Robotique et comportements réactifs

Véhicule de Braitenberg, Architecture de subsomption, Boïds

Mise à jour : Mars 2018

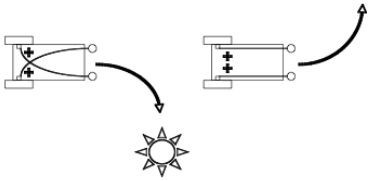
Rendez-vous sur la page http://pages.isir.upmc.fr/~bredeche/ - onglet 3i025 pour télécharger le code source Python nécessaire pour faire les exercices ci-dessous (dépendances: Pygame, Matplotlib). Dans l'archive que vous récupèrerez, seul le fichier *multirobots.py* vous intéresse. Pour toutes les questions qui suivent, vous devrez faire une copie de ce fichier avant de commencer. Il existe une aide sommaire dans les commentaires au début du fichier, et le fichier contient tout ce qu'il faut pour vous aider à utiliser les fonctions utiles à la réalisation des questions ci-dessous. Passez un peu de temps pour vous familiariser avec le fonction **stepController(.)**.

Pour l'évaluation, vous devez conserver un fichier prêt à l'emploi pour chaque question de chaque exercice.

Remarques préalables: (1) le terme "murs" désigne ici indifféremment les blocs murs et les bords de l'arène ; (2) angles des 8 senseurs de proximité p/r à l'avant du robot sont: -170°,-80°,-40°,-20°,+20°,40°,80°,+170°

Important: dans la fonction stepController(.), seul le dernier appel à la fonction setTranslationValue(value) (resp. la fonction setRotationValue(value)) est pris en compte (i.e. tout nouvel appel écrase le précédent, l'exécution de la commande de translation (resp. de rotation) est réalisée entre deux appels de la fonction stepController(.))

Véhicules de Braitenberg [Braitenberg, 1984]



L'image ci-contre montre deux exemples de véhicules de Braitenberg. Dans ces exemples, les connexions reliant les entrées sensorielles (ici: deux photorécepteurs) et les sorties motrices (ici: les deux moteurs) sont soit excitatrices, soit inhibitrices.

En s'inspirant de ces deux exemples, on vous demande de programmer quatre comportements:

(1) comportement "aller vers les murs et *ignorer* les robots", (fichier *braitenberg_loveWall.py*, à créer)

(2) comportement "éviter les murs et *ignorer* les robots", (fichier *braitenberg_hateWall.py*, à créer)

(3) comportement "aller vers les robots et *ignorer* les murs" (fichier *braitenberg_loveBot.py*, à créer)

(4) comportement "éviter les autres robots et *ignorer* les murs" (fichier *braitenberg_hateBot.py*, à créer).

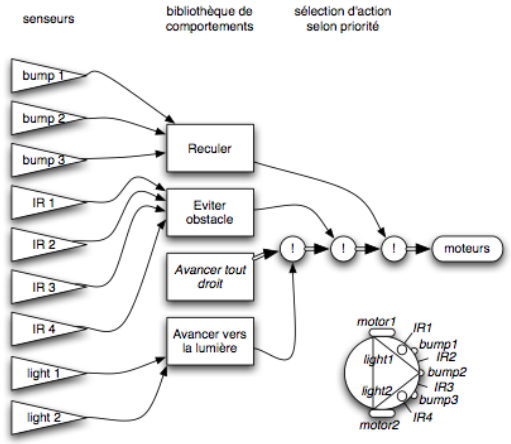
(5) comportement "éviter les obstacles (murs et robots) (fichier *braitenberg_avoider.py*, à créer).

Par défaut (i.e. en l'absence d'obstacles), les comportements commandent un déplacement en ligne droite.

Testez avec 1 seul agent, puis avec 10 agents (variable *nbAgents)* afin de valider vos comportements.

Important: le contrôleur d'un véhicule de Braitenberg utilise uniquement des connexions inhibitrices, excitatrices et des sommes. Vous devez donc déterminer les commandes motrices comme une combinaison linéaire des entrées sensorielles (i.e. ***surtout pas de if-then-else!****)*

Architecture de subsomption [Brooks, 1986]



La figure ci-dessus donne un exemple d'architecture de subsomption, composée de comportements et d'un processus de sélection d'action qui précise l'ordre de priorité des comportements en fonction de leur activation possible.

Dans l'exercice précédent, les comportements (3) et (4) ne permettent pas de gérer l'évitement d'obstacle. Pour remédier à cela, créez un fichier *subsomption.py* afin d'implémenter pour chaque agent une architecture de Subsomption disposant de trois blocs de comportement.

On considèrera que les comportements 2, 3 et 5 ne sont pas éligible en l'absence de stimulation sensorielle. Implémentez aussi un nouveau comportement "aller tout droit" qui n'utilise pas les senseurs.

Stratégie 1: solo (comportement d'exploration)
- Aller tout droit
- Eviter les obstacles (comportement no.5)

Stratégie 2: bot lover (comportement d'agrégation)
- Aller tout droit (nouveau comportement)
- Eviter les murs (comportement no.2)
- Aller vers les robots (comportement no.3)

A vous de fixer la stratégie en terme de sélection d'action (i.e. l'ordre de priorité des comportements).

Testez votre programme en utilisant 1 robot, puis 10 robots, pour chaque stratégie. Puis mettez simultanément 4 robots qui suivent la stratégie 1 et un robot qui suit la stratégie 2.

Vous pouvez commencer le projet.