MOOS-IvP : Autonomie et Prise de Décisions

Simon Rohou

Février 2019

V1.01
Supports de cours disponibles sur
`www.simon-rohou.fr/cours/moos-ivp/`

ENSTA Bretagne

Section 1

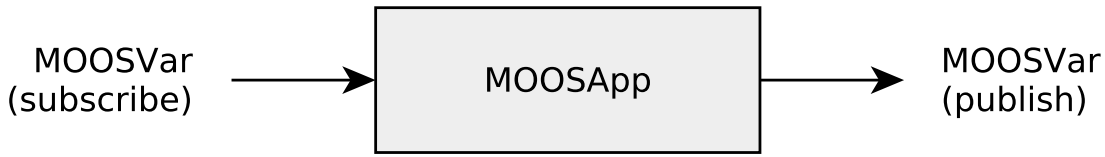
Rappel des cours 1 et 2

Rappel des cours 1 et 2

Une application MOOS : résumé

Interface d'une application MOOS :

- une `MOOSApp` réalise une tâche particulière
- elle a son propre processus
- elle communique avec d'autres `MOOSApp` *via* la MOOSDB à travers des variables `MOOSVar`
- cette communication se fait au sein d'une architecture *publish-subscribe* :
  - elle s'abonne à des `MOOSVar` (*subscribe*)
  - elle publie des `MOOSVar` (*publish*)

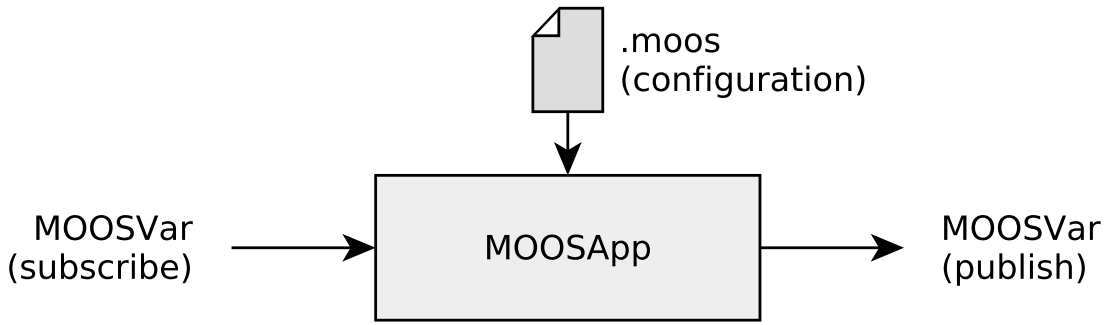


Rappel des cours 1 et 2

Une application MOOS : résumé

Paramétrage d'une application MOOS :

- une `MOOSApp` peut-être configurée *via* un fichier `.moos`
- les informations paramétrées dans ce fichier sont propres à cette application

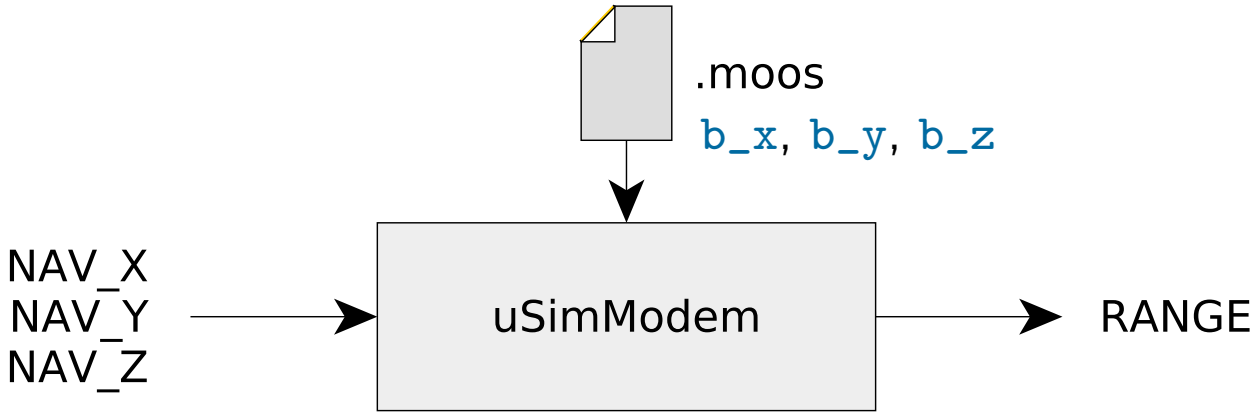


Rappel des cours 1 et 2

L'exemple de uSimModem

Pour l'application `uSimModem` :

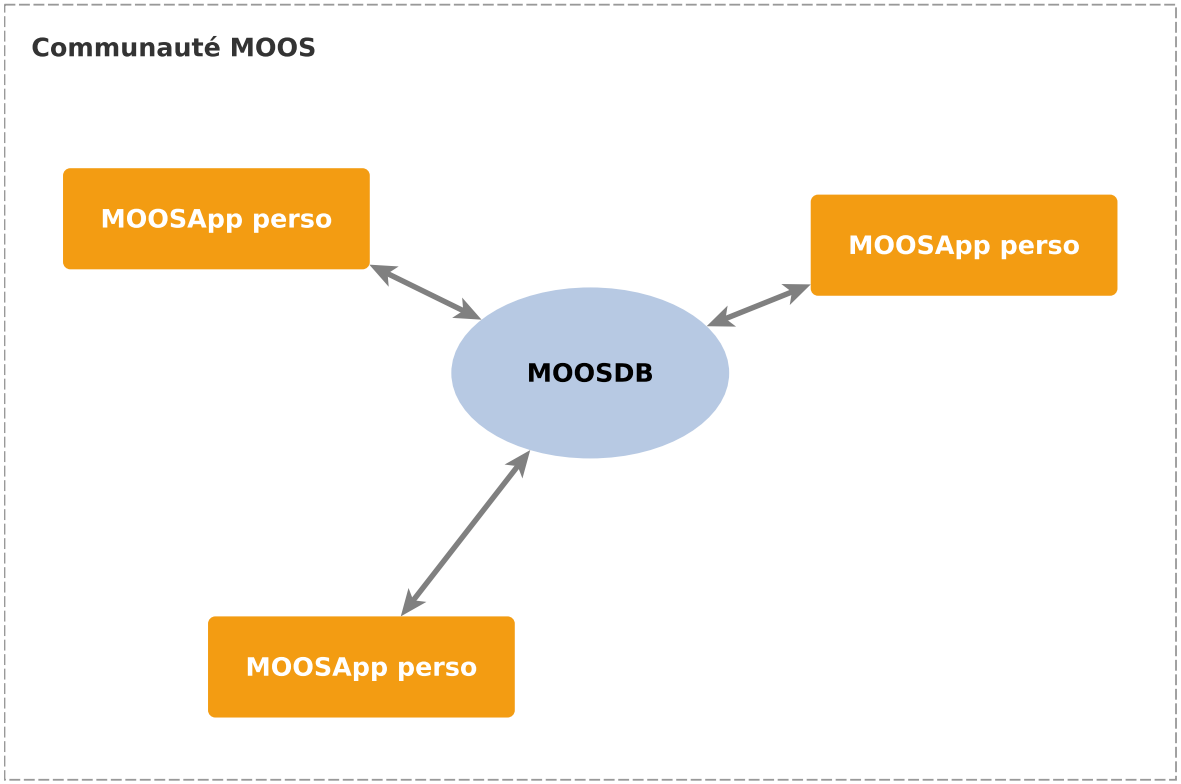
- abonnements : `NAV_X`, `NAV_Y`, `NAV_Z` (la position du robot)
- publications : `RANGE`
- configurations : `b_x`, `b_y`, `b_z` (la position de la balise)



Rappel des cours 1 et 2

Modularité d'une communauté MOOS

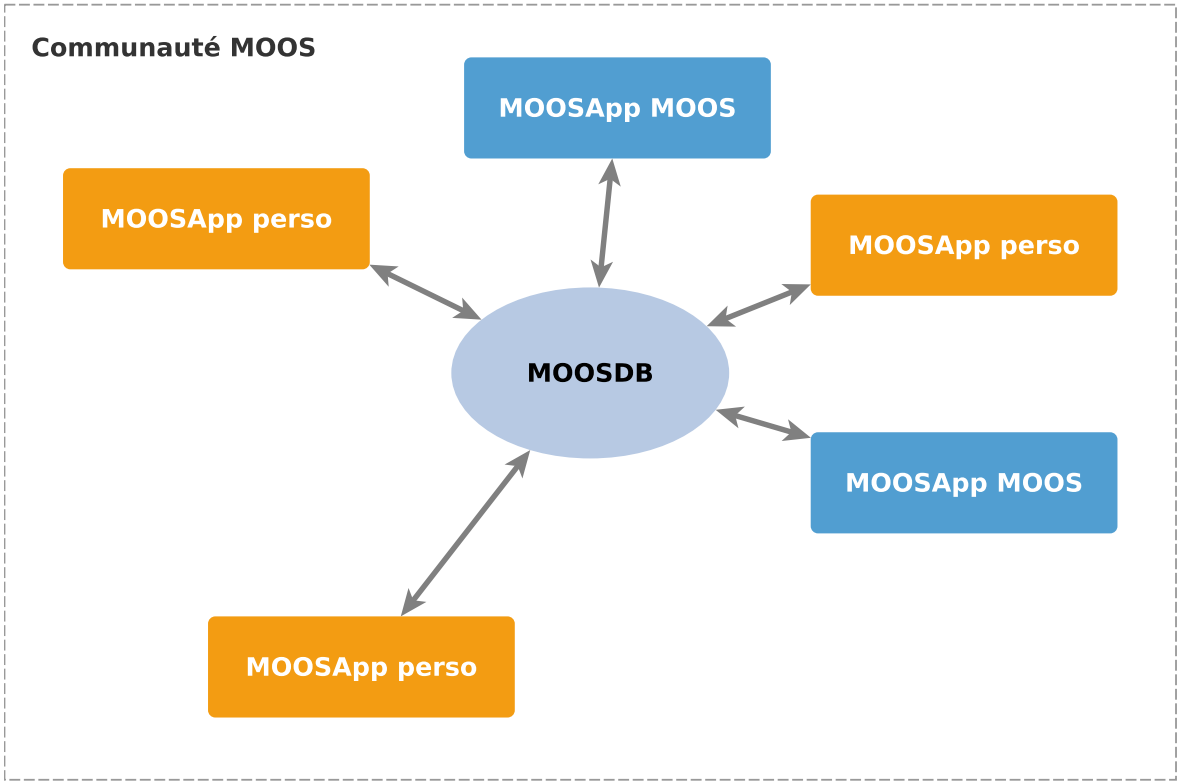
L'ajout d'une nouvelle `MOOSApp` se fait simplement en complétant le fichier `.moos` de la mission.



Rappel des cours 1 et 2

Modularité d'une communauté MOOS

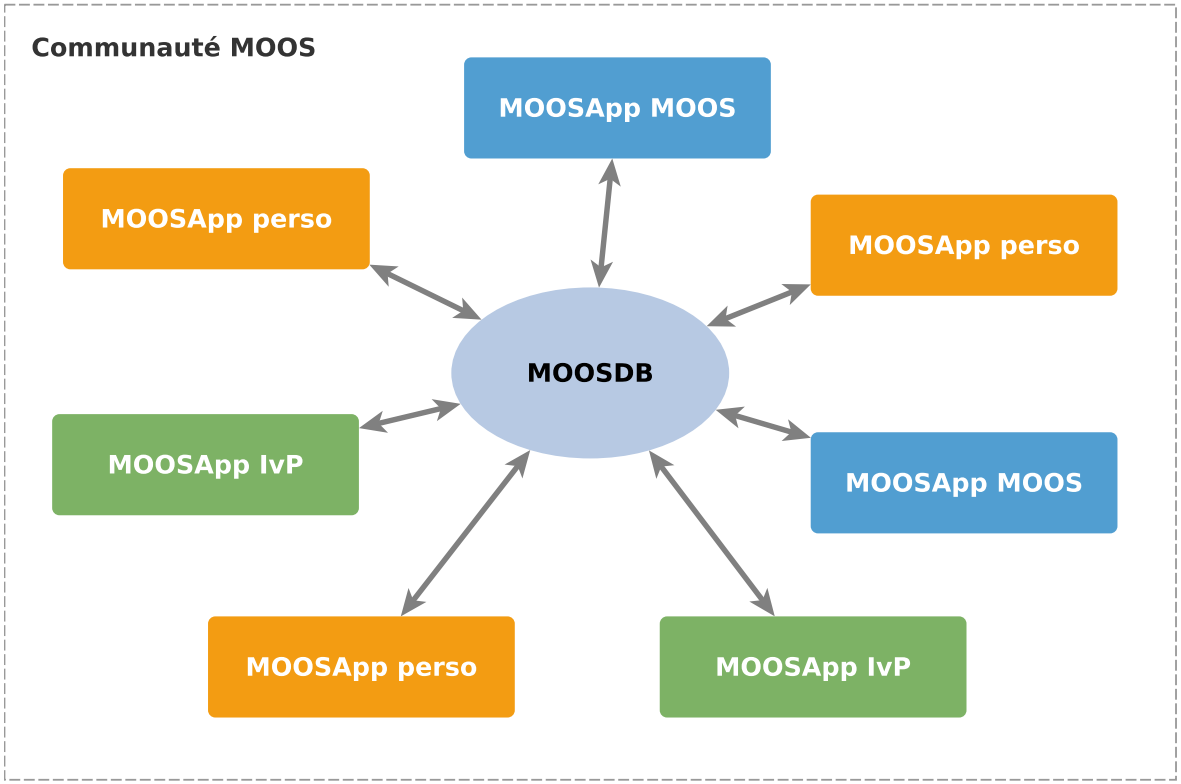
L'ajout d'une nouvelle `MOOSApp` se fait simplement en complétant le fichier `.moos` de la mission.



Rappel des cours 1 et 2

Modularité d'une communauté MOOS

L'ajout d'une nouvelle `MOOSApp` se fait simplement en complétant le fichier `.moos` de la mission.



Rappel des cours 1 et 2

Accès à la documentation des MOOSApp

- **Documentation rapide :**
  Dans un terminal, entrer l'une des lignes de commande :

```
> pMoosApplication -h
> pMoosApplication -e
> pMoosApplication -i
```

  - `-h` (`--help`) : donne le synopsis de l'application
  - `-e` (`--example`) : donne un exemple de configuration `.moos`
  - `-i` (`--interface`) : résume les publications et abonnements

- **Documentation complète :**
  Voir la documentation en ligne de MOOS-IvP sur
  `http://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php`

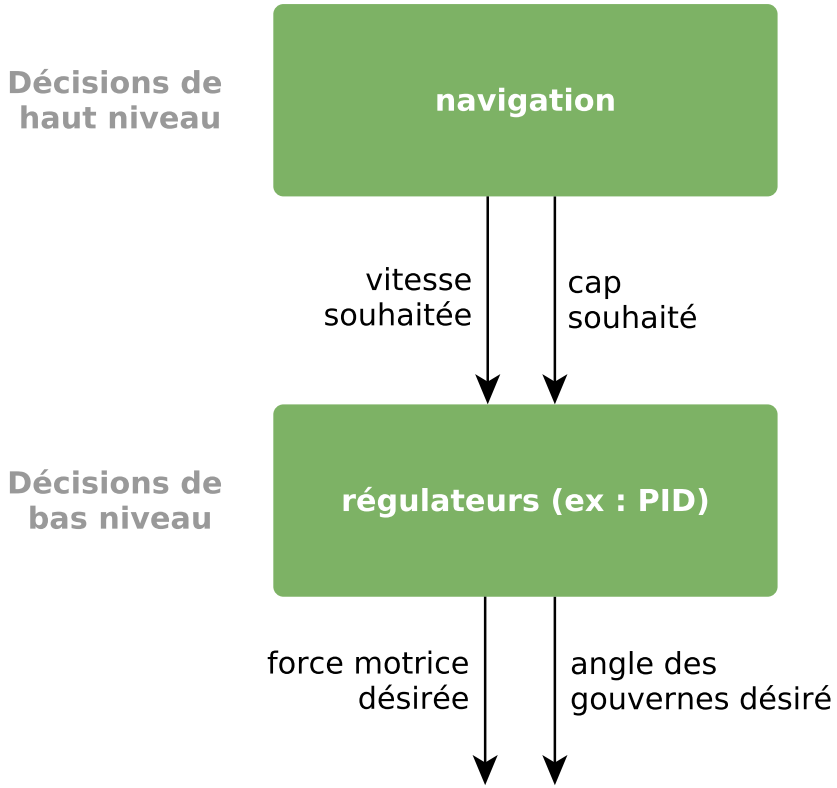
Section 2

Introduction

Introduction

De l'autonomie à différents niveaux

Plusieurs niveaux d'abstraction :

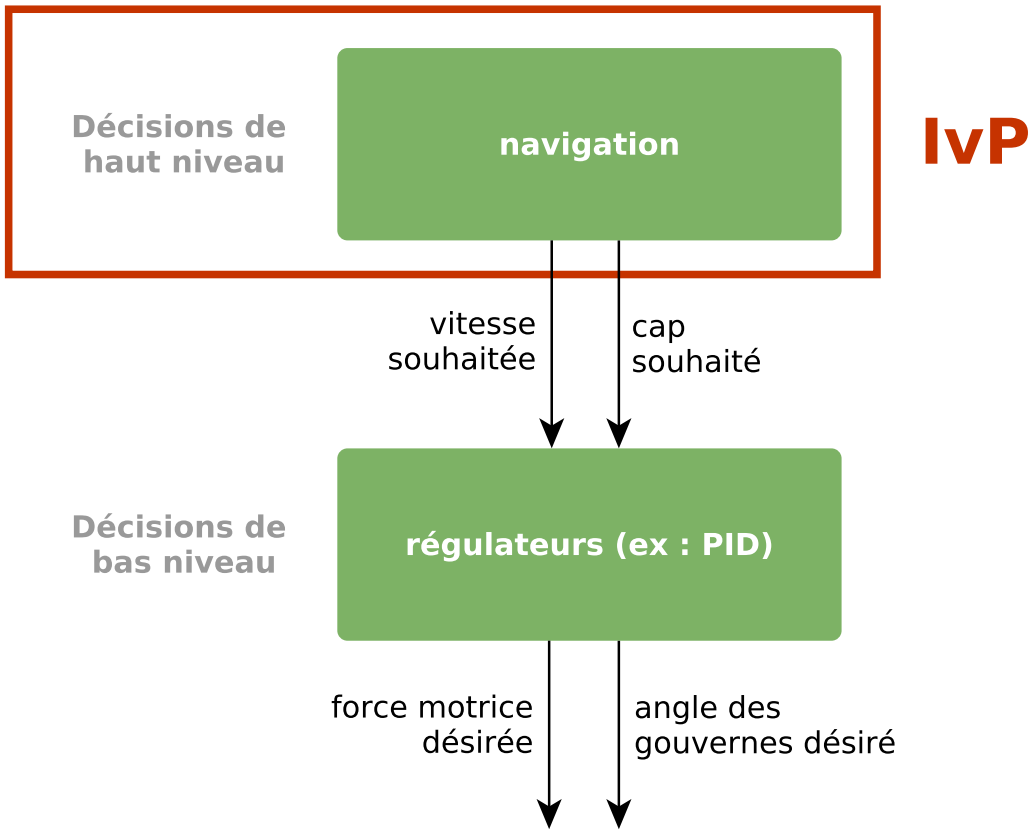


Dans ce cours, on s'intéresse aux décisions de haut niveau.

Introduction

De l'autonomie à différents niveaux

Plusieurs niveaux d'abstraction :



Dans ce cours, on s'intéresse aux décisions de haut niveau.

Introduction

Des robots semi-autonomes

Certains robots deviennent autonomes lorsqu'ils quittent leur zone de viabilité ou lorsqu'ils représentent une menace.

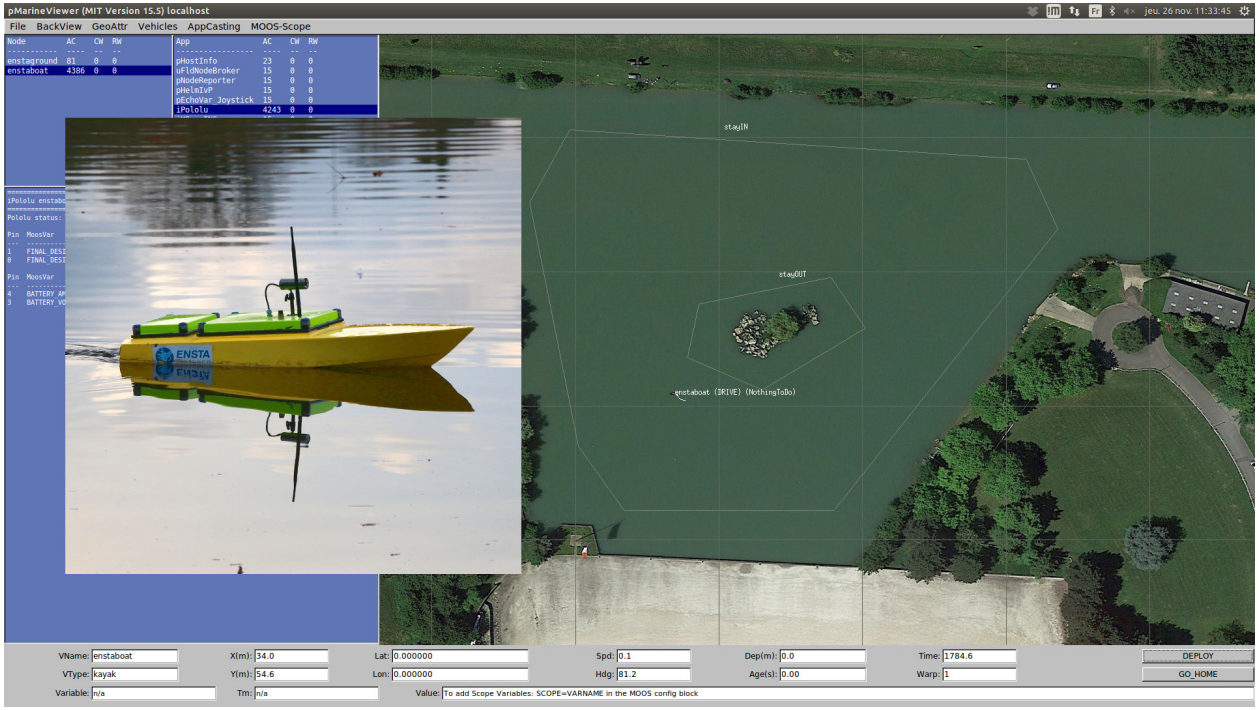


Figure – Un robot de surface de l'ENSTA Bretagne, équipé de MOOS-IvP
Démonstration d'une mission semi-autonome sur le lac de Polytechnique

Introduction

Des robots complètement autonomes

Des AUV [1] sont utilisés pour localiser l'épave du Boeing 777 de la Malaysia Airlines, à 4000m de profondeur au large de l'Australie.

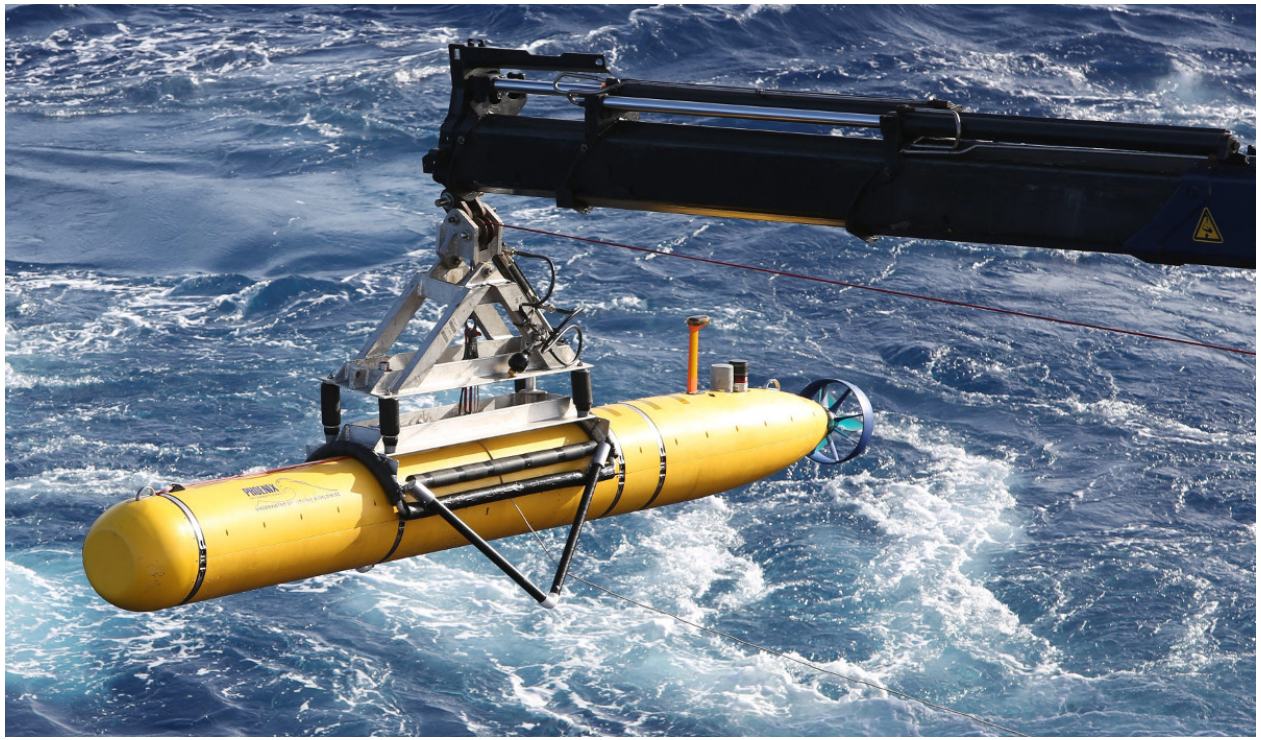


Figure – Un Bluefin 21, équipé de MOOS-IvP

1. Autonomous Underwater Vehicles

Introduction

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.

Exemple :
Coordonner une meute de robots dérivants à travers l'océan.

Introduction

De la modularité dans l'autonomie

Exemple :

Coordonner une meute de robots dérivants à travers l'océan.

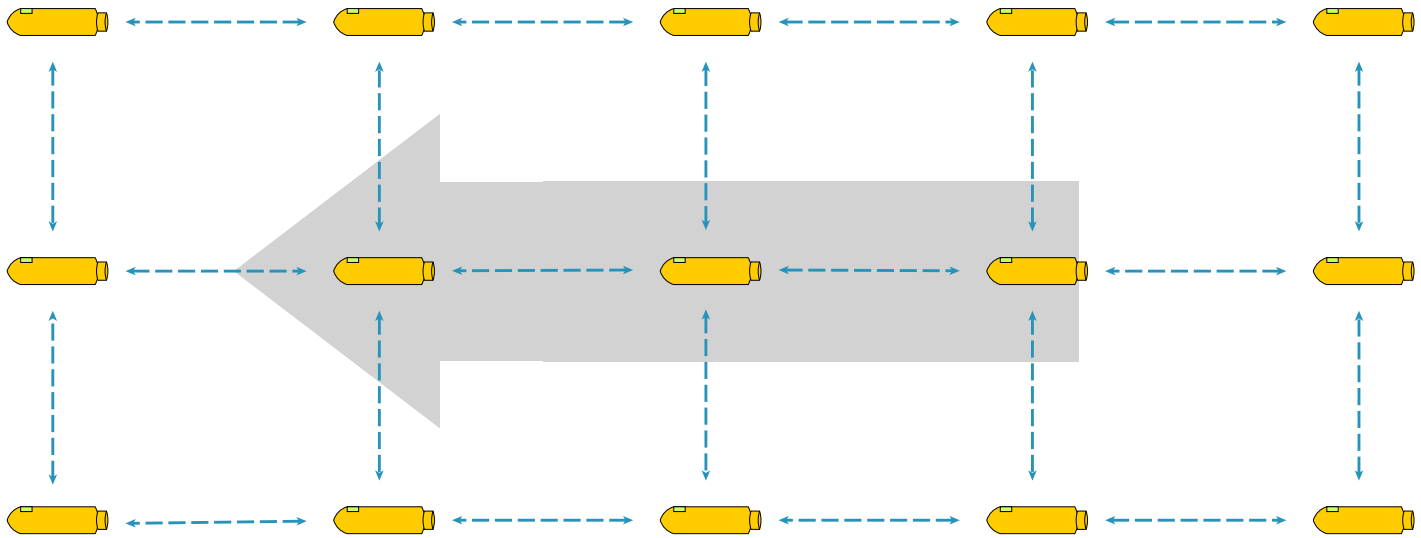


Figure – 15 robots autonomes avançant en se maintenant en formation

Introduction

De la modularité dans l'autonomie

Exemple :
Coordonner une meute de robots dérivants à travers l'océan.

La **loi de contrôle** de chaque robot devra prendre en compte :

- la direction souhaitée vers l'Ouest
- le positionnement correct au sein du groupe
  - et donc un placement relatif par rapport à chaque voisin

Que faire lorsqu'une **perturbation** se présente ?

- panne de l'un des robots
- plate-forme pétrolière sur la trajectoire du groupe

Introduction

De la modularité dans l'autonomie

Exemple :

Coordonner une meute de robots dérivants à travers l'océan.

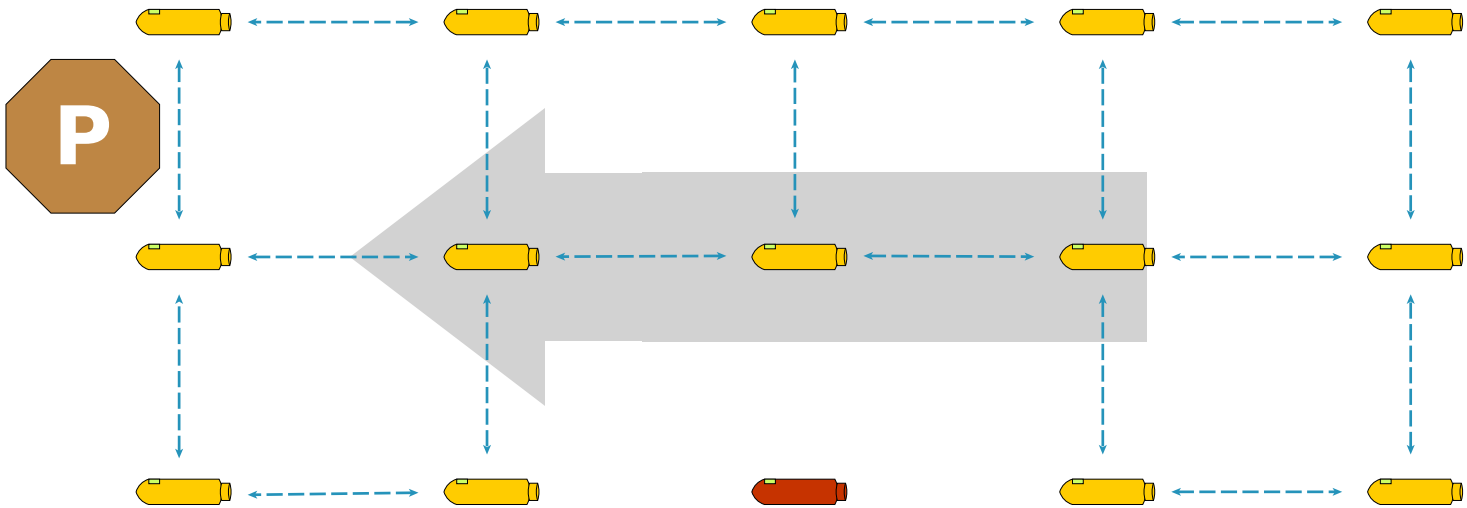


Figure – 15 robots autonomes avançant en se maintenant en formation – 2 perturbations surgissent : une plate-forme pétrolière sur la trajectoire du groupe ainsi qu'une panne de l'un des robots

Introduction

De la modularité dans l'autonomie

Exemple :

Coordonner une meute de robots dérivants à travers l'océan.

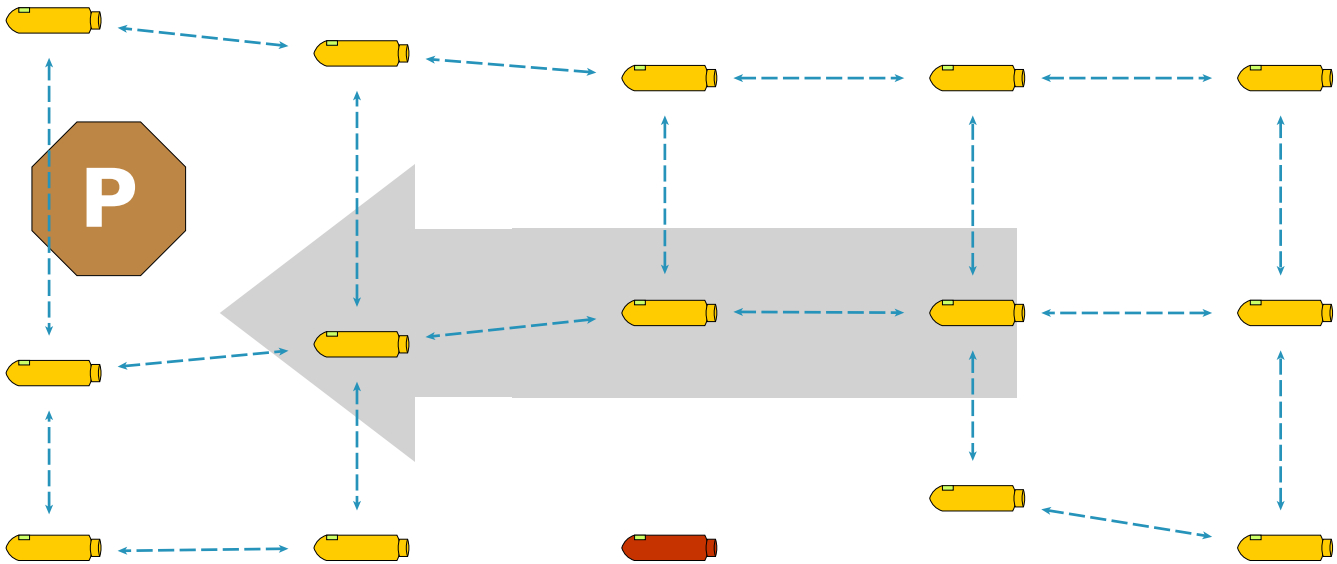


Figure – 15 robots autonomes avançant en se maintenant en formation – 2 perturbations surgissent : une plate-forme pétrolière sur la trajectoire du groupe ainsi qu'une panne de l'un des robots

Introduction

De la modularité dans l'autonomie

Exemple :

Coordonner une meute de robots dérivants à travers l'océan.

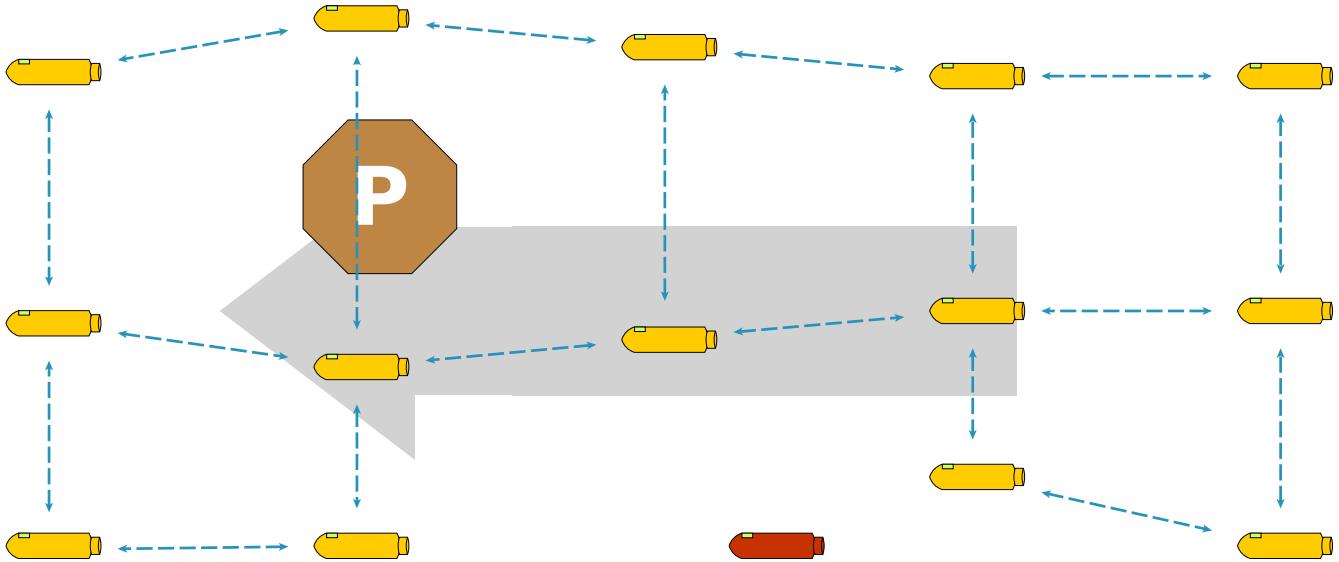


Figure – 15 robots autonomes avançant en se maintenant en formation – 2 perturbations surgissent : une plate-forme pétrolière sur la trajectoire du groupe ainsi qu'une panne de l'un des robots

Introduction

De la modularité dans l'autonomie

Exemple :

Coordonner une meute de robots dérivants à travers l'océan.

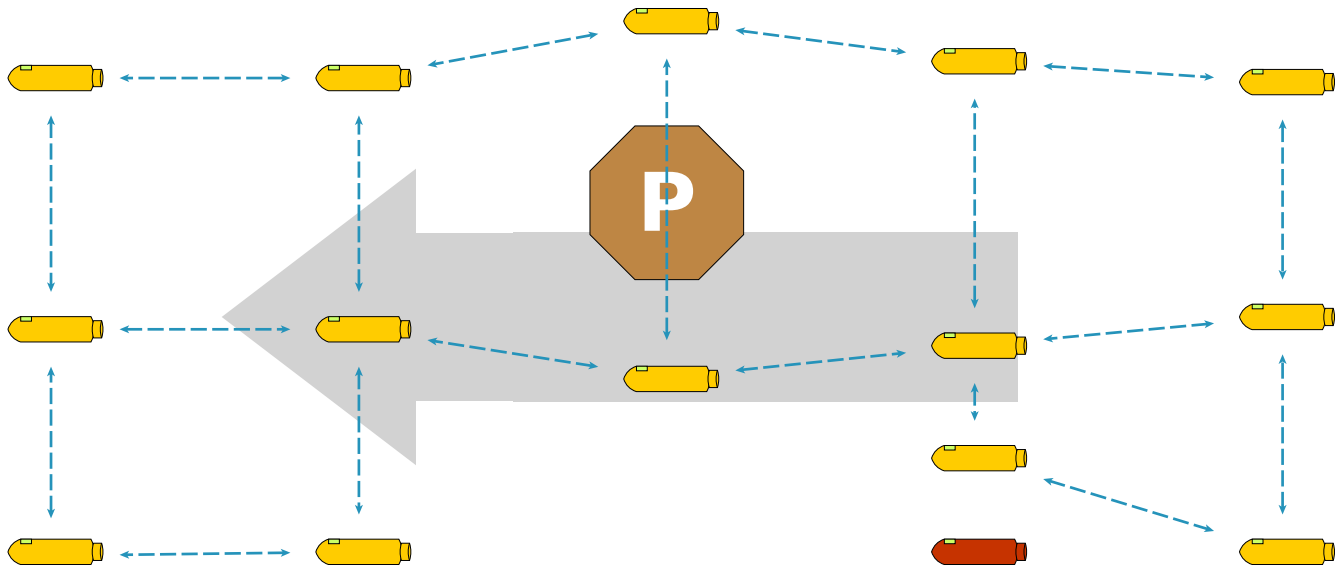


Figure – 15 robots autonomes avançant en se maintenant en formation – 2 perturbations surgissent : une plate-forme pétrolière sur la trajectoire du groupe ainsi qu'une panne de l'un des robots

Introduction

De la modularité dans l'autonomie

Exemple :

Coordonner une meute de robots dérivants à travers l'océan.

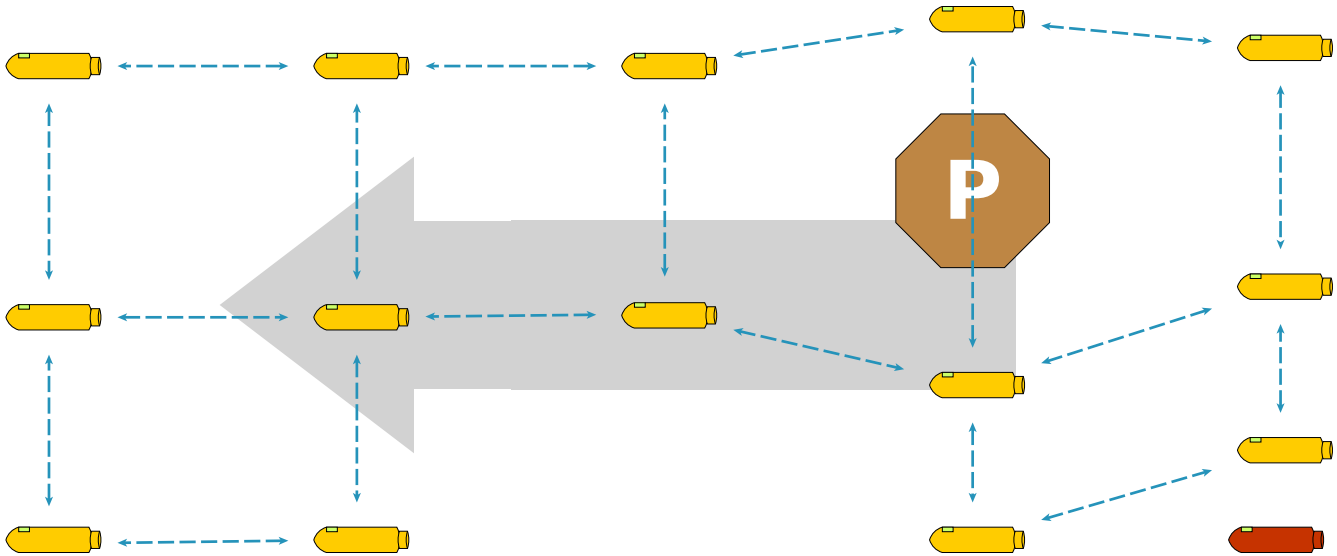


Figure – 15 robots autonomes avançant en se maintenant en formation – 2 perturbations surgissent : une plate-forme pétrolière sur la trajectoire du groupe ainsi qu'une panne de l'un des robots

Section 3

Gestion des processus de prise de décisions par IvP

Gestion des processus de prise de décisions par IvP

MOOS-IvP : une extension de MOOS

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

- **IvP** est une extension de **MOOS**
- des chercheurs du MIT [2] 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.

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

Gestion des processus de prise de décisions par IvP

MOOS-IvP : une extension de MOOS

L'élément phare de **MOOS-IvP** est son moteur de prise de décision, baptisé le **Helm** et généreusement livré avec de nombreuses **behaviors** prêtes à l'emploi :

- `Waypoint` : faisant suivre un chemin prédéfini à un véhicule
- `AvoidCollision` : gestion de collisions entre véhicules
- `PeriodicSurface` : permettant de faire surface régulièrement
- `ConstantDepth` : maintenant le robot à profondeur constante
- ...

On parle ici de « *behavior based architecture* ».

Gestion des processus de prise de décisions par 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

Gestion des processus de prise de décisions par 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.

Gestion des processus de prise de décisions par 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.

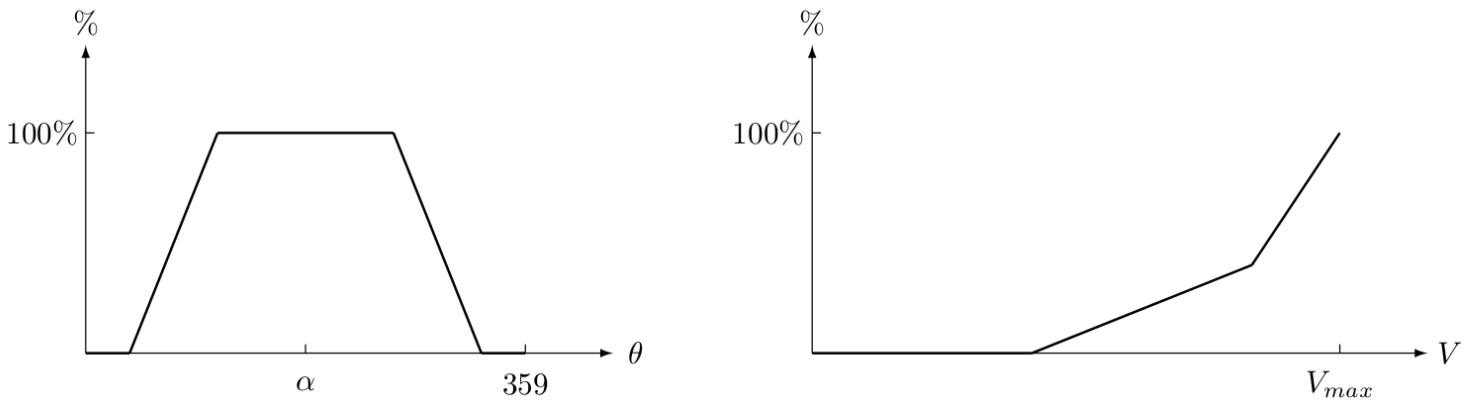
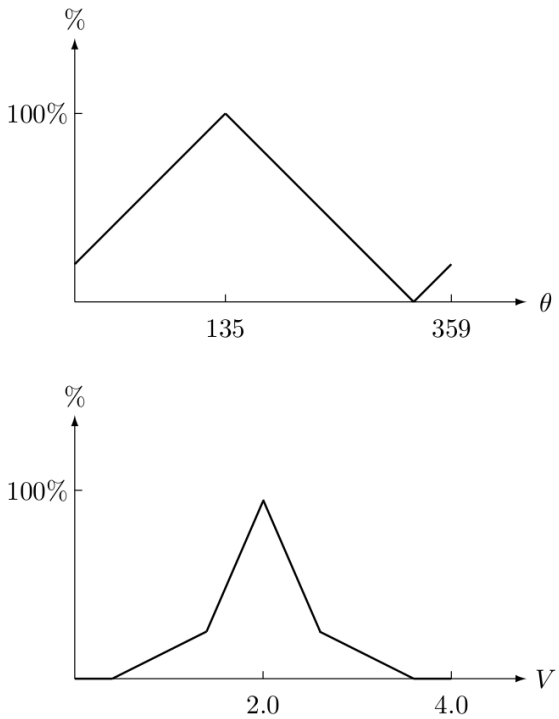


Figure – Des fonctions de prise de décision — ici, la décision proposée est d'aller *globalement* dans la direction de α en favorisant *franchement* la vitesse V_{max}.

Gestion des processus de prise de décisions par IvP

Behavior : produit cartésien parmi les variables de décision



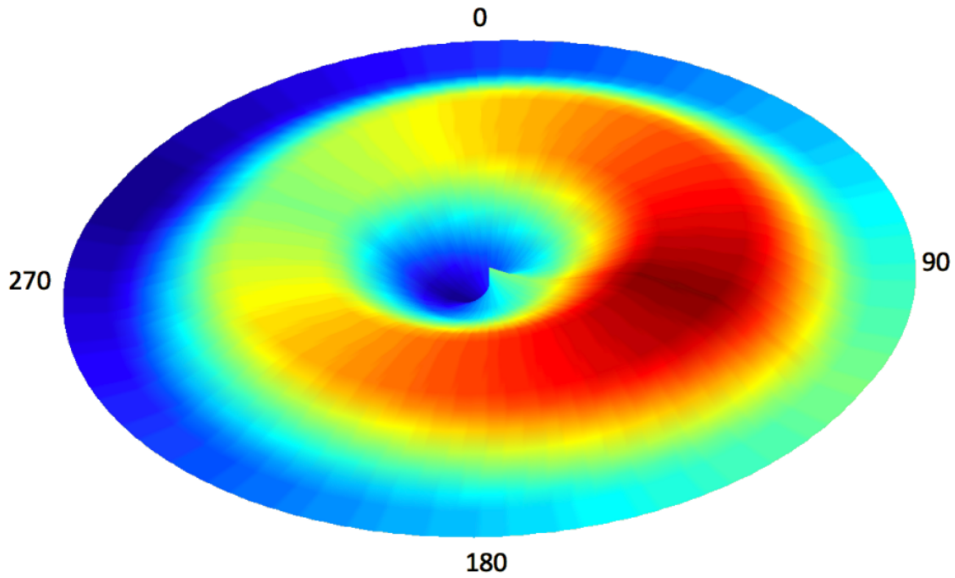


Figure – Représentation de fonctions IvP selon 2 domaines : le cap du robot de 0 à 360° et sa vitesse de 0 à 4m/s. Ici, le point le plus chaud donne la décision finale : $\theta_D = 135°$ et $V_D = 2.0$m/s.

Gestion des processus de prise de décisions par 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.

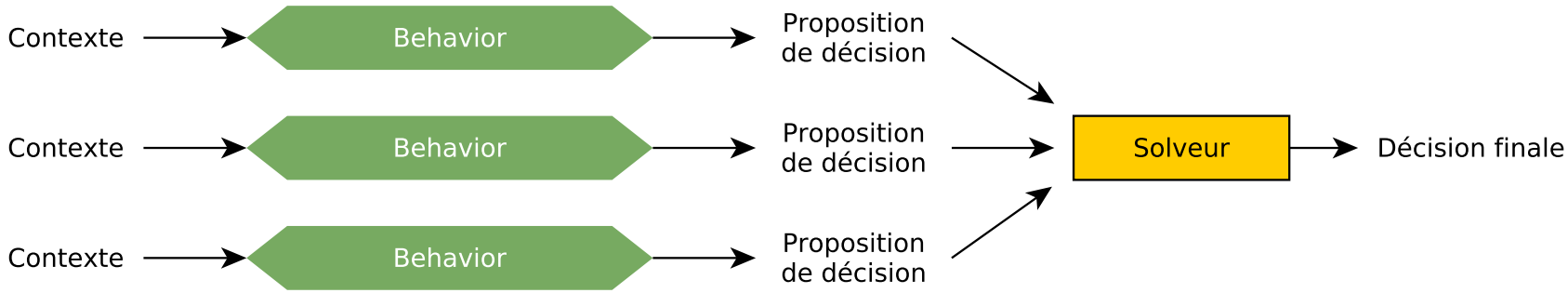


Figure – Plusieurs behaviors concurrentes – le solveur se charge de synthétiser un comportement en s'appuyant sur les fonctions IvP

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

Gestion des processus de prise de décisions par 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.

Gestion des processus de prise de décisions par IvP

Comment les poids sont-ils associés aux behaviors ?

On distingue donc 2 poids pour une behavior :

- un poids **dynamique** fixé par le concepteur
  qui change en fonction du contexte : c'est la **fonction IvP** f_i
- un poids **constant** w_i fixé par l'utilisateur
  qui détermine l'importance attribuée à cette behavior

Par exemple : une behavior d'évitement d'obstacle

- son poids f_i est maximal lorsqu'il y a risque de collision
- son poids f_i est nul lorsque l'objet est suffisamment éloigné
- avec w_i, l'utilisateur peut choisir de relativiser ce comportement d'évitement en favorisant d'autres behaviors

Gestion des processus de prise de décisions par 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.

Gestion des processus de prise de décisions par IvP

Différents degrés de tolérance pour une behavior

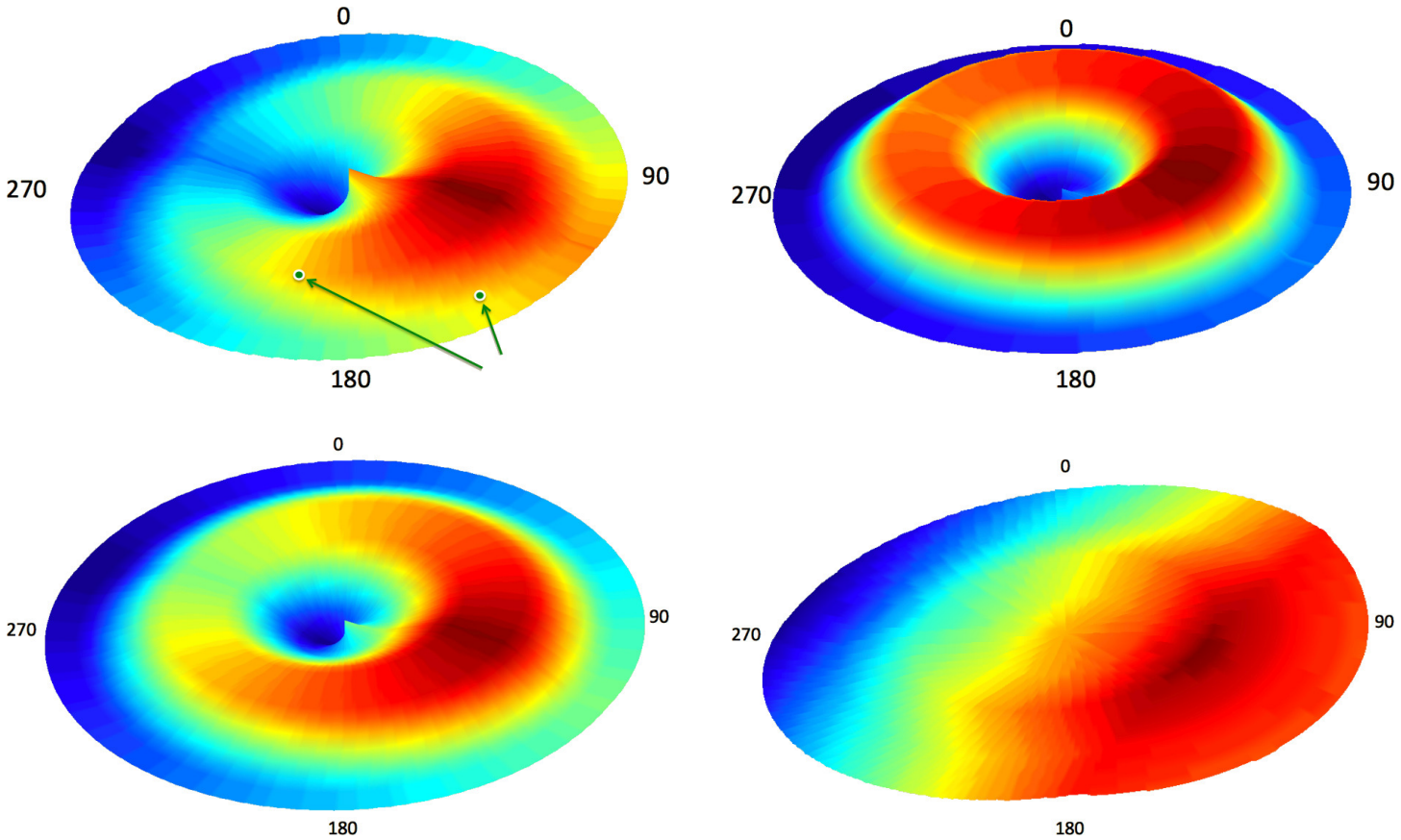


Figure – Différents profils de décision pour un même contexte – dans chaque cas, la décision est la même (au point chaud) – mais la décision sera différente en présence d'autres behaviors

Gestion des processus de prise de décisions par IvP

Un exemple de behavior dominante

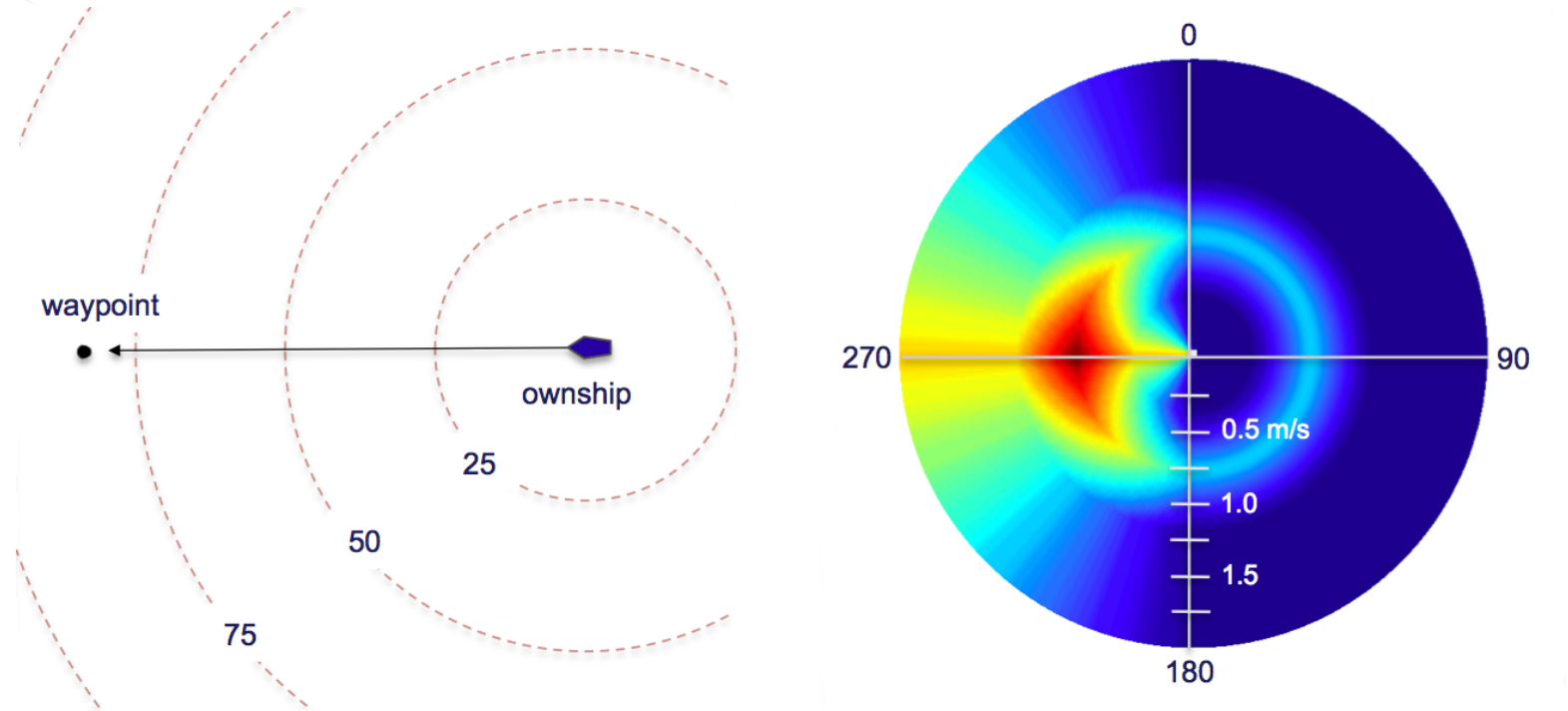


Figure – Suivi de cibles – à gauche, le contexte ; à droite, la projection des fonctions IvP – le robot est clairement amené à se diriger à l'Ouest pour atteindre son waypoint.

Gestion des processus de prise de décisions par IvP

Un exemple de behavior tolérante

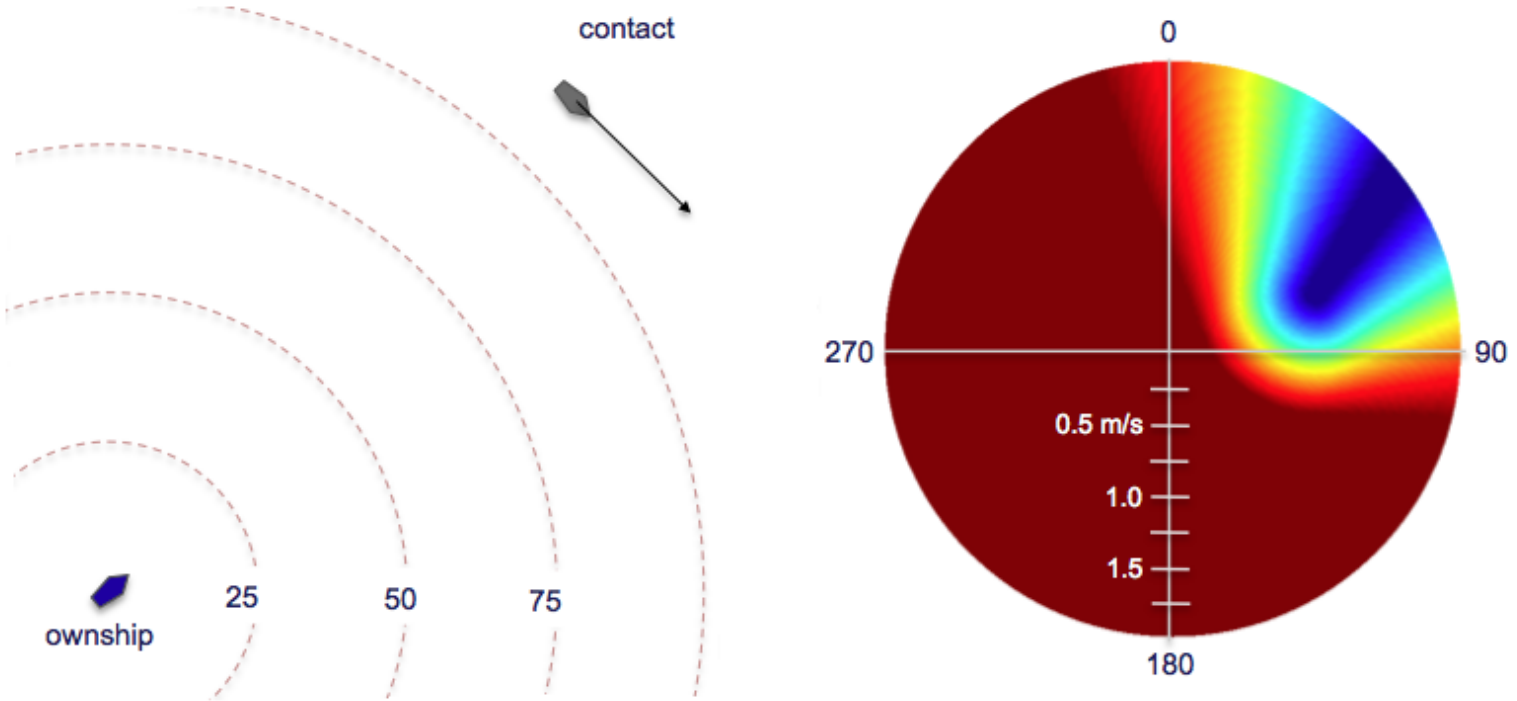
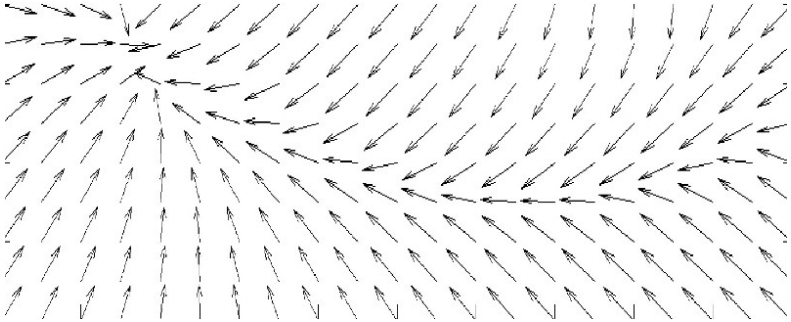


Figure – Évitement d'obstacles – à gauche, le contexte ; à droite, la projection des fonctions IvP – le robot est libre dans ses déplacements tant qu'il ne se dirige pas sur l'obstacle.

Gestion des processus de prise de décisions par 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

Gestion des processus de prise de décisions par 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.

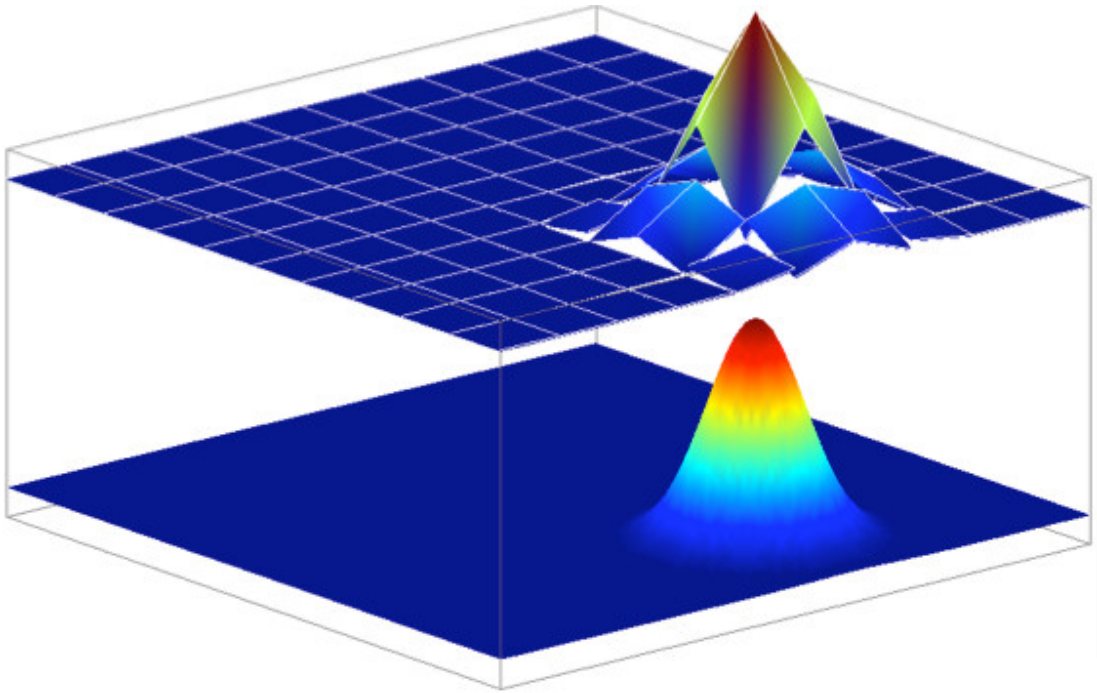
L'utilisateur n'a pas besoin de connaître les fonctions IvP d'une behavior pour s'en servir.

Gestion des processus de prise de décisions par IvP

IvP : la notion d'Interval Programming

Interval Programming $\neq$ Interval Analysis

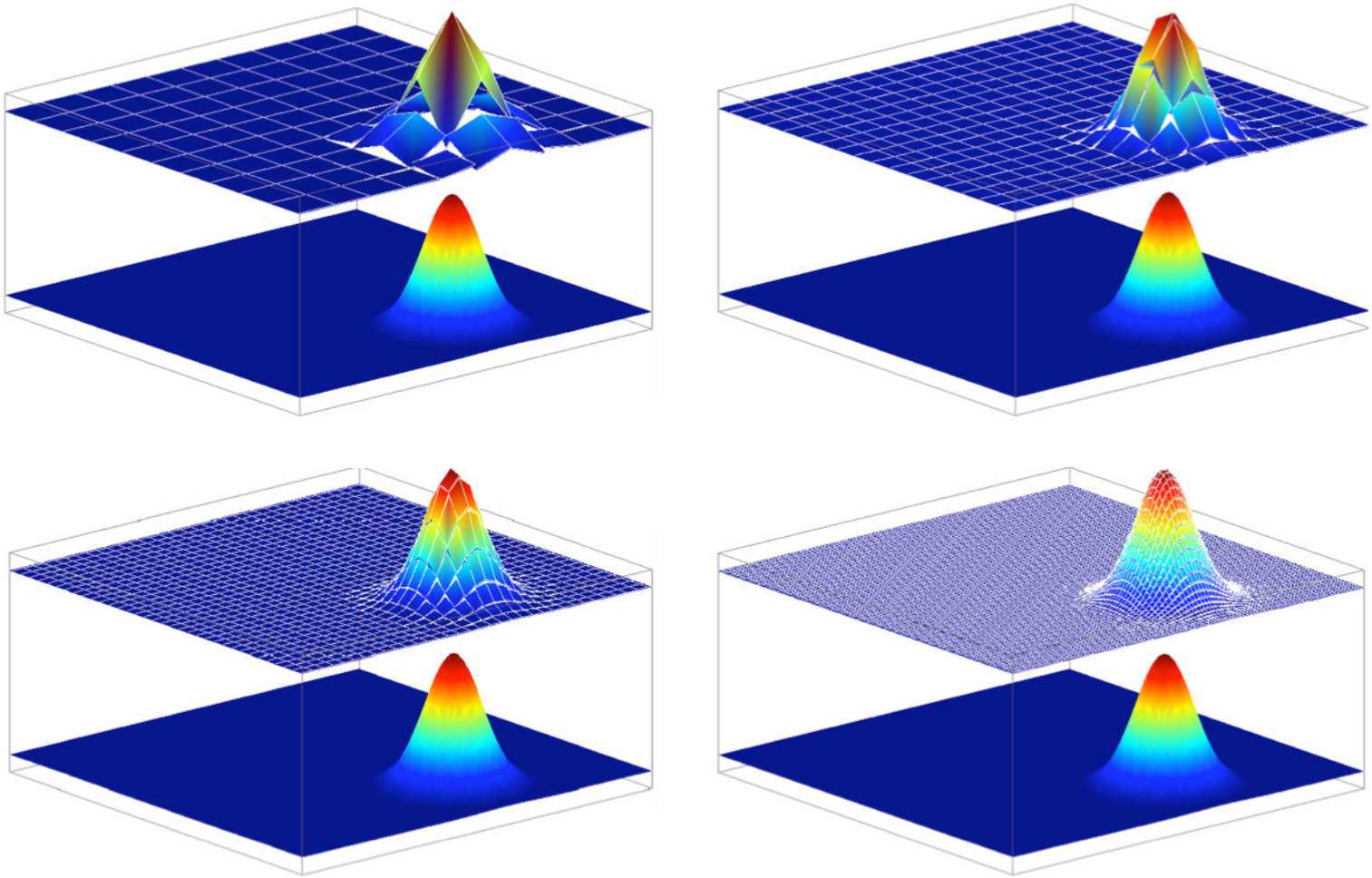
Une décision est prise sur un domaine en fonction des poids calculés en chaque point de ce domaine. En pratique, ce domaine est **discrétisé** : il est caractérisé par un ensemble d'**intervalles**.



Gestion des processus de prise de décisions par IvP

IvP : la notion d'Interval Programming

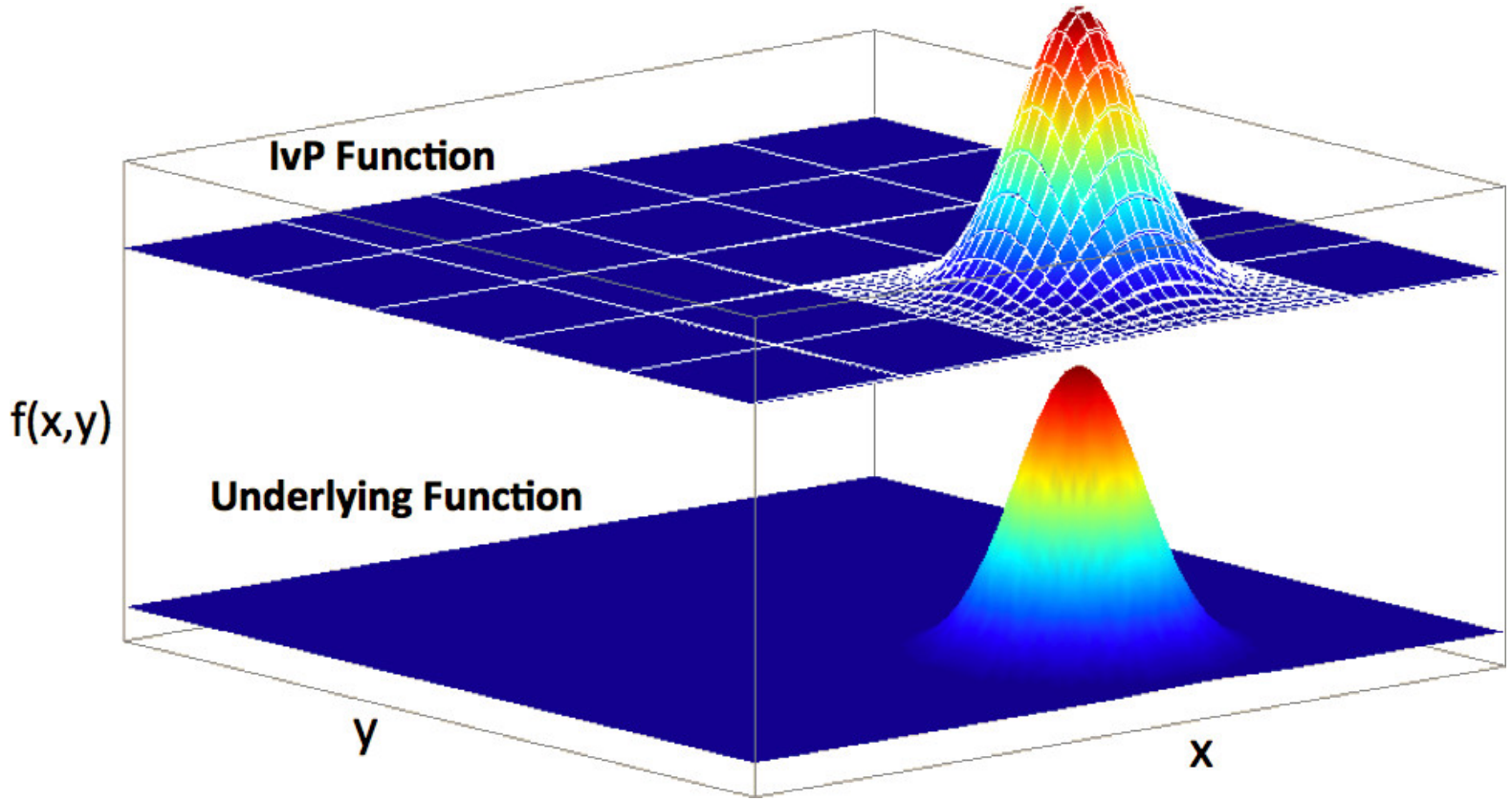
Plus la précision du domaine sera grande, plus la décision sera fine.
Le calcul en sera d'autant plus conséquent.



Gestion des processus de prise de décisions par IvP

IvP : la notion d'Interval Programming

IvP propose un découpage optimisé du domaine :



Section 4

Mise en place dans une communauté MOOS

Mise en place dans une communauté MOOS

L'application pHelmIvP

Le solveur IvP est hébergé par la MOOSApp `pHelmIvP` :

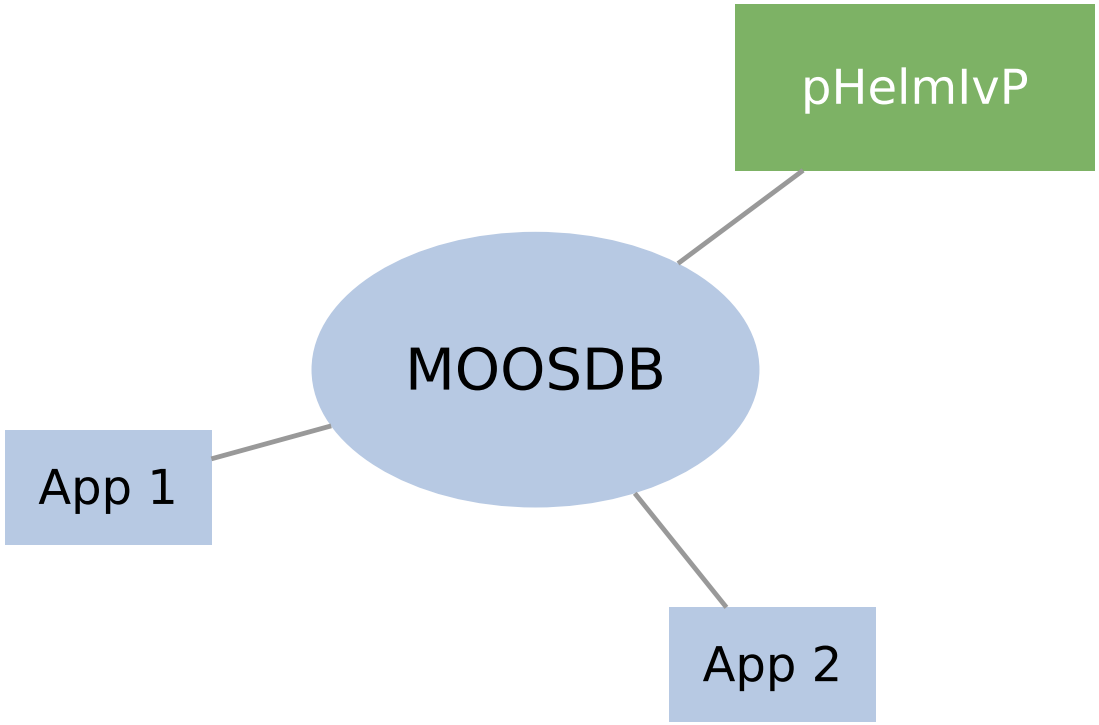
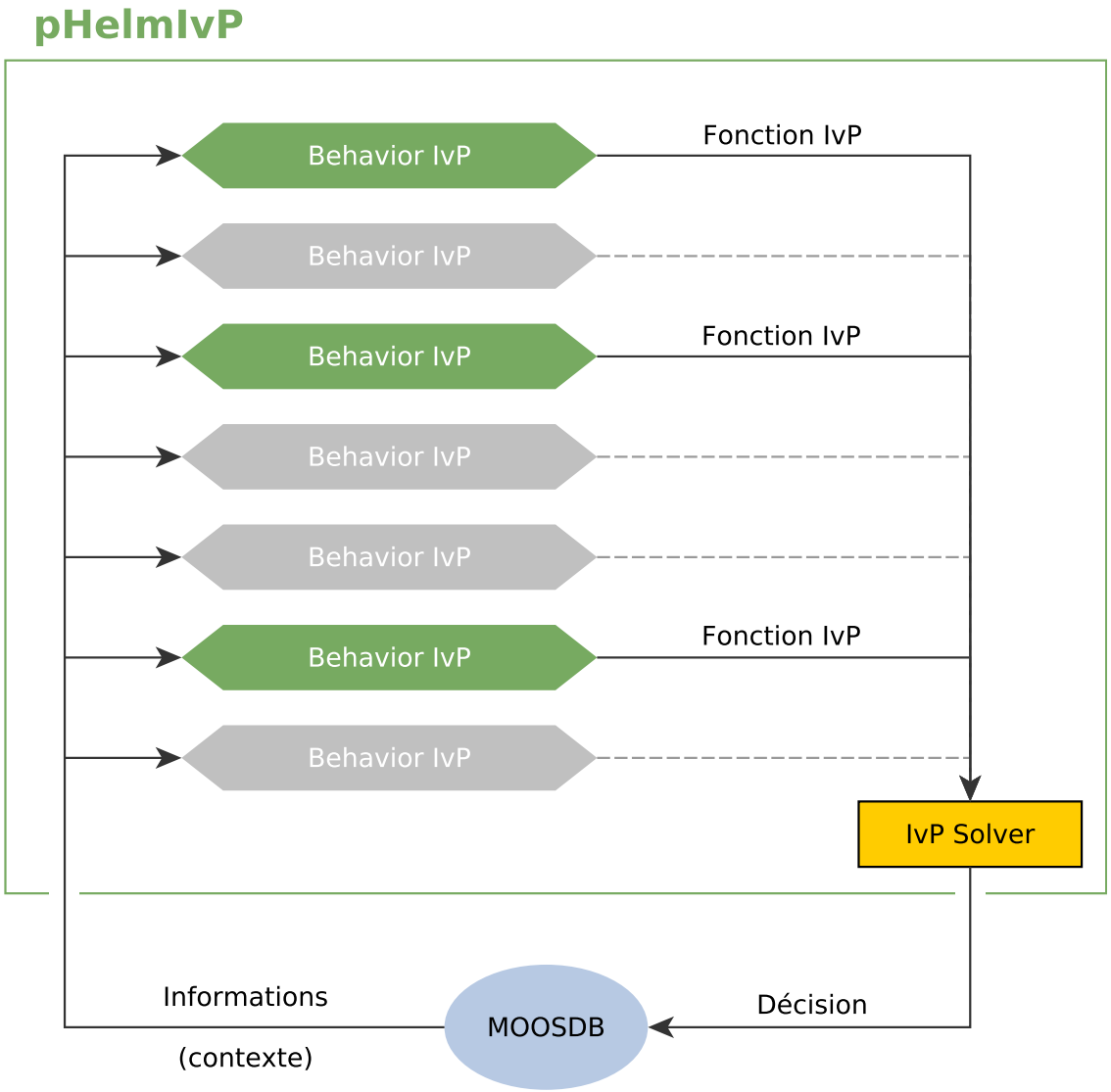


Figure – L'application `pHelmIvP` prend place dans une communauté

Mise en place dans une communauté MOOS

L'application pHelmIvP

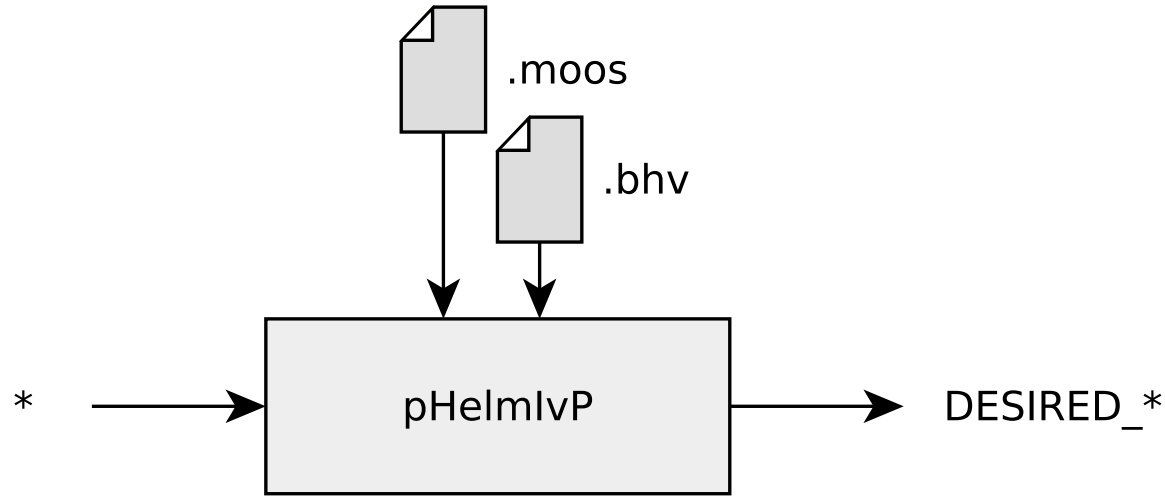


Mise en place dans une communauté MOOS

L'application pHelmIvP

L'application `pHelmIvP` :

- est configurable dans un `.moos`
- s'abonne aux variables requises par ses behaviors (contexte)
- publie, notamment, des décisions : `DESIRED_*`



Un fichier `.bhv` permet de configurer les behaviors de la mission.

Mise en place dans une communauté MOOS

L'application pHelmIvP : configuration du .moos

Exemple de configuration pour un robot sous-marin.

Fichier `exemple.moos` :

```
1   ProcessConfig = pHelmIvP
2   {
3     AppTick   = 4 // fréquence d'exécution de la méthode Iterate
4     CommsTick = 4 // fréquence d'analyse des messages reçus
5
6     // Configuration des chemins d'accès
7     ivp_behavior_dir = /home/user/moos-ivp-user/lib/
8     behaviors        = filename.bhv
9
10    // Définitions des domaines de décision
11    domain           =  course:0:359:360 // cap
12    domain           =  speed:0:2:21
13    domain           =  depth:0:1000:1001
14    // domaine       =  variable:valeur_min:valeur_max:nb_points
15    ...
16  }
```

Mise en place dans une communauté MOOS

L'application pHelmIvP : configuration du .bhv

Exemple de configuration pour un robot sous-marin.

Fichier `exemple.bhv` :

```
1   // Définition de variables communes aux behaviors
2   // Elles permettent, par exemple, de décomposer une mission
3   initialize  DEPLOY = false
4   initialize  RETURN = false
5
6   // Configuration d'une behavior de suivi de point
7   Behavior = BHV_Waypoint
8   {
9     ...
10  }
11
12  // Configuration d'une behavior de profondeur désirée
13  Behavior = BHV_ConstantDepth
14  {
15    ...
16  }
```

Mise en place dans une communauté MOOS

Paramètres communs à toutes les behaviors

Toutes les behaviors héritent d'une classe permettant une bonne intégration des comportements dans IvP. Ainsi, certains paramètres sont communs :

```
1   Behavior = BHV_Quelconque
2   {
3     name       = WAYPT_RETURN
4     pwt        = 100 // "poids" de la behavior (relatif)
5     condition = RETURN = true // condition d'exécution
6     condition = DEPLOY = true
7   }
```

De la même manière que pour les MOOSApp, les behaviors sont configurées par blocs dans un fichier dédié : `.bhv`.
Les paramètres qui nous intéressent sont à préciser.

Section 5

Présentation de quelques behaviors IvP

Présentation de quelques behaviors IvP

La behavior BHV_StationKeep

Comportement : atteindre une position et s'y maintenir.

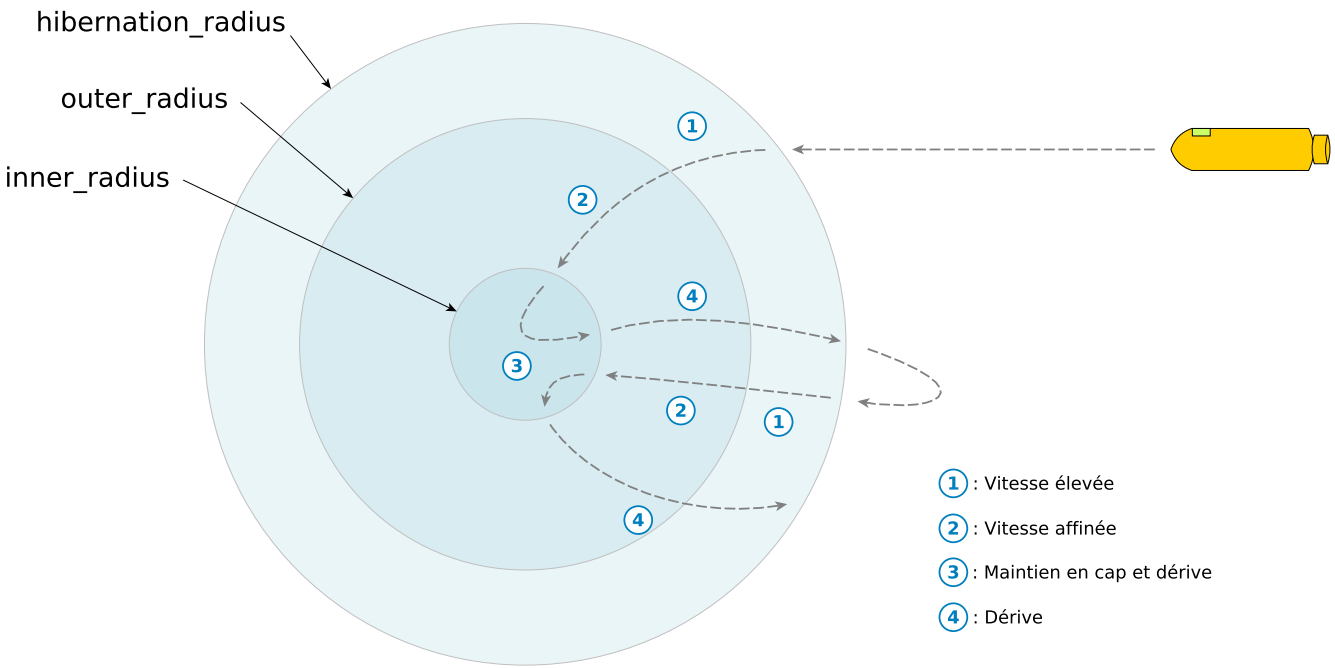


Figure – *Station Keeping* – la position à atteindre est définie par `inner_radius` et une dérive est tolérée tant que le robot reste dans la zone déterminée par `hibernation_radius`. Le paramètre `outer_radius` permet de préciser une vitesse intermédiaire.

Présentation de quelques behaviors IvP

La behavior BHV_StationKeep

Exemple de configuration dans un fichier `.bhv` :

```
1   Behavior = BHV_StationKeep
2   {
3     // Configuration générale de la behavior
4     name         = station-keep
5     pwt          = 100              // poids
6     condition    = MODE==SKEEPING // condition d'exécution
7
8     // Configuration propre à cette behavior
9     station_pt     = 100,250
10    inner_radius   = 4
11    outer_radius   = 15
12    outer_speed    = 1.2
13    transit_speed  = 2.5
14  }
```

Voir la documentation complète sur :
`http://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=Helm.BehaviorStationKeep`

Présentation de quelques behaviors IvP

La behavior BHV_Waypoint

Comportement : suivre une série de points prédéfinis.

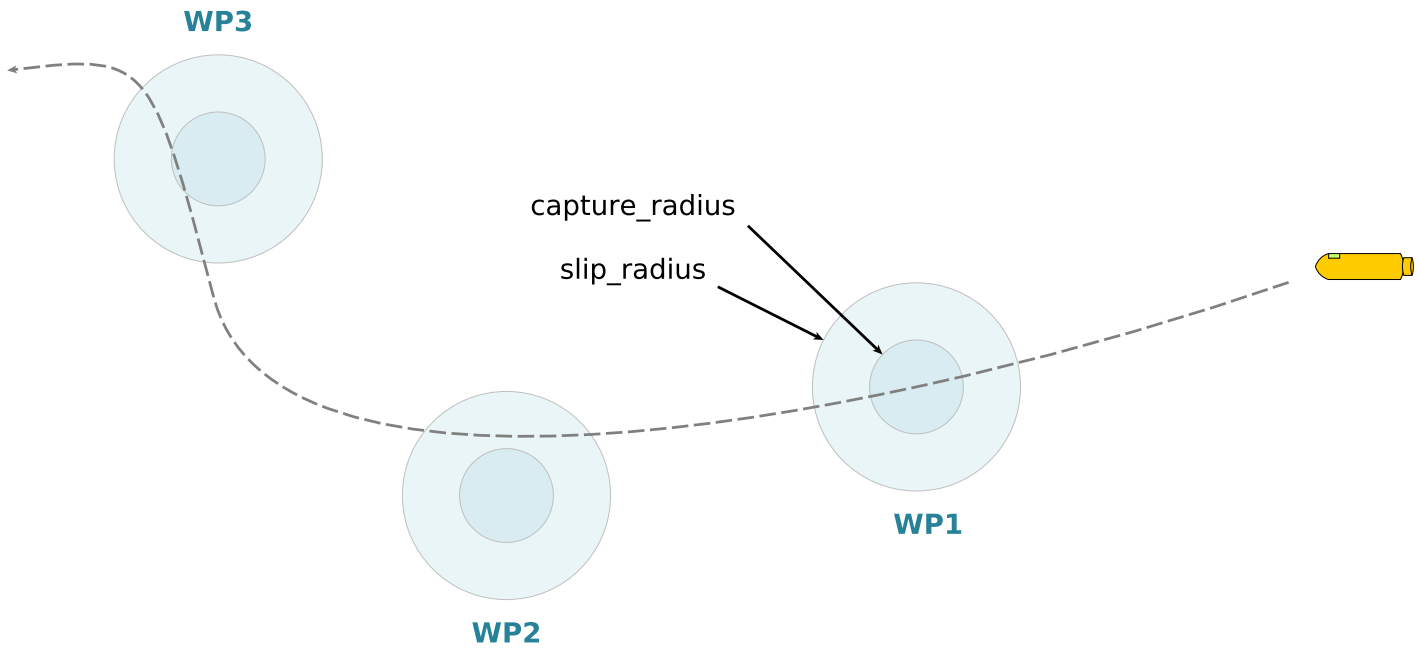


Figure – *Waypoint* – la position à atteindre est définie par `capture_radius` et le waypoint est validé si le robot s'éloigne de ce point après avoir traversé `slip_radius`

Présentation de quelques behaviors IvP

La behavior BHV_Waypoint

Exemple de configuration dans un fichier `.bhv` :

```
1   Behavior = BHV_Waypoint
2   {
3     // Configuration générale de la behavior
4     name         = transit
5     pwt          = 100                  // poids
6     condition    = MODE==TRANSITING // condition d'exécution
7
8     // Configuration propre à cette behavior
9     capture_radius = 3
10    capture_line = false
11    points = pts={-200,-200:30,-60} // liste de points à suivre
12    slip_radius = 15
13    speed = 1.2
14  }
```

Voir la documentation complète sur :
`http://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=Helm.BehaviorWaypoint`

Présentation de quelques behaviors IvP

La behavior BHV_Waypoint

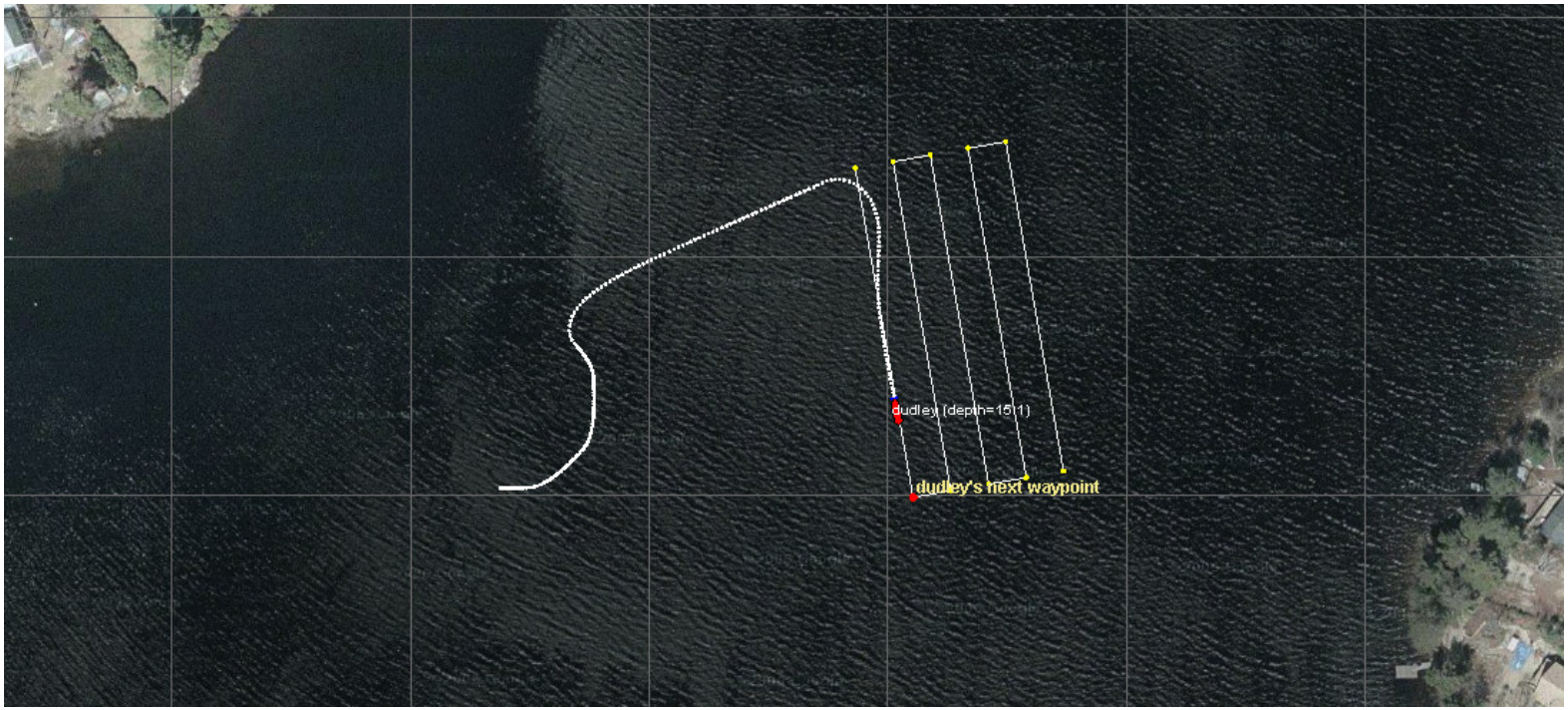


Figure – Exemple d'utilisation pour un suivi de ligne – la behavior affiche elle-même des informations dans l'interface de visualisation `pMarineViewer`

Présentation de quelques behaviors 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.

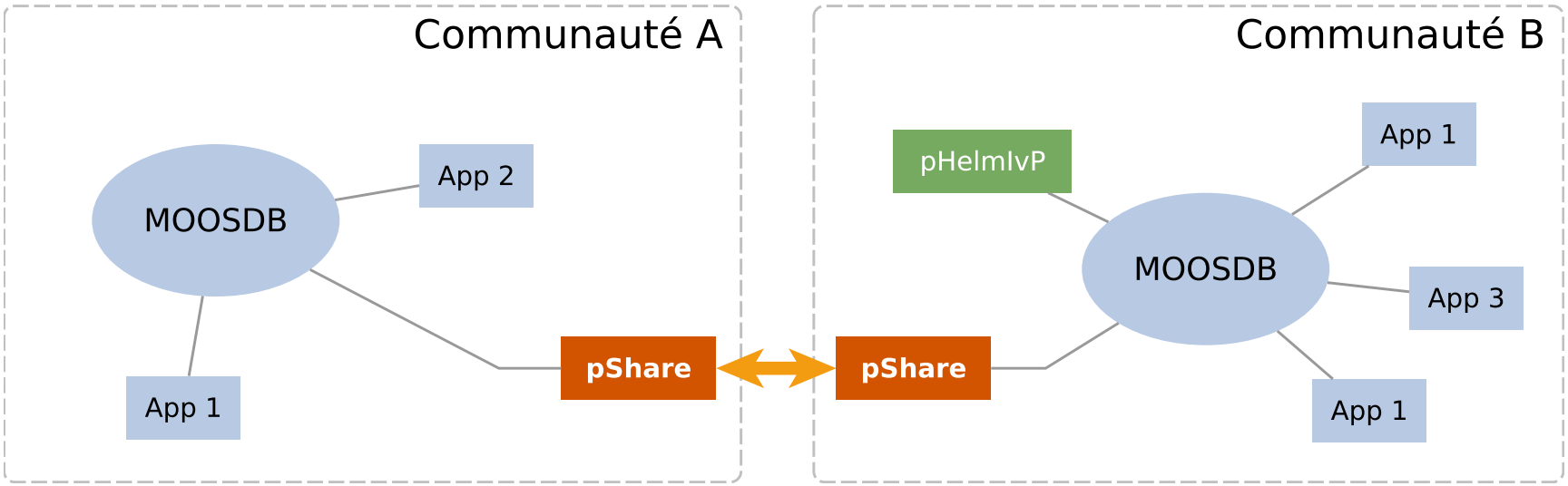


Figure – Évitement d'obstacle – le robot *B* va chercher à éviter le robot *A*. La position du robot *A* est communiquée au robot *B* par la MOOSApp `pShare` ; cette information est lue par la behavior `BHV_AvoidCollision`.

Présentation de quelques behaviors IvP

La behavior BHV_AvoidCollision

Exemple de configuration dans un fichier `.bhv` :

```
1   Behavior = BHV_AvoidCollision
2   {
3     // Configuration générale de la behavior
4     name         = avdcollision
5     pwt          = 200           // poids
6     condition    = AVOID = true // condition d'exécution
7
8     // Identification du robot à éviter
9     contact = henry // nom de la communauté MOOS du robot
10
11    // Configuration propre à cette behavior
12    completed_dist = 500
13    pwt_inner_dist = 50
14    pwt_outer_dist = 200
15  }
```

Voir la documentation complète sur :
http://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=Helm.BehaviorAvdCollision

Présentation de quelques behaviors IvP

La behavior BHV_AvoidCollision

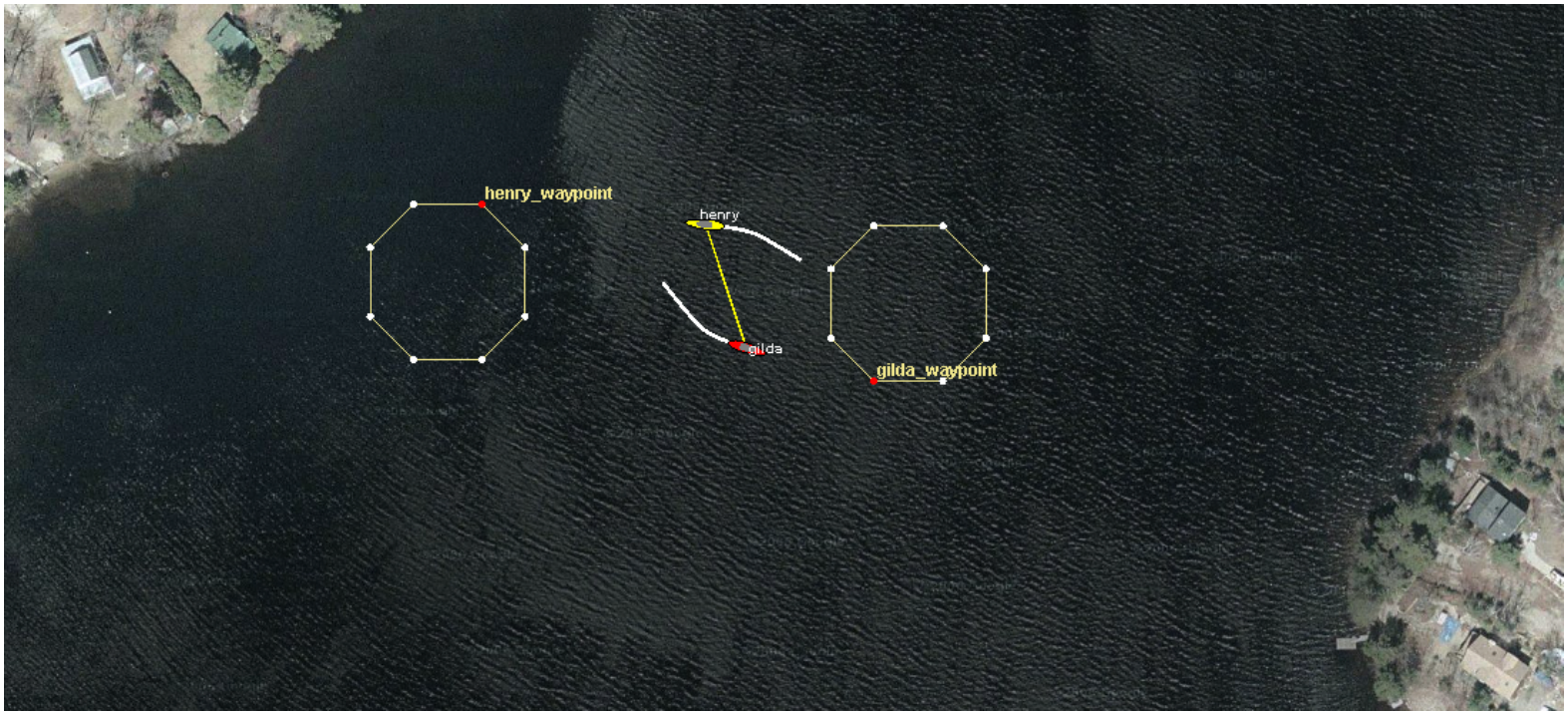


Figure – Exemple d'utilisation pour un évitement de robot – ici, les deux robots poursuivent leur mission (passer d'un polygone à l'autre) tout en évitant une collision. Une fusion de fonctions IvP a eu lieu et a permis un changement de comportement en douceur.

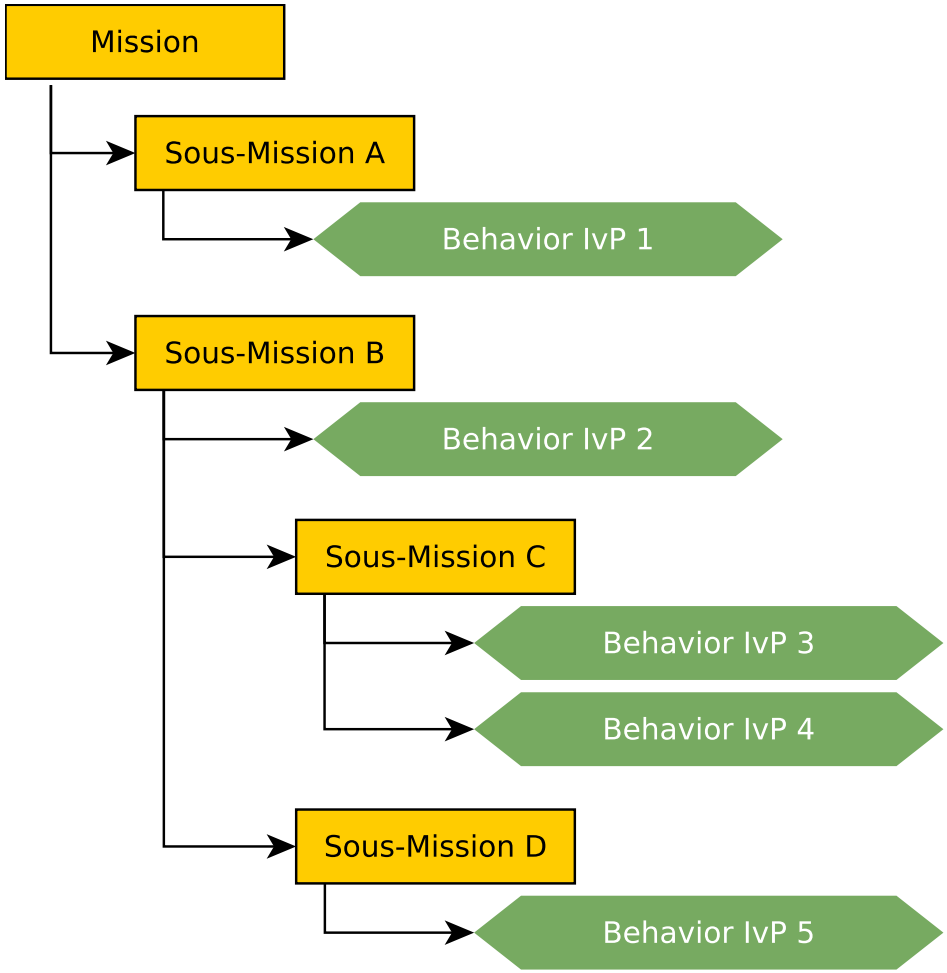
Section 6

Organisation d'une mission

Organisation d'une mission

Découpage d'une mission en sous-missions

Lorsque les missions se complexifient, il convient d'organiser les tâches. Par exemple :



Organisation d'une mission

Découpage d'une mission en sous-missions

Chaque sous-mission est appelée « *mode* ».

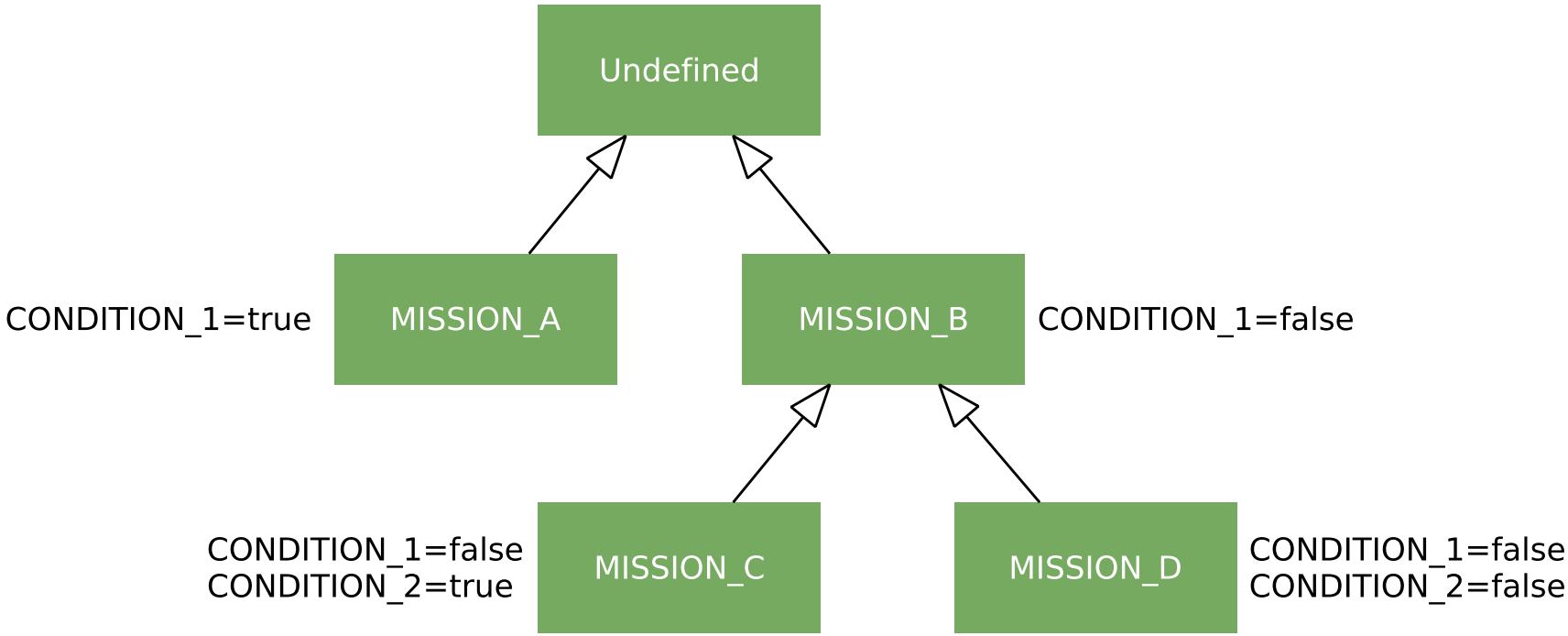


Figure – Un héritage de missions – le passage d'un mode à l'autre se fait par des conditions sur des MOOSVar, ici : `CONDITION_1` et `CONDITION_2`.

Organisation d'une mission

Configuration des modes dans pHelmIvP

La configuration de ces *modes* se fait dans l'en-tête du fichier `.bhv`

```
1   // Niveau 1
2
3     set MODE = MISSION_A {
4       CONDITION_1 = true
5     } MISSION_B  // 'else'
6
7   // Niveau 2
8
9     set MODE = MISSION_C {
10      MODE = MISSION_B // héritage, C se déroule pendant B
11      CONDITION_2 = true
12    }
13
14    set MODE = MISSION_D {
15      MODE = MISSION_B // héritage
16      CONDITION_2 = false
17    }
```

Organisation d'une mission

Passage d'un mode à un autre

Une behavior peut publier des variables MOOS lorsqu'elle prend fin.
Le paramètre endflag permet cette action :

```
1   Behavior = BHV_Quelconque
2   {
3     name      = nom_bhv
4     pwt       = 100
5     condition = MODE == MISSION_C
6
7     duration = 300 // la behavior se terminera quoiqu'il
8                    // arrive après 300s d'exécution
9
10    // une nouvelle valeur pour CONDITION_2 sera
11    // alors publiée dans la MOOSDB : 'false'
12    endflag  = CONDITION_2 = false
13
14    // cela entrainera la fin du mode MISSION_C
15    // et le début du mode MISSION_D
16  }
```

Section 7

Pour aller plus loin

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

- 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