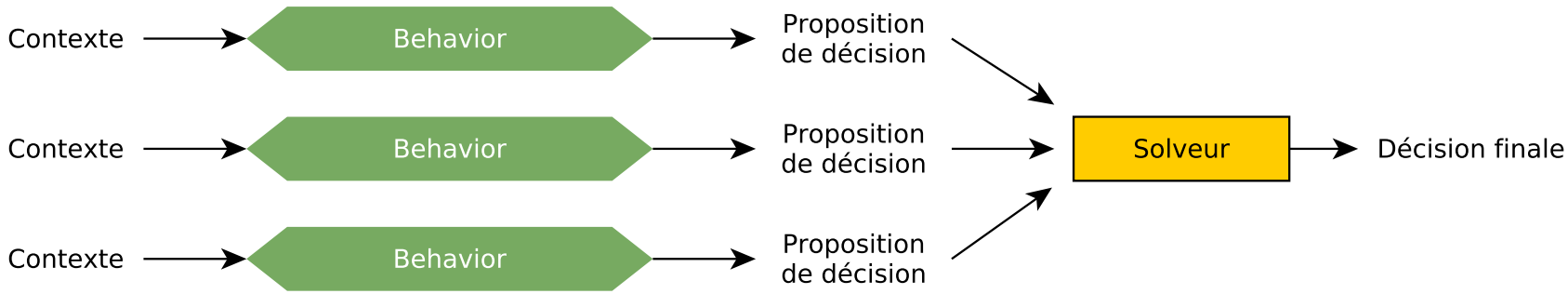


Figure – Plusieurs behaviors concurrentes – le solveur se charge de synthétiser un comportement en s'appuyant sur les fonctions IvP

Le solveur, c'est l'application `pHelmIvP`.

Prises de décisions avec IvP

Fusion de behaviors : le solveur IvP

Comment le solveur fait cette synthèse ?

- chaque fonction IvP f_i est associée à un **poids** w_i
- les fonctions pondérées sont ensuite **fusionnées** :

$$\vec{x}^{\,*} = \underset{\vec{x}}{\operatorname{argmax}} \sum_{i=0}^{k-1} w_i f_i(\vec{x}) \tag{1}$$

Le résultat est un simple point dans l'espace de décision.
Sa valeur est publiée par le solveur dans la MOOSDB.

Prises de décisions avec IvP

Quel est l'intérêt des fonctions IvP ?

La conception d'une behavior doit se faire en pensant à son intégration dans le solveur.

Le profil de la fonction IvP doit déterminer le degré de **tolérance** de la behavior : une fonction présentant une forte dérivée s'imposera davantage qu'une fonction plate.

Prises de décisions avec IvP

Différents degrés de tolérance pour une behavior

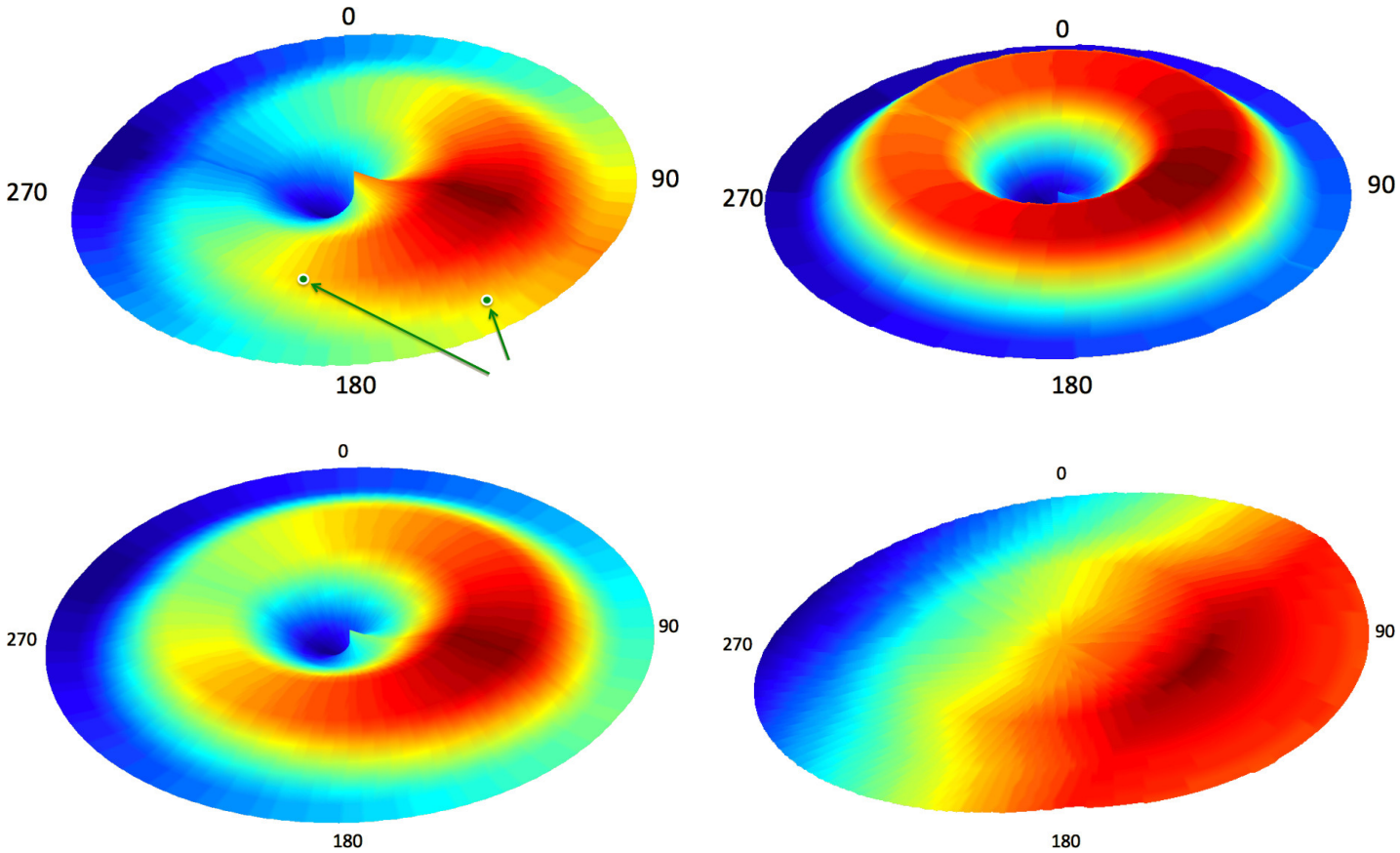


Figure – Différents profils de décision pour un même contexte – dans chaque cas, la décision est la même (au point chaud) – mais la décision sera différente en présence d'autres behaviors

Prises de décisions avec IvP

Un exemple de behavior dominante

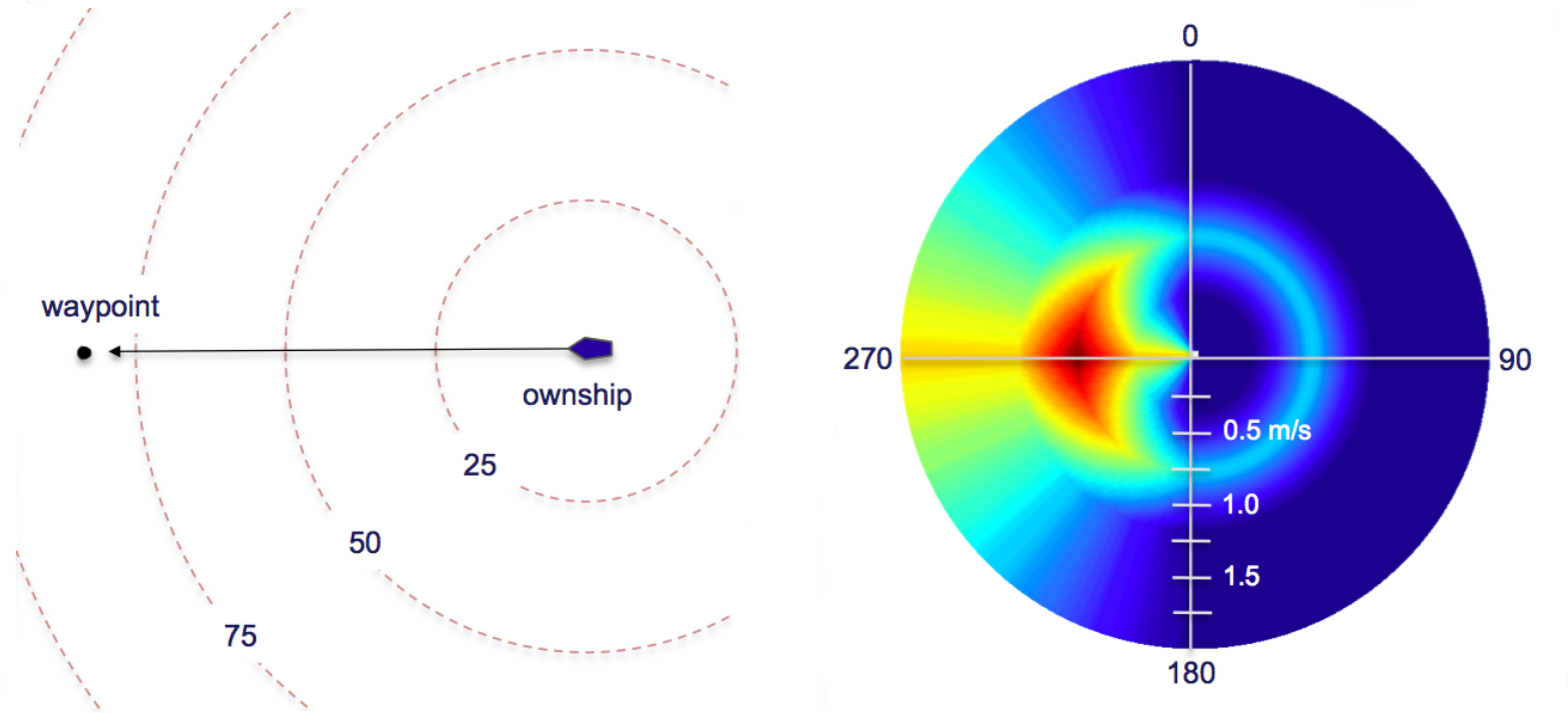


Figure – Suivi de cibles – à gauche, le contexte ; à droite, la projection des fonctions IvP – le robot est clairement amené à se diriger à l'Ouest pour atteindre son waypoint.

Prises de décisions avec IvP

Un exemple de behavior tolérante

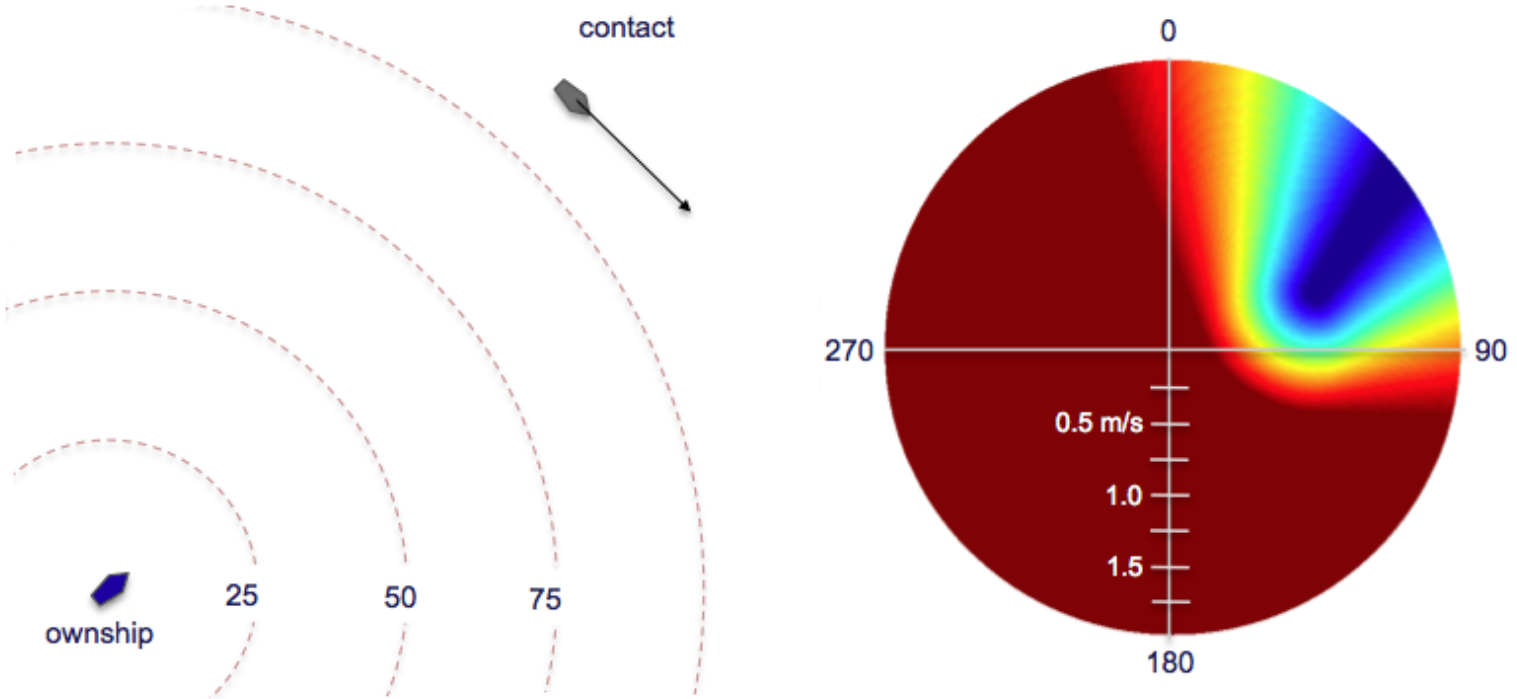
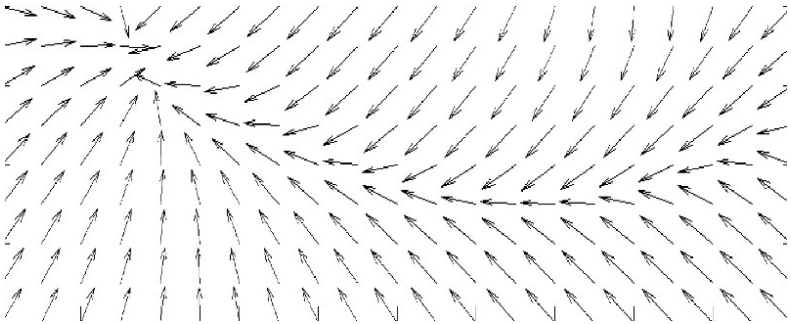


Figure – Évitement d'obstacles – à gauche, le contexte ; à droite, la projection des fonctions IvP – le robot est libre dans ses déplacements tant qu'il ne se dirige pas sur l'obstacle.

Prises de décisions avec IvP

Quelle différence avec une commande par chp. de vecteur ?



La **modularité**.

On retrouve ici les avantages d'un middleware :

- **séparer** les tâches
  et donc clarifier la répartition du travail au sein d'une équipe
- **paralléliser** facilement les processus
  et tirer profit des n cœurs du processeur utilisé
- **réutiliser** les behaviors déjà implémentées
  qui répondent à nos problématiques

Prises de décisions avec IvP

Quelle différence avec une commande par chp. de vecteur ?

Ici, le concepteur respecte le formalisme d'IvP.

L'intégration de plusieurs behaviors se fait donc simplement.

Concrètement, une behavior est une **bibliothèque** développée en C++. Son développement et sa compilation peut se faire indépendamment de tout autre projet l'utilisant.

L'utilisateur n'a pas besoin de connaître les fonctions IvP d'une behavior pour s'en servir.

Prises de décisions avec IvP

La behavior BHV_StationKeep

Comportement : atteindre une position et s'y maintenir.

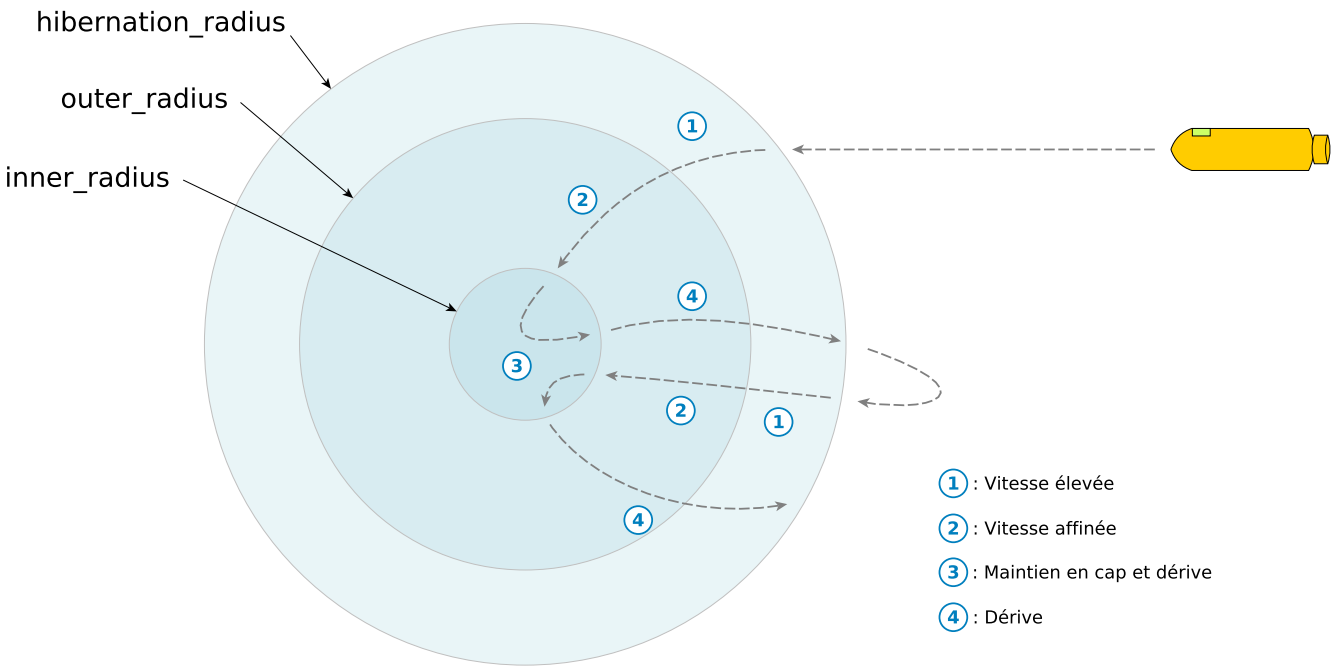


Figure – *Station Keeping* – la position à atteindre est définie par `inner_radius` et une dérive est tolérée tant que le robot reste dans la zone déterminée par `hibernation_radius`. Le paramètre `outer_radius` permet de préciser une vitesse intermédiaire.

Prises de décisions avec IvP

La behavior BHV_Waypoint

Comportement : suivre une série de points prédéfinis.

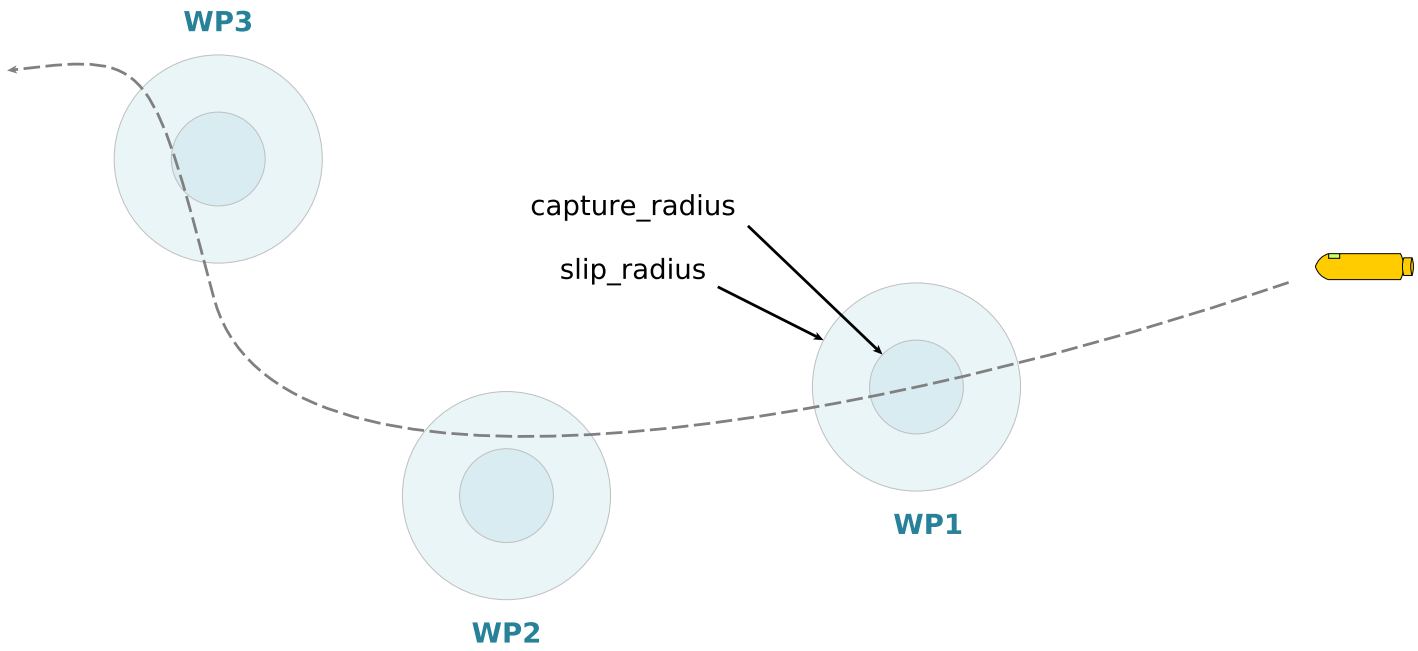


Figure – *Waypoint* – la position à atteindre est définie par `capture_radius` et le waypoint est validé si le robot s'éloigne de ce point après avoir traversé `slip_radius`

Prises de décisions avec IvP

La behavior BHV_Waypoint

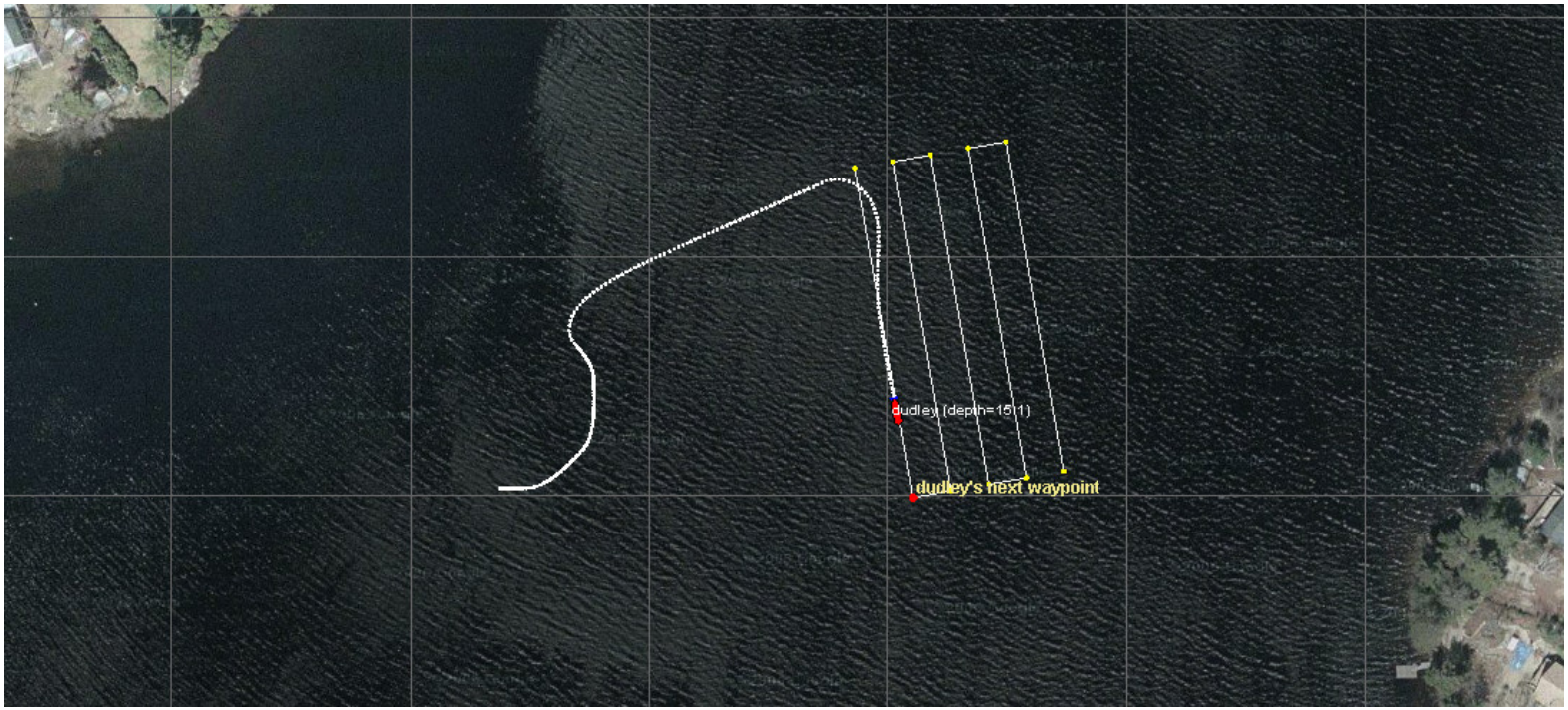


Figure – Exemple d'utilisation pour un suivi de ligne – la behavior affiche elle-même des informations dans l'interface de visualisation `pMarineViewer`

Prises de décisions avec IvP

La behavior BHV_AvoidCollision

Comportement : éviter un obstacle mobile.

Cette behavior est une behavior de contact : elle conditionne un comportement en fonction d'autres véhicules, mobiles.

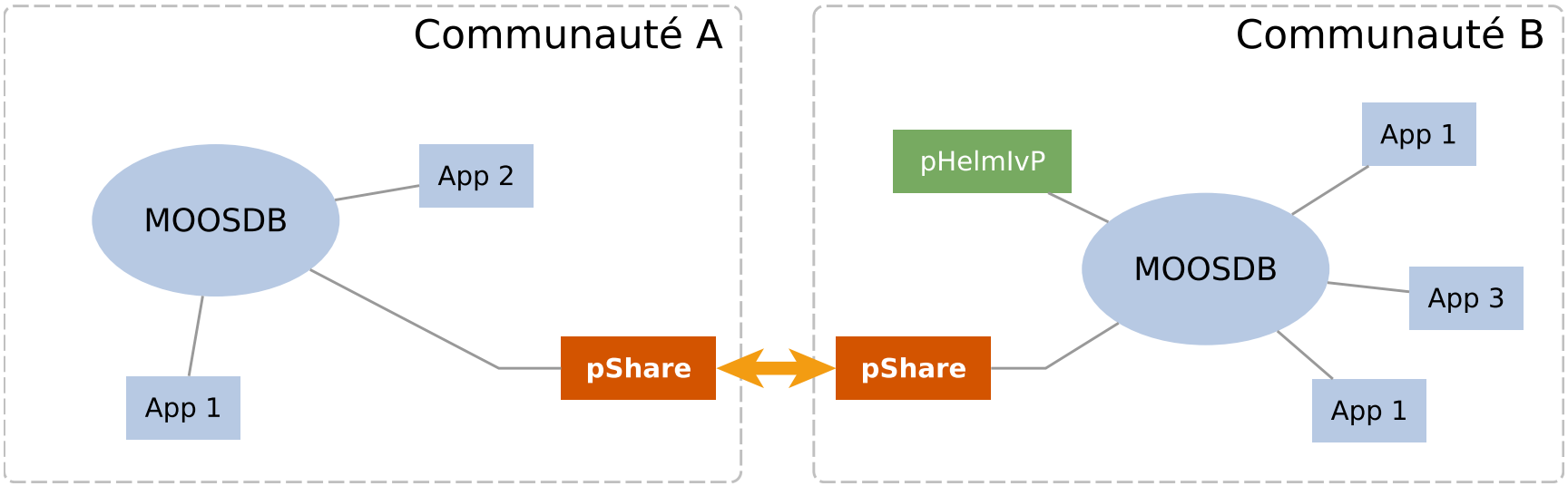


Figure – Évitement d'obstacle – le robot *B* va chercher à éviter le robot *A*. La position du robot *A* est communiquée au robot *B* par la MOOSApp `pShare` ; cette information est lue par la behavior `BHV_AvoidCollision`.

Prises de décisions avec IvP

La behavior BHV_AvoidCollision

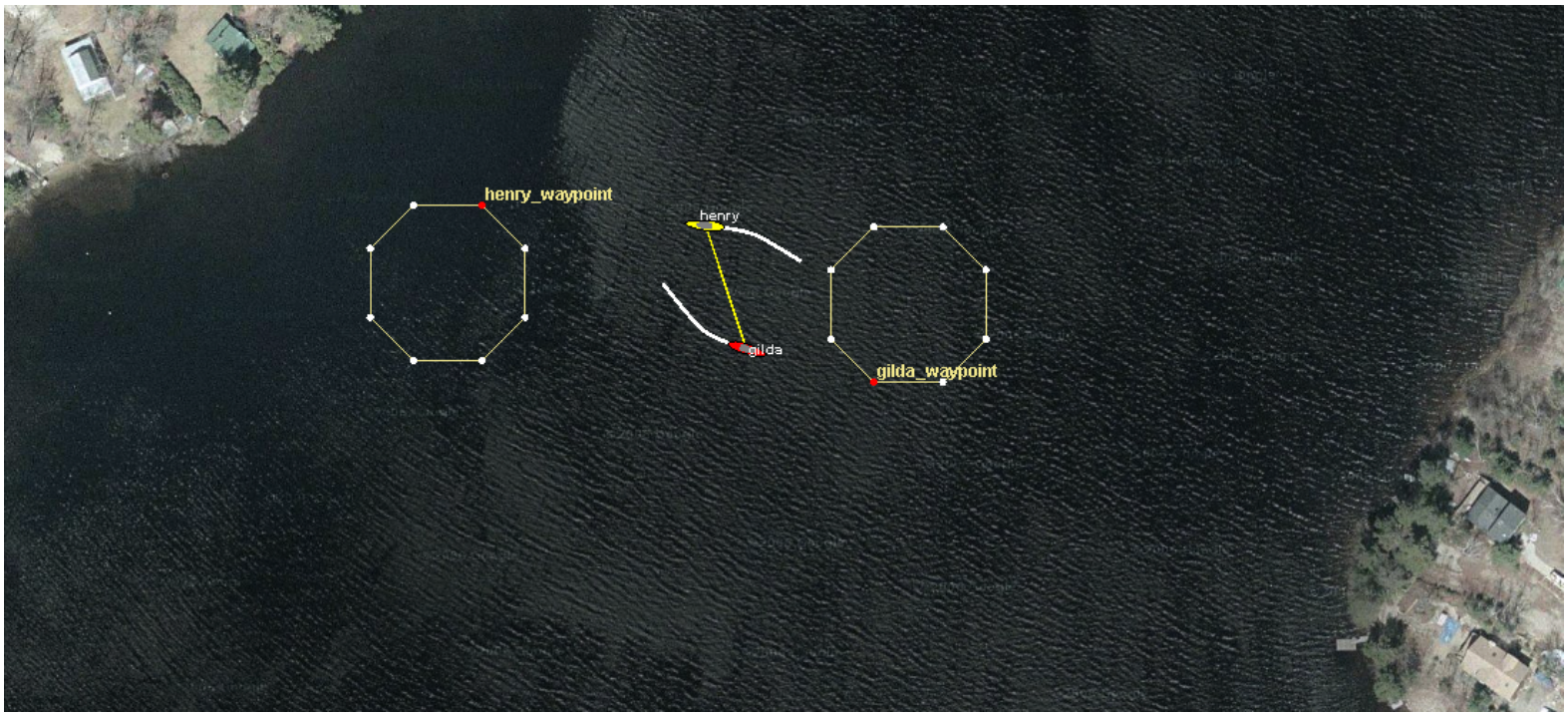


Figure – Exemple d'utilisation pour un évitement de robot – ici, les deux robots poursuivent leur mission (passer d'un polygone à l'autre) tout en évitant une collision. Une fusion de fonctions IvP a eu lieu et a permis un changement de comportement en douceur.

Section 5

Références

Références

Pour aller plus loin

La documentation MOOS-IvP couvre largement le sujet avec :

- d'autres behaviors prêtes à l'emploi
- de nombreux exemples d'utilisation dans les *labs*
- une documentation sur la conception d'une behavior de A à Z
- des outils de visualisation des fonctions IvP

`http://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php`

Références

Liens, publications

- Page officielle de la V10 de MOOS (Oxford)
- Site officiel de MOOS-IvP – documentation en ligne

Publications :

– M. Benjamin, H. Schmidt, P. Newman, J. Leonard,
Nested Autonomy for Unmanned Marine Vehicles with MOOS-IvP,
Journal of Field Robotics, Volume 27, Issue 6, pp. 834-875,
November 2010.

– M. Benjamin,
The Interval Programming Model for Multi-Objective Decision Making,
AI Memo, 2004-021, September 2004.

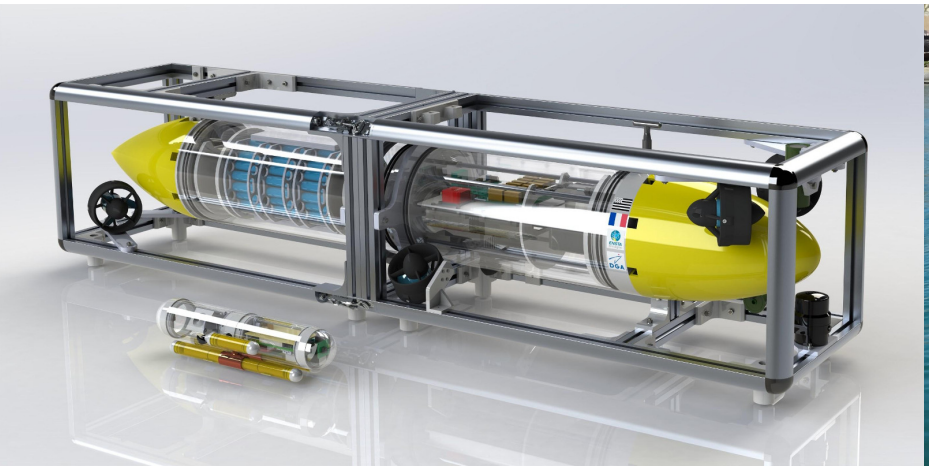
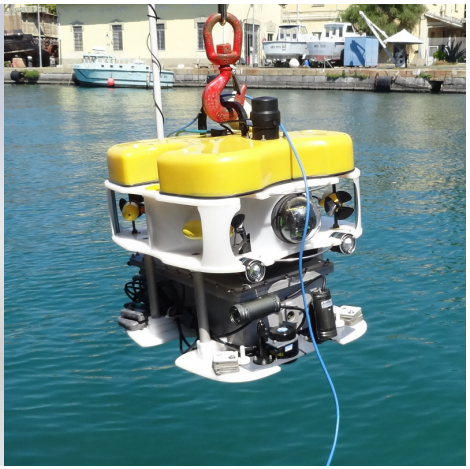
Crédits :

– les images de fonctions IvP proviennent des travaux de M. Benjamin

Références

MOOS-IvP à l'ENSTA Bretagne

MOOS-IvP est aussi utilisé sur les robots de l'ENSTA Bretagne :

Une `MOOSApp` a été développée pour chaque capteur de ces robots.

Le code est disponible sur Github :

`www.github.com/ENSTABretagneRobotics/moos-ivp-enstabretagne`

Références

Cours

Enseignement de MOOS-IvP à l'**ENSTA Bretagne** :
`http://simon-rohou.fr/cours/moos-ivp/`

Cours proposé en 3 parties :

- Partie 1/3 - Introduction au Middleware MOOS

  1h de cours + 3h de TP

- Partie 2/3 - Outils et visualisation

  1h de cours + 3h de TP

- Partie 3/3 - Autonomie et Prise de Décisions

  1h de cours + 3h de TP

Références

Cours

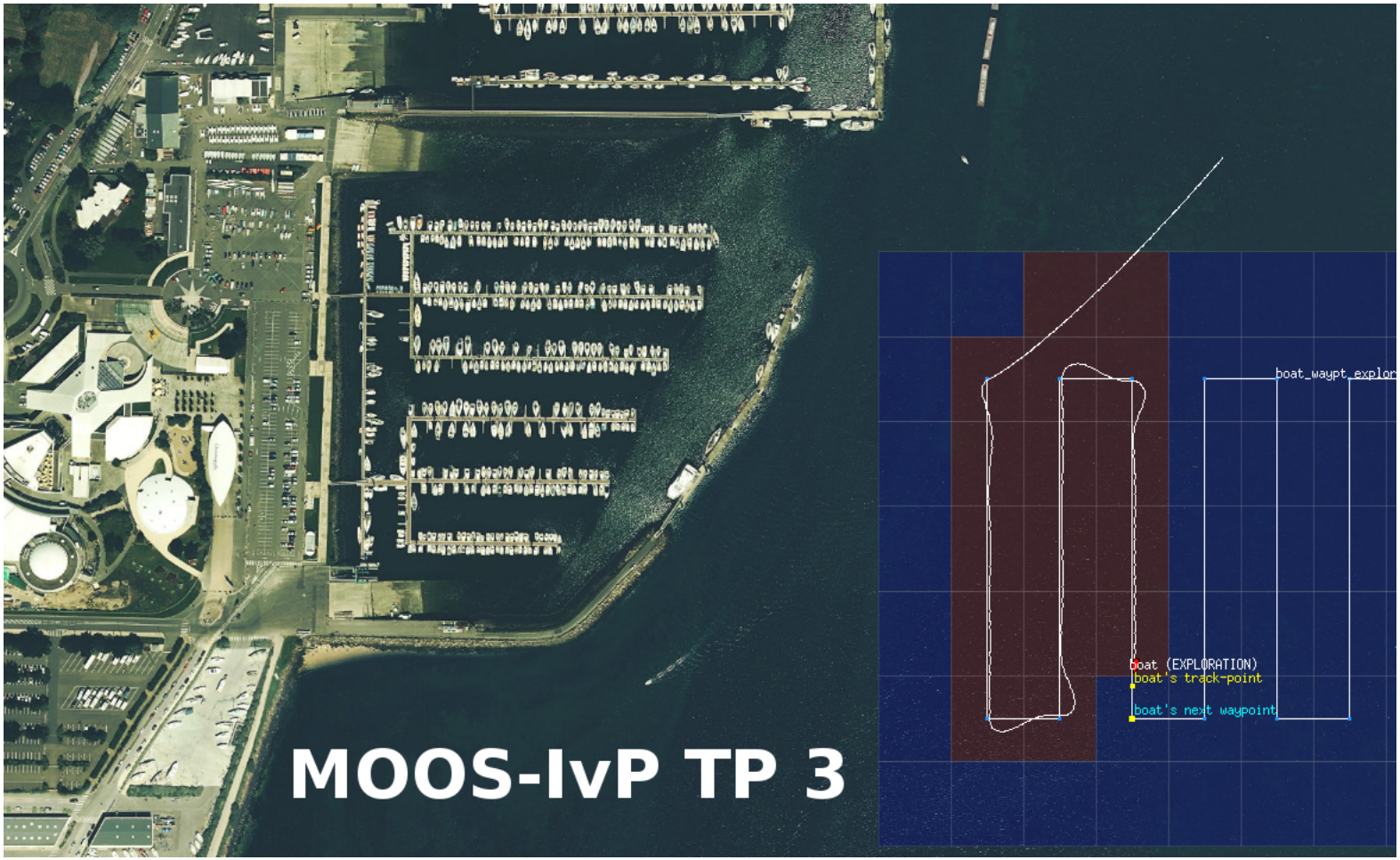


Figure – TP extrait d'un cours sur MOOS-IvP

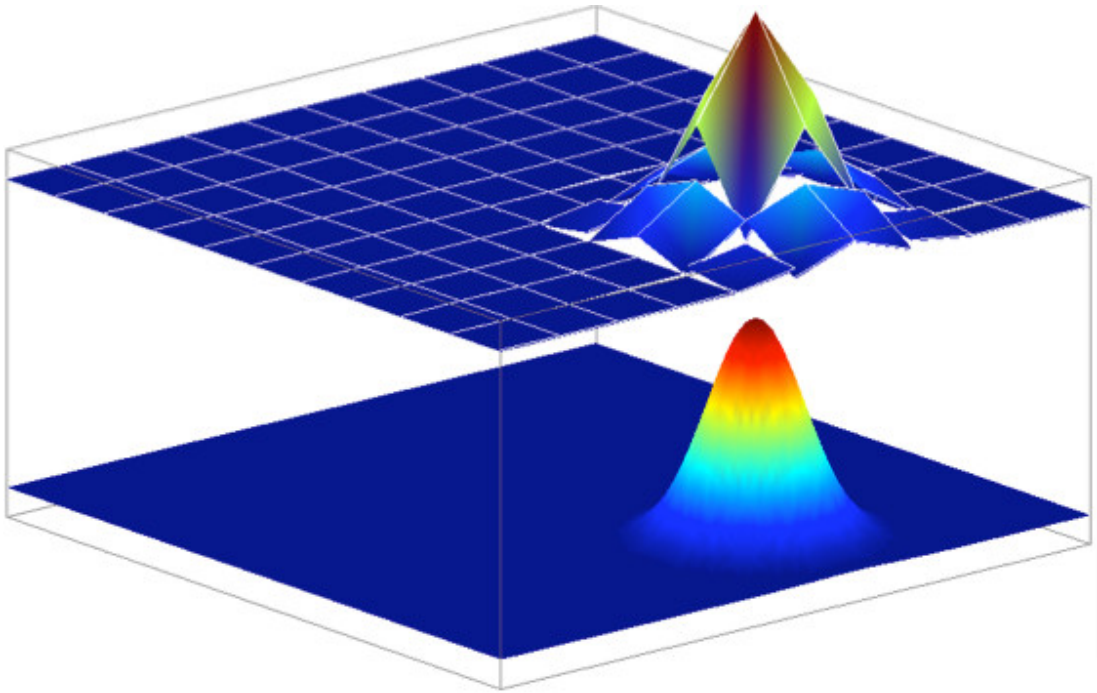
ENSTA Bretagne

Annexes

IvP : la notion d'Interval Programming

Interval Programming $\neq$ Interval Analysis

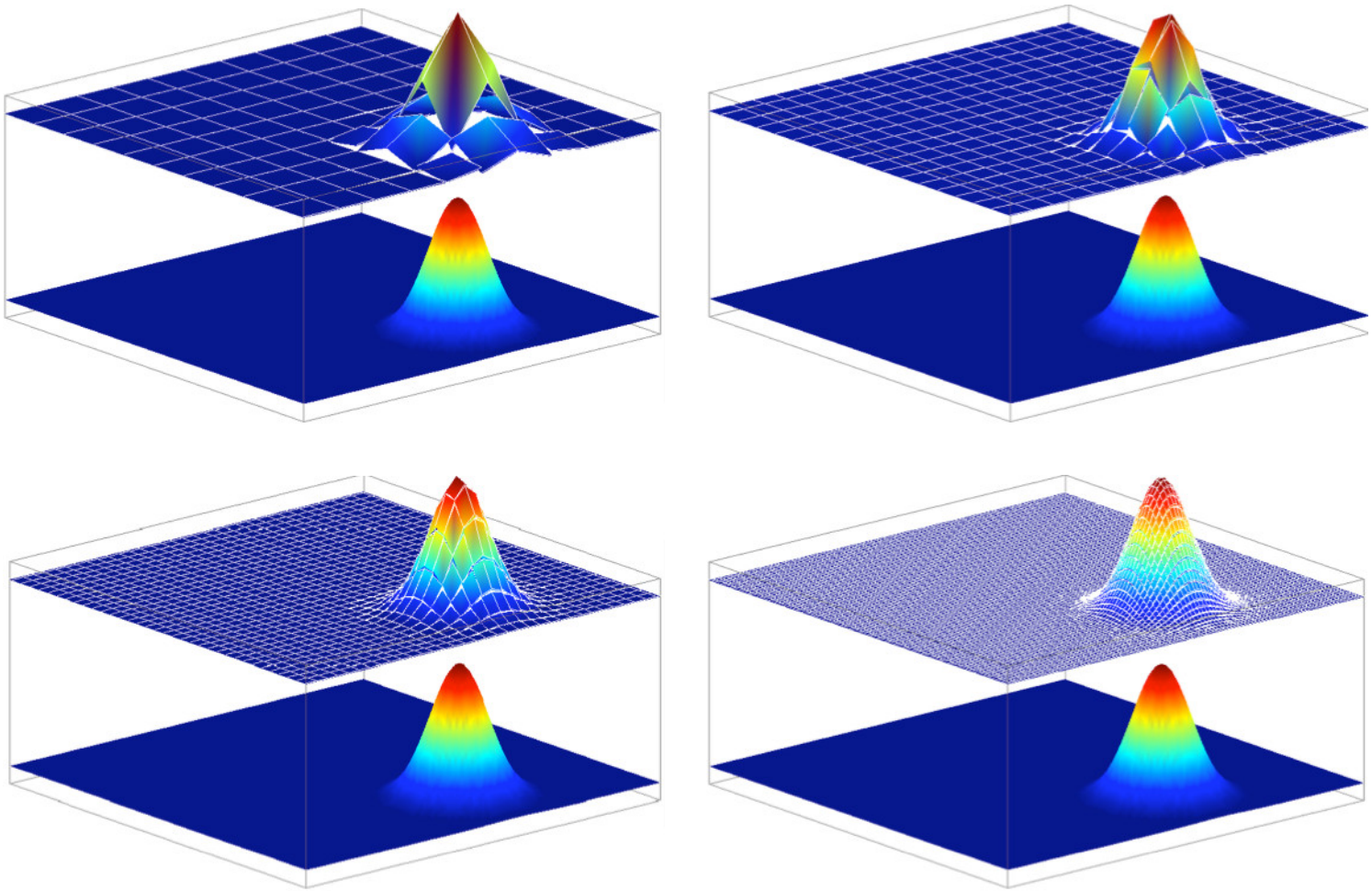
Une décision est prise sur un domaine en fonction des poids calculés en chaque point de ce domaine. En pratique, ce domaine est **discrétisé** : il est caractérisé par un ensemble d'**intervalles**.



Annexes

IvP : la notion d'Interval Programming

Plus la précision du domaine sera grande, plus la décision sera fine.
Le calcul en sera d'autant plus conséquent.



Annexes

IvP : la notion d'Interval Programming

IvP propose un découpage optimisé du domaine :