Bordeaux INP – Option 3A IA – Année 2021/2022

Fiche de projet semestriel

Titre :

Comportements réactifs et apprentissage en environnement multi-agent

Description :

La librairie pyvrep-epuck[1] a été développée dans le but de permettre à des étudiants aux profils variés de s'initier au contrôle robotique et à l'interaction multi-robot. Elle permet de programmer des boucles sensorimotrices bas niveau par des fonctions Python, de les combiner pour créer des comportements plus complexes, et de lancer ces boucles de contrôle sur une population de robots mobiles dans un simulateur. La définition des comportements est réalisée selon une approche "bottom-up" (du plus simple au plus complexe), dans l'esprit de la "behavior-based robotics" proposée par Brooks[2] ; et plus précisément le cadre formel des "véhicules de Braitenberg"[3].

La libraire est interfacée avec le simulateur 3D CoppeliaSim[4] et permet de contrôler des robots mobiles simulés Epucks. Ces robots sont constitués d'une base circulaire, de deux roues, de capteurs de proximités, de capteurs au sols, ainsi que d'une caméra.

Les sessions pratiques du cours sont réalisées dans des Jupyter Notebooks[5], qui contiennent à la fois les instructions de la session ainsi que du code Python. La librairie permet de se connecter au simulateur puis de programmer des comportements de façon très intuitive et interactive (voir Fig. 1). Par exemple, exécuter une cellule du notebook contenant le code `robot.left_speed = 1` a un effet immédiat dans le simulateur : la roue gauche va se mettre en rotation et le robot va donc tourner sur lui-même.

Les objectifs principaux du projet sont d'une part d'étendre la librairie pour intégrer un simulateur 2D permettant la simulation de nombreux agents dans des environnement à la dynamique complexe ; d'autre part de permettre l'interaction avec des algorithmes d'apprentissage par renforcement pour l'optimisation des boucles de contrôle[6]. Du côté du simulateur, nous utiliserons simple-playgrounds[7] (ex-Flatland[8]), dont le développement a

---

[1] https://github.com/clement-moulin-frier/pyvrep_epuck
[2] Brooks R. A. (1991), *Intelligence without representation*, Artificial Intelligence, vol. 47, no. 1-3
[3] Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: MIT Press
[4] https://www.coppeliarobotics.com/
[5] https://jupyter.org/
[6] Nous utiliserons https://github.com/DLR-RM/stable-baselines3
[7] https://github.com/mgarciaortiz/simple-playgrounds/
[8] https://arxiv.org/abs/1809.00510

commencé à l'ENSTA-Paristech et qui se poursuit actuellement par Michael Gracia Ortiz de la City University à Londres (qui co-encadrera ce projet). Il s'agit d'un simulateur rapide en 2D à la physique réaliste, permettant d'intégrer de nombreux robots (jusqu'à une trentaine sur une machine standard) ainsi que des objets à la dynamique variée (voir Fig. 2).

Client/tuteur :
Projet proposé par Clément Moulin-Frier (Inria-Flowers) et Michael Garcia-Ortiz (City University, Londres, UK)
Contacts : clement.moulin-frier@inria.fr ; Michael.Garcia-Ortiz@city.ac.uk

Livrables :

Les objectifs du projet sont les suivants :

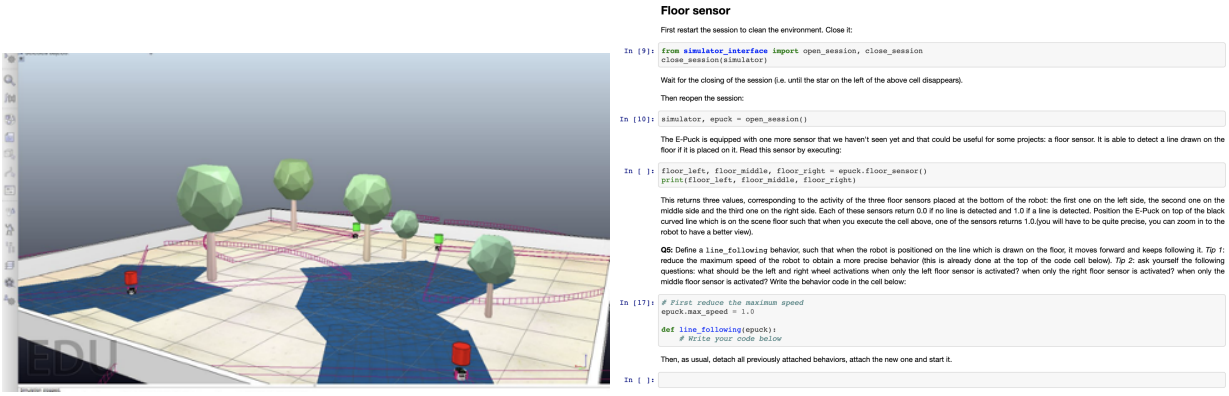
- Familiarisation avec les principes de la "behavior-based robotics" par la lecture de quelques articles scientifiques (voir les notes de bas de page ci-dessus)
- Prise en main de la librairie pyvrep-epuck et des jupyter notebooks associés
- Prise en main de l'environnement de simulation simple-playgrounds
  - Cahier des charges, contenant le détail de l'architecture logicielle proposée
- Abstraction des structures de données de la librairie pour pouvoir l'interfacer avec différents simulateurs ou robots physiques
- Intégration de simple-playgrounds dans pyvrep-epuck
- Adaptation des jupyter-notebook utilisés dans le cours (VREP -> simple-playgrounds)
- Intégration d'un algorithme d'apprentissage par renforcement

Environnement technique :
Python 3, Jupyter Notebooks, CoppeliaSim, simple-playgrounds, stable-baselines (voir liens dans les notes de bas de page)

Ressources *:*
Voir liens dans les notes de bas de page

Points particuliers :
Les pré-requis sont les suivants :
- Compétences solides en programmation orientée-objet, notamment en Python
- Lecture et écriture en anglais (les jupyter notebooks sont en anglais).
- Connaissances des principes de l'apprentissage par renforcement

- Intérêt pour les systèmes autonomes et l'interaction multi-agent, en intelligence artificielle et en biologie



Floor sensor

First restart the session to clean the environment. Close it:

```
In [9]: from simulator_interface import open_session, close_session
        close_session(simulator)
```

Wait for the closing of the session (i.e. until the star on the left of the above cell disappears).

Then reopen the session:

```
In [10]: simulator, epuck = open_session()
```

The E-Puck is equipped with one more sensor that we haven't seen yet and that could be useful for some projects: a floor sensor. It is able to detect a line drawn on the floor if it is placed on it. Read this sensor by executing:

```
In [ ]: floor_left, floor_middle, floor_right = epuck.floor_sensor()
        print(floor_left, floor_middle, floor_right)
```

This returns three values, corresponding to the activity of the three floor sensors placed at the bottom of the robot: the first one on the left side, the second one on the middle side and the third one on the right side. Each of these sensors return 0.0 if no line is detected and 1.0 if a line is detected. Position the E-Puck on top of the black curved line which is on the scene floor such that when you execute the cell above, one of the sensors returns 1.0.(you will have to be quite precise, you can zoom in to the robot to have a better view).

Q5: Define a `line_following` behavior, such that when the robot is positioned on the line which is drawn on the floor, it moves forward and keeps following it. *Tip 1*: reduce the maximum speed of the robot to obtain a more precise behavior (this is already done at the top of the code cell below). *Tip 2*: ask yourself the following questions: what should be the left and right wheel activations when only the left floor sensor is activated? when only the right floor sensor is activated? when only the middle floor sensor is activated? Write the behavior code in the cell below:

```
In [17]: # First reduce the maximum speed
         epuck.max_speed = 1.0

         def line_following(epuck):
             # Write your code below
```

Then, as usual, detach all previously attached behaviors, attach the new one and start it.

```
In [ ]:
```

Figure 1. La librairie pyvrep-epuck fournit une plateforme éducative pour aborder des problèmes de robotique, d'apprentissage machine et de simulation multi-agent de façon interactive et intuitive. Elle repose actuellement sur le couplage d'un simulateur robotique 3D (à gauche) et de Jupyter Notebook (à droite). Ces derniers permettent de réaliser des supports de travaux pratiques interactifs, sous la forme de pages web (contenant du texte, des images, des liens …) dans lesquelles les étudiants peuvent exécuter du code Python. La librairie pyvrep-epuck permet ainsi de contrôler le simulateur directement depuis ces pages web (lecture des capteurs des robots, envoi de commandes motrices, contrôle d'éléments de l'environnement). Elle permet de définir des boucles de contrôle de manière simple, sous la forme de fonctions prenant en argument les valeurs des capteurs et retournant des activations sur les moteurs. La capture d'écran du simulateur provient du projet d'un groupe d'étudiants : deux robots « prédateurs » (en rouge) implémentent des boucles de contrôle pour chasser deux robots « proies » (en vert clair) dans un environnement reproduisant un cycle de saisons été-hiver.

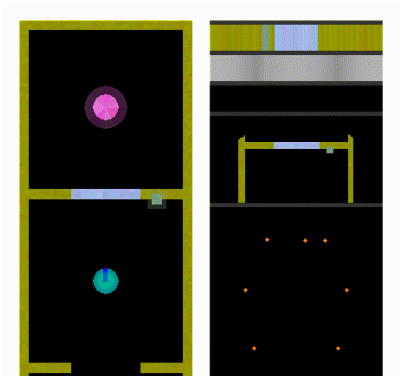


Figure 2. L'objectif du projet est d'étendre la librairie pyvrep-epuck pour pouvoir l'interfacer le simulateur simple-playground[9] et de permettre l'optimisation de boucles de contrôle par apprentissage par renforcement.

[9] Voir une version animée de cette image sur https://github.com/mgarciaortiz/simple-playgrounds/