Robotique, Projet II

David Baschung, Groupe 6

IN.2022 Robotique 2019, Cours BSc, 2e Sem.
Université de Fribourg
david.baschung@unifr.ch

Abstract

Ce rapport traite l'implémentation de deux controlleurs sur des e-pucks 2. Le premier controlleur consiste à effectuer une tâche en 3 étapes nécessitant l'utilisation de la caméra, des capteurs de sol et latéraux ainsi que la communication virtuelle entre robots. Le deuxième controlleur est un défi faisant usage des capteurs de sol pour analyser un trajet tracé par une ligne noire, afin d'en reconnaître la forme géométrique. Le but est donc d'exécuter un processus complexe à partir de connaissances acquises sur le robot. Il s'agit de la suite d'un 1er rapport détaillant les moyens de perception du robot, ceux-ci ne seront plus abordés.

Keywords: e-puck, 2, capteur, senseur, infrarouge, proximité, sol, caméra, mur, ligne, couleur

Contents

Chapter 1

Introduction

Dans le cadre du cours de robotique, nous avons mené un projet visant à découvrir de petits robots mobiles. Ce premier projet nous a permis d'acquérir quelques bases avec des e-pucks, nous enseignant les rudiments liés à l'équipement infrarouge de proximité, à une mini-caméra et à la communication virtuelle. Suite à cette introduction, nous pouvons nous intéresser à la combinaison de ces connaissances pour contrôler efficacement les robots.

2 chapitres seront abordés. Le premier traite une tâche de base imposée à 3 robots simultanément, qui doivent travailler en synergie pour la résoudre. Les robots utilisent leurs propres moyens pour détecter un bloc de couleur et une zone singulière dans laquelle ils sont placés. Ensuite ils communiquent afin de s'orienter vers la zone qui leur a été assignée préalablement. Les résultats obtenus pour révoquer ce problème sont probants, grâce à l'utilisation de la caméra et de la proximité par infrarouge. Nous avons fait usage de controlleurs de mouvements implémentés dans le premier projet, à savoir le *lover, l'explorer, le line-follower* et *le wall-follower*

Le second chapitre décrit un défi personnalisé. Nous avons tenté de reconnaître un polygone dessiné sur une table par de larges lignes noires. La travail du robot consistait à en effectuer le tour pour le distinguer parmi différentes formes prédéterminées. Il devait être capable de distinguer une silhouette d'une autre, uniquement sur la base du trajet effectué.
La difficulté consistait dans la disposition de la figure. Il fallait pouvoir la contourner dans les deux sens, admettre tout point de départ et la distinguer parmi les autres.
Grâce au *line-follower*, nous avons pu contourner la forme et analyser les mouvements du robots. Différentes implémentations de reconnaissance sont ensuite détaillées selon les individualités des formes.

Ces deux tâches nous auront permis d'appréhender la résolution de problèmes complexes mais réels. Il a fallu développer des concepts de résolution basé sur des observations antérieures, qui nous ont permis de réaliser tout le potentiel de ces automates en apparence sommaires.

Chapter 2

Tâche de base

Chapitre sur la tâche de base imposée.

2.1 Description du problème

La tâche de base utilise 3 robots de type e-puck 2. Dans un premier temps, nous les avons testés en utilisant le simulateur Webots, avant d'utiliser de vrais robots.

L'environnement physique (réel ou virtuel) est une arène rectangulaire de 80x160cm, séparée en 3 parties par des lignes noires au sol. Dans chaque partie se trouve un robot et un bloc de couleur, placés de manière aléatoire. Les blocs colorés des zones latérales, rouge et bleu, peuvent être permutés initialement, tandis que le vert reste au centre. Additionnellement, des blocs blancs sont rajoutés pour cacher le bloc rouge et le bleu du robot central, ainsi que dans les coins d'une zone latérale seule.

La tâche consiste à reconnaître la zone et son bloc coloré. Elle est divisée en 3 étapes et astreinte aux robots comme suit :

1. Les robots doivent trouver la couleur du bloc érigé dans leur zone. A ce moment, les robots doivent y rester confinés.
2. Ils détectent ensuite la zone dans laquelle ils ont été placés. (les propriétés des zones leur sont connues.)
3. Ils doivent finalement se dirigier vers leur nouvelle zone selon cette règle : Le robot de gauche (sans blocs blancs dans les coins) doit se diriger vers le bloc rouge. Le robot de droite (avec blocs) se dirigera vers le bleu.

Notons que deux règles feront foi en permanence : un robot n'est pas autorisé à franchir une ligne avant la phase 3 mais il peut la suivre. De plus, les couleurs ne seront jamais placées au bord d'une zone.

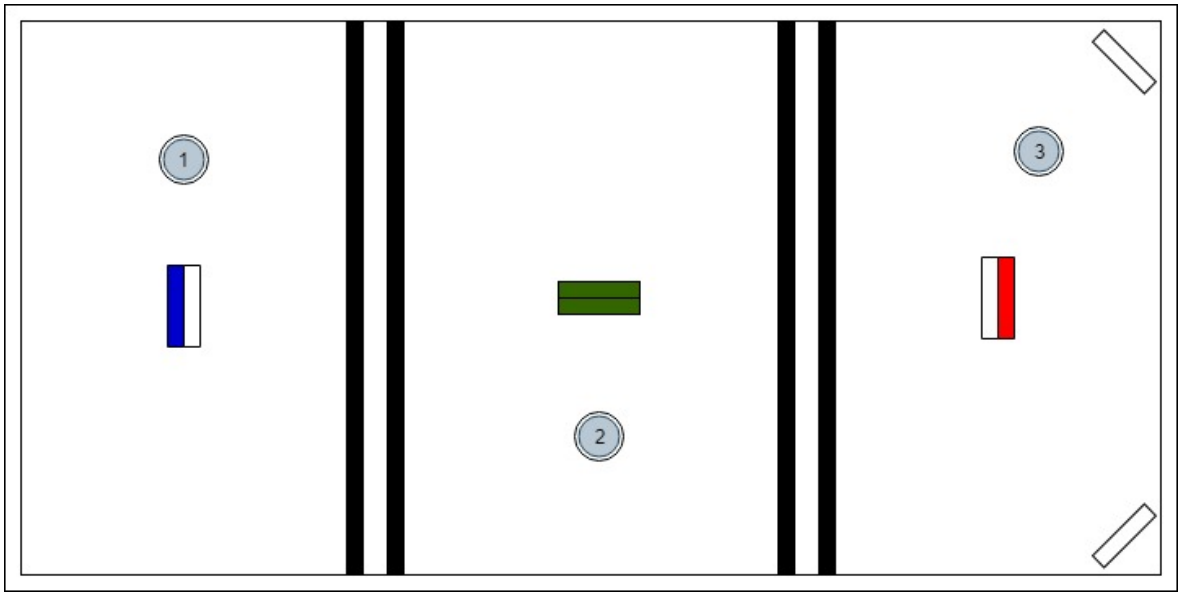

Figure 2.1: Environnement utilisé Deux lignes séparent chaque zone.

2.2 Stratégie de solution

Cette tâche nécessite de nombreux éléments de détection. Pour la résoudre, nous pouvons la décomposer en différents états propres à chaque phase.

1. Dans la première étape, l'e-puck doit pouvoir détecter sa couleur, il doit donc se déplacer, sans dépasser la limite de sa zone. Un controlleur propice à cette situation est *l'explorer*, qui, couplé d'une détection des lignes, pourra éviter les murs et parcourir sa zone aléatoirement. Cette situation se maintient jusqu'à la découverte du bloc coloré grâce à la caméra à l'avant du robot. Comme le robot doit inéluctablement s'approcher du bloc coloré, nous utiliserons le *Lover* dès que la couleur est aperçue. Notons que certaines opérations peuvent interrompre cet état, comme la détection d'une ligne en apercevant un autre bloc sur la table.

2. Dans la deuxième étape, il s'agit de déterminer la zone courante. Les zones latérales sont différenciables par leur 3 limites murales, le coté droit possédant aussi des coins arrondis. La partie centrale est flanquées de deux lignes. Il faudra donc tester les limites en effectuant le contour de la zone. Pour cela nous avons pris une ligne comme point de départ. Celle-ci doit être longée jusqu'au mur, dont nous avons le point de départ. En suivant le mur, on observe d'éventuels angles droits avec les capteurs de proximité, ainsi que la présence d'une éventuelle 2e ligne au sol. Ces 2 éléments combinés peuvent déterminer la zone d'un robot.

3. Dans la troisième étape, il "suffit" de se diriger vers la zone et d'avancer. Nous devrons toutefois prendre garde aux éventuelles collisions entre robots, c'est pourquoi nous ne longerons qu'un côté du mur à chaque fois, et ce jusqu'à avoir franchi les lignes jusqu'à la bonne zone.

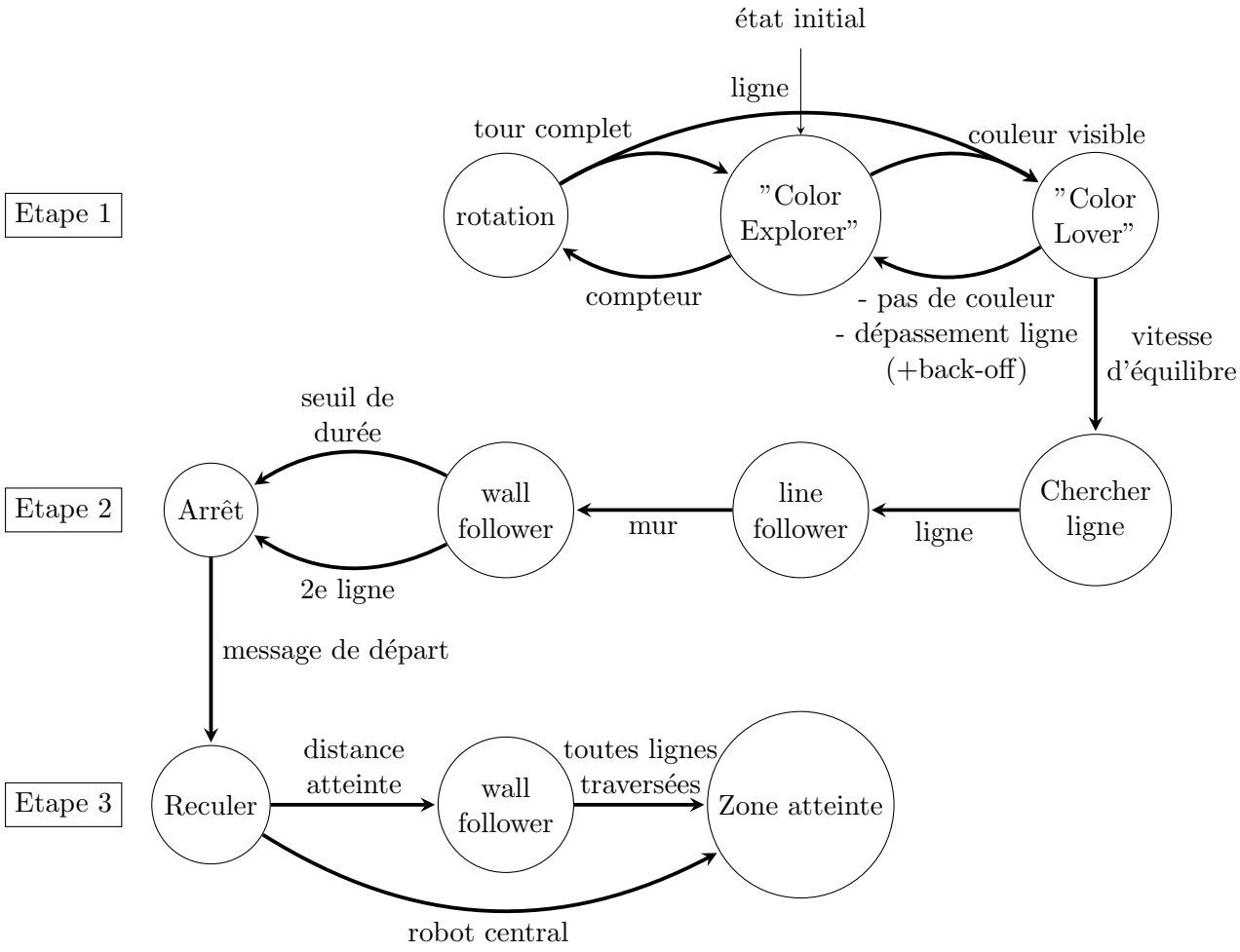

Figure 2.2: Etats utilisés pour la tâche de base

2.3 Implementation

L'implémentation de notre stratégie repose essentiellement sur des controlleurs déjà utilisiés auparavant. Quelques détails restent à préciser toutefois.
Le controlleur *Explorer* et le *Lover* de la première phase ont étés modifiés. Ceux-ci sont configurés pour détecter aussi une ligne au sol pour éviter tout dépassement. Si une ligne est rencontrée en pemière phase, le robot recule et tourne. Ensuite, un état supplémentaire a été rajouté. Celui-ci permet de scanner les environs à la recherche du bloc coloré. Il permet simplement de faire tourner le robot sur un tour grâce à un compteur.
L'étape 2 est caractérisée par une suite séquentielle d'événements. L'événement le plus important est le wall-following : si le temps est long, un compteur élimine la probabilité que l'e-puck soit au centre. Une variable supplémentaire est également observée à ce stade, il s'agit de la détection d'un mur par les capteurs avants gauche et droit simultanément. Ceci signifie qu'il y a un angle droit, appartenant à la zone gauche.
La troisième étape nécéssitera d'implémenter à nouveau le *wall-following*. Selon la position initiale et le nombre de lignes détectées, on pourra facilement déterminer si l'on atteint la zone désirée.

2.4 Validation

Les solutions testées se sont révélées efficaces pour résoudre la tâche en entier. Si le rajout d'un état de rotation était une bonne idée dans la première étape, il n'en reste pas moins que nous avions eu quelques problèmes avec la détection des couleurs par la caméra. L'algorithme utilisé pour le projet 1 et faisant abstraction de la luminosité était heureusement déjà suffisant pour une détection rapprochée de l'objet de couleur. Il aura néanmoins fallu ici garantir toutes les étapes de la tâche de base sans pouvoir observer les objets à distance.
Nous pourrions éventuellement citer la difficulté d'obtenir un scan complet de la zone dans la première phase. Parfois un peu lent, *l'explorer* doit rediriger l'automate vers le centre, pour une position du robot aléatoire. Si cette méthode est potentiellement infinie, la probabilité d'approcher l'objet en peu de temps n'est pas toujours élevée.

Chapter 3

Défi

Chapitre sur le défi choisi par notre groupe.

3.1 Description du défi

Pour ce défi, nous n'utiliserons qu'un seul e-puck. Nous les avons testés sur de vrais robots uniquement.

• Notre défi consistait à reconnaître une forme dessinée sur une table par d'épaisses lignes noires. Cette forme devait être un polygone fermé répertorié à l'avance par le programme et reconnaissable parmi plusieurs d'entre eux, soit 4 au minimum. En l'occurrence, nous avions pour mission d'implémenter un controlleur de type line-follower, qui force le robot à suivre tout le contour de la ligne jusqu'à ce que la reconnaissance soit faite. Cela signifie que certaines propriétés de la forme peuvent être reconnues, soit en étant exclusives, ou par une fonction d'historique du suivi de la ligne. De plus, la forme doit être reconnaissable dans les deux sens de lecture et à partir de n'importe quel point de départ.

3.2 Stratégie de solution

Notre stratégie de résolution s'est portée sur l'aspect séquentiel des lignes disposées bout-à-bout. Au début, le puck avance et trouve la ligne, puis il longe la ligne jusqu'au bout de manière aussi droite que possible, puis pivote à l'extrémité pour trouver la ligne suivante. Nous pouvons ainsi déterminer avec une bonne précision l'angle de jointure et éventuellement la longueur de la ligne précédente.

Nous avons choisi d'analyser quatre formes de base : un rectangle, un losange, un octogone et une croix. Nos essais se sont déroulés en deux temps : en premier lieu nous avons répertorié quatre formes de base qui puissent être reconnues selon leurs angles exacts. Cette méthode a l'avantage d'autoriser une mise à l'échelle des formes, sans perturber la détection. Nous avons ensuite mené une deuxième approche, plus fexible dans la conception, en créant un éditeur graphique qui allait nous permettre de générer la signature de la forme. Nous détaillerons ces deux approches ci-dessous.

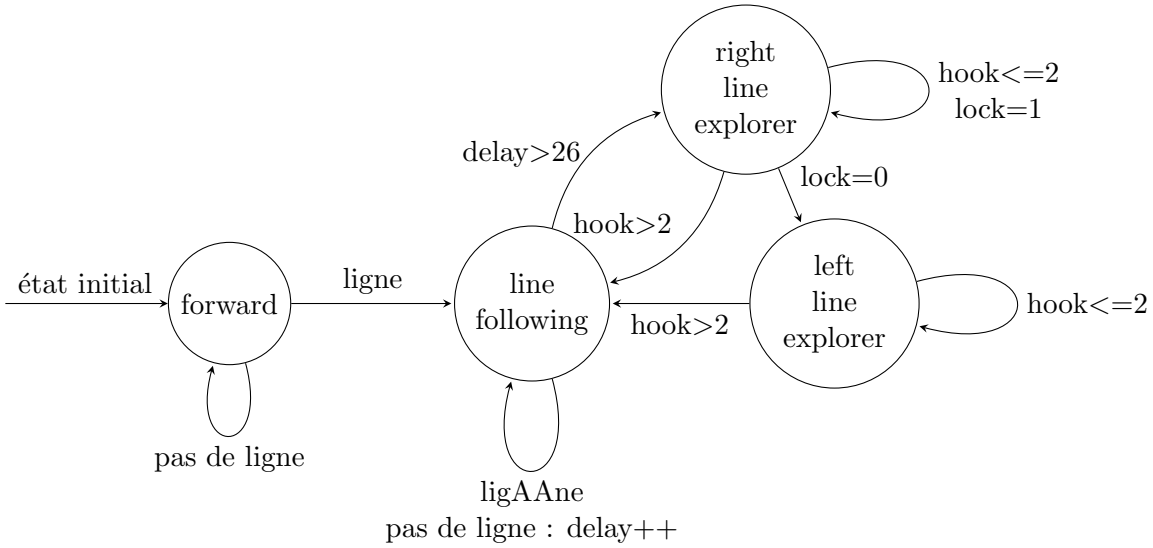

Figure 3.1: Etats du line follower pour le challenge

3.3 Implémentation

Nos implémentations utilisent un controlleur de type line-follower limité. Lorsque le robot arrive au bout d'une ligne il avance jusqu'à être centré (delay), tourne à gauche à moins de 180◦ et à droite s'il n'y a aucune ligne. Lorsqu'il en trouve une, un compteur (hook) assure qu'il sera centré sur cette ligne.

3.3.1 Détection limitée aux angles

Ceci est notre première méthode de détection, elle évalue les 4 formes pré-définies. Elle est basée sur un compteur qui se déclenche lorsque le robot arrive sur un joint entre deux lignes. Celui-ci s'incrémente en permanence tant que le robot n'a pas retrouvé une ligne. Une fois l'angle terminé, le compteur vérifie s'il est dans l'intervalle existant pour une marge particulière d'une forme. Nous pouvons par exemple citer l'octogone, qui possède un angle aigu. Lorsqu'une propriété particulière d'une forme est détectée, on incrémente un autre compteur, propre à la forme elle-même. Dès que le compteur d'une forme dépasse un seuil réglé à 14, la détection de la forme est validée.

```
1  if (timestep >= 190 && timestep <= 196) {
               rectangle++;
3          }
       if (timestep >= 56 && timestep <= 80) {
5          rectangleTwo++;
       }
```

Listing 3.1: Le rectangle possède des angles droits. La variante gauche et droite est vérifiée

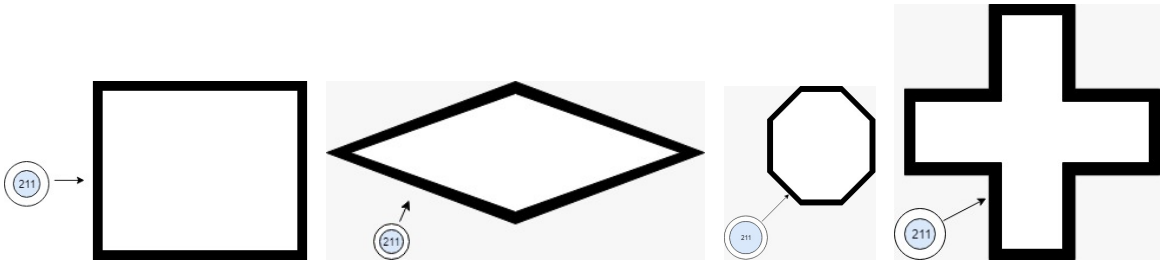

Figure 3.2: Figures utilisées dans la méthode de détection des angles

3.3.2 Détection avec événements

Signatures

Cette seconde méthode fait usage de fichiers *.csv* contenant la signature de formes. Une signature correspond à toute la séquence d'événements qui surviennent en faisant le tour complet d'une forme. Nous avons créé un petit éditeur graphique permettant de générer ces signatures en dessinant les polygones. Contrairement à la méthode des angles, elle répertorie successivement ce que le robot doit détecter. Ainsi lorsqu'il effectuera une suite d'angles et de droites, il les enregistrera et nous pourrons les comparer à la suite d'événements dans la signature. Ces événements contiennent notamment la distance des segments et l'angle entre deux d'entre eux.

angle (rad)	along block	time (s)	distance (cm/s)	left wheel (cm/s)	right wheel (cm/s)
0	1	3.882	10	2.576	2.576
1.571	0	2.165	0	-2.576	2.576
0	1	3.882	10	2.576	2.576
-1.571	0	2.165	0	2.576	-2.576
(...)	(...)	(...)	(...)	(...)	(...)

Table 3.1: extrait d'un fichier *.csv* contenant la signature d'une forme.

Nous avons choisi de mettre les angles et les droites dans la signature. Cela signifie que la forme exacte du polygone peut être garantie, mais que sa mise à l'échelle est fixe. Notons cependant que cette implémentation est nécessaire à cause du potentiel manque de précision du dessinateur, ce qui généralise l'imprécision des angles.

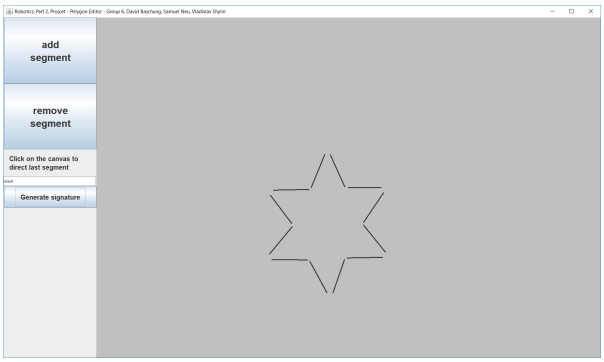

Figure 3.3: Editeur pour générer les signatures.

Détection

La détection se partitionne ainsi : à chaque fois que le robot atteint le sommet d'une ligne ou termine un virage, un événement s'enregistre. Celui-ci contient toutes les mêmes données que dans l'événement d'une signature.
Le principe de détection se fait ainsi : chaque seconde, les événements enregistrés sont parcourus pour être comparés aux signatures. Si le premier élément enregistré est compatible avec un ou plusieurs élément(s) signé(s) il s'agit d'un/de potentiel(s) point(s) de départ pour contourner la forme. Dans ce cas, on effectue un parcous secondaire : on compare ensemble les événements signés et enregistrés précédent le point, jusqu'à ce que les enregistrements ne correspondent plus. Nous le faisons également pour les événements suivants (2e sens de lecture du robot). Si les événements correspondent, on a détecté la forme.

Pour déterminer si les événements correspondent un à un en line-follow, nous contrôlons :

- l'angle, dans une marge de 20◦ par côté
- le temps écoulé
- la distance, selon temps et vitesse, droite ou en radians

3.4 Validation

Dans la détection par les angles, nous avons obtenus des résultats probants qui garantissaient une détection adéquate des formes. Les angles sont détectés correctement et avec une bonne précision. A titre d'exemple nous pouvons comparer le carré et le losange pour nous en convaincre, la marge que possède un e-puck pour tourner d'un segment à l'autre du losange est très proche de celle d'un carré.
Sie cette méthode ne garantit pas que des segments ne soient pas trop longs ou courts, se chevauchent, ou que la mise à l'échelle soit fixe, elle offre une très bonne probabilité de détection. (Nous avons effectué 20 tentatives successives sans faute).

La méthode d'analyse par fichier-signature ne nous as en revanche pas apporté autant de satisfaction. Si nous avons réussi à générer des signatures plutôt cohérentes avec les mouvements du robot et à implémenter un controlleur, il n'a pas été possible de faire de lien régulier entre une signature et une forme. D'une part, cette méthode, supposée fiable, reste très complexe et il peut subsister des erreurs ou des différences dans l'interprétation.
Un élément notoire de cette marge d'erreur a été particulièrement visible avec la charge des batteries et le changement du robot. En appliquant une certaine vitesse aux roues, les données étaient parfois faussées et produisaient des différences de vitesses allant parfois jusqu'à 148% entre différents e-pucks.

Chapter 4

Conclusion

Les résultats obtenus dans cette étude montrent qu'ils est possible d'obtenir des controlleurs complexes plutôt fiables à partir d'éléments simples. En prenant connaissances des imprécisions et des capacités des robots au préalable, on peut anticiper leur comportement. Nous avons pu constater que les différents contrôleurs de braitenberg étaient fiables dans le contexte de la tâche de base. Le challenge nous a démontré qu'il était possible d'effectuer une implémentation correcte, à moins de prévoir une certaine marge d'erreur

Bibliography

[1] Justin Zobel. *Writing for Computer Science*, 2nd edition. Springer-Verlag, London, 2004, 275 pages.

Appendix

Appendix A Experimental Results

Project description and guidelines

Robotics, BSc Course, 2nd Sem., Dr. Julien Nembrini, Manuel Mondal

Handout on Thursday Mars 21 2019 | Week 7 to 13 | Due on **Monday May 16 2019, 12h00**

Presentation on **Thursday May 23 2019**

The remaining part of the semester will be devoted to a group project which consists of a basic task and a challenge. The challenge can be freely defined by the students but must be presented to the teaching team for approval before starting work on it.

Important dates :

April 3, 23h00 Short description of challenge idea, sent per email to the teaching team

April 4 Challenge idea discussion

April 11 Final short challenge description, sent per email

May 16, 12h00 Hard deadline for submitting the final project (including presentation)

May 23, 13h15 Final presentation

The techniques exercised during the first part of the semester are sufficient in combination to solve the task proposed. All behaviors needed are variants of Explorer/Lover behaviors or PID controllers. However, it is allowed to use other techniques if wished. As during the first part of the semester, all robots should use the exact same code.

Grade The final project is evaluated on the basis of :

- the project code/algorithm (30%)
- the challenge code/algorithm (20%)
- the report (30%)
- the video (10%)
- the final presentation (10%)

Important elements include :

- dynamicity, robustness and efficiency of the algorithm
- clarity and readability of the code
- structure and clarity of the report
- orthography and grammar

Hand in :

Upload your code, report, video and presentation (pdf) on Moodle before May 16, 12h00 (noon). You may amend your presentation between the deadline and May 23, but you must submit it before 12h00 (noon) on May 23.

Zip files not following the naming convention will not be considered.

Basic task

The basic task consists of letting 3 robots (1) find the color block in their respective area of the arena, (2) detect which area they are located in based, on differences in the environment, and (3) go the their assigned area based on information communicated by others (robot in the left area → red, middle area → no movement, right area → blue).

Environment

In the basic task, the robot arena, through which the e-pucks navigate, has a height and width of 80x160 centimeters. The inside is divided in three areas by lines on the ground. Each area contains blocks, possibly with one white and the other colored (see fig 1). Apart from the fact that they are joined and that the white block hides the color from other areas, the configuration and the position of the blocks are not fixed : the algorithm should be robust to blocks positioned differently and with colors distributed differently over the areas. Note that the blocks will not be placed near the walls or the lines.

FIGURE 1 – Project arena (three areas, each one with a group of colored blocks)

In general, robots are not allowed to cross lines on the ground. Only at the end of the task are they allowed to cross them to reach their target areas. They are however allowed to follow lines.

We assume that 3 robots are in the arena and that the communication is reliable. Each robot is placed randomly in each area. It then follows the following phases (in order) :

— Phase 1 : Find the color of the block inside its area.
— Phase 2 : Detect in which area it is located.
— Phase 3 : Move to its assigned area based on color and area detected by all robots.

Phase 1, Finding color blocks

Task to achieve : the 3 robots need to find the color block in their area and wait for other to do the same.

During this phase, the robots are not allowed to cross lines on the ground.

1. Each robots efficiently roam/scan its area in search of a colored block. In order to speed up this behaviour, the robot has to use its camera to steer towards a recognized color.
2. Once the first color is found, he approaches it and stops in front of it in order to be sure that it is indeed in its area.
3. Once the color is confirmed, it signals other robtos that he completed Phase 1.
4. When all robots have completed Phase 1, Phase 2 starts.

Phase 2, Detecting area

Task to achieve : the 3 robots need to detect in which area they are located by detecting differences on the wall geometry.

1. Each robot efficiently explores its area to detect differences in the environment. It is allowed to follow lines and walls.
2. Once it has recognised the area, it notifies other robots that it has completed Phase 2 by sending the area it is in (left, right, or middle) together with the color of the block it detected to the others (red, green or blue).
3. Once each robot has detected its area, phase 3 starts.

Phase 3, Moving to the assigned area

Task to achieve : Using the information communicated by other robots, the robots have to move towards the area assigned to them after the following rule :

— left area → red area,

— middle area → no movement,

— right area → blue area.

The information communicated allows each robot to cross lines the exact number of times it has to (e.g. 4x from left to right), meaning that a robust line crossing behaviour has to be developed. Once it has reached the corresponding area, each robot must signal task completion with the appropriate LED pattern.

Challenge

The challenge consists of defining, designing and solving an additional challenging task.

The challenge can be freely defined by the groups, but must be described in a short email (5-6 sentences at most) that will be sent before **April 3 23h00**. This idea will be discussed with the teaching team on **April 4** to get approval before starting work on it. A final short description will be sent per email before **April 11, 23h00**. Possible challenges are :

- autonomous "cars"
- collectively solve a maze
- recognize objects/forms
- neural network to improve robot behaviour
- make multi-robot formations
- communicate with other media (sound, light, etc)
- **your proposal**

Each group will have to choose a different challenge. If two groups choose something similar, variants that are sufficiently different will have to be proposed. If you are hesitating between multiple ideas, feel free to include all of them in your initial mail.

Guidelines

You are free to choose which strategy you implement for solving the basic task and the challenge, but keep the following guidelines in mind :

- You are expected to discuss the problem, including coding and testing, in group of three persons. Please inform if you wish to change group before starting the project.

- **Report :** each student has to write his own report in Latex, maximum 12 pages + appendices (including this document as Appendix A). The report should present the project (basic task + challenge) as a whole. An empty template is provided on moodle
- **Code :** one code per group. We ask you to hand in a modular, readable and commented code. Your code will be thoroughly read.
- **Video :** one video per group, max 4 minutes. An uninterrupted and accelerated shot of the whole basic task process (phases 1 to 3) must be included. Any acceleration must be annotated in the video and further annotations for clarity are encouraged.
- **Presentation :** each student has to do his own presentation (max. 4 minutes) focusing on a particularly interesting aspect of the project (task or challenge), different for each student.

- The e-pucks need to collaborate and communicate. 3 e-pucks have to be used.
- Navigating the environment can be done using the lover/explorer behaviors and PID controller (and possibly others). Be careful to the robustness of other controllers you may choose to use.
- Don't use any hard coded behavior. The e-pucks should be able to complete the basic task even if their starting positions change or a different arena is used (all the robots basically know is that the arena is divided into 3 areas divided by lines, that they start each one in a part with one colored block).
- Make your solution as dynamic (independent of initial configuration), robust (catch error situations) and efficient (achieving the task in a short time) as possible.
- Document your code (events, methods, algorithms, etc.).
- Use the LED patterns described in the appendix in order to indicate to an observer what the robots are doing. If you add custom patterns, document it using the same tabular presentation as in the appendix.

Appendix : LED patterns

The following is a list of the LED patterns that must be used. For each pattern, the indexes of the LEDs are given. The LEDs have to be turned on long enough to be visible in the video.

Behaviours		LED indexes	Comment
Recognized colors :	Red	3	
	Green	2	if used in other context
	Blue	1	
General behaviours :	Explorer	0 + 1/2/3	the recognized color
	Lover	1/2/3	the recognized color
	Line following	0	
	Wall following	0,1,2/0,2,3	side dependent
Phase 1 completed	In front of red block	2,3	
	In front of green block	1,2	
	In front of blue block	2,3	
Phase 2 completed		0,1,3	
Task completed		ad-lib	choose your favorite !
Others			to be specified by the students

Appendix B Source Code

Place to list source code.

B.1 some appendix