

From motion effects to affordances : Bayesian learning of high-level actions

Stanislas LEROY

Master 2 parcours Intelligence Artificielle

2016-2017

Encadrants : Stéphane DONCIEUX et Alexandre CONINX

Institut des Systèmes Intelligents et de Robotique, France

Université Claude Bernard Lyon 1, France

Résumé L'apprentissage des affordances est une étape critique dans l'approche de la robotique développementale. Les données requises pour un tel apprentissage peuvent être fournies par une approche de babillage consistant à appliquer des actions aléatoires et à observer les effets résultants, mais cette approche soulève des défis spécifiques. Définir à priori un ensemble limité d'actions possibles et leurs effets correspondants constitue une limitation forte de ce que peut réaliser un robot. Au contraire, n'appliquer que des actions aléatoires a pour résultat un vaste ensemble d'effets possibles qui peuvent être difficiles à exploiter, en particulier si les affordances sont représentées par des structures discrètes tels que des Réseaux Bayésiens. Durant notre stage, une méthode de clustering est proposée pour construire automatiquement un ensemble limité d'effets sur la base de données générées par une méthode de babillage reposant sur des actions aléatoires et une tâche donnée.

Mots-clés : Robotique développementale, affordance, clustering d'effets

Abstract Learning affordances is a critical step in a developmental robotics approach. The data required for such learning can be provided by a babbling approach consisting in applying random action and observing the corresponding effects, but this approach raises specific challenges. Defining a priori a limited set of possible actions and corresponding effects is a strong limitation to what the robot can achieve. On the contrary, applying completely random actions results in a large set of possible effects that may be hard to exploit, in particular if the affordance is represented by discrete structures such as Bayesian networks. In our internship, a clustering method is proposed to automatically build a limited set of effects on the basis of data generated by a babbling method relying on random actions and on a given task.

Keywords : Developmental Robotics, Affordance, Effect clusterization

Remerciements

MON stage de 6 mois au sein de l'équipe AMAC à l'ISIR a été une formidable expérience. Je voudrai remercier Stéphane Doncieux, Alexandre Coninx et Carlos Maestre pour leurs nombreux conseils avisés et pour les nombreux moments passés à échanger tout au long de mon stage. En effet, bénéficier de leurs expérience et savoirs a été un véritable atout pour mener à bien ce stage, notamment pour surmonter les nombreux défis auxquels j'ai été confrontés.

L'ISIR est un excellent laboratoire de robotique en France et avoir la possibilité d'y faire un stage a été une extraordinaire opportunité, notamment parce que j'ai pu travailler sur un domaine passionnant et relativement jeune : la robotique développementale. Par ailleurs, cette immersion dans un laboratoire de recherche m'a donné l'opportunité de découvrir davantage le monde de la recherche.

D'autre part, reprendre des études en septembre 2016 après avoir travaillé plusieurs années a constitué un véritable défi pour moi. En effet, cela supposait de quitter la ville où je vivais, mes amis et mon confort matériel ainsi que de revenir sur les bancs de l'université. Avec le recul, cette année écoulée m'aura conforté dans ma décision de reprendre des études dans ce domaine.

1 Introduction

Artificial intelligence is the science of making machines do things that would require intelligence if done by men. Marvin Minsky

DANS le cadre de mon stage, je souhaitais découvrir davantage le domaine de la robotique développementale découvert durant mon semestre de cours à l'Université de Lyon 1. Mes recherches de stage ainsi que les conseils avisés de mes professeurs m'ont permis de concentrer celles-ci sur un nombre restreint de laboratoires, dont l'ISIR.

La robotique moderne est apparue au début du siècle dernier et englobe la robotique industrielle, médicale, militaire et domestique. Si l'on s'y rapporte, la robotique développementale constitue un domaine de recherche émergent dans la mesure où son émergence date d'une quinzaine d'années. La robotique développementale est issue principalement de la robotique mais tire également ses racines des neurosciences cognitives et de la psychologie cognitive.

L'environnement et les objets que côtoient les êtres humains ont des formes et des apparences très différentes. Les robots, par nature, sont voués à être déployés dans le même environnement que le nôtre. Les êtres humains sont capables de reconnaître des objets, pour certains inconnus, et de les classer dans des catégories contenant des objets connus, cela sans effort et très rapidement. D'autre part, l'être humain apprend à s'en servir sans qu'il soit besoin, généralement, de lui expliquer la marche à suivre.

Pour des raisons de coût (humain, matériel et en temps), il n'est pas possible de doter les robots d'une connaissance exhaustive de leur environnement et des différents objets qui le composent. En effet, l'environnement étant en perpétuelle évolution, cette exhaustivité nécessiterait une mise à jour à intervalle très régulier des connaissances. Ceci est incompatible avec le nombre très important de robots en usage dans les prochaines années et la diversité des environnements associés. En outre, il est également impossible de prévoir tous les mouvements et actions nécessaires pour manipuler les objets qu'un robot peut être amené à rencontrer.

Puisque qu'il n'est pas possible de concevoir des robots ayant une connaissance complète de leur environnement, la robotique développementale s'inspire des recherches menées en neurosciences pour doter les robots de leurs propres capacités cognitives sous la forme de modèles mathématiques. L'objectif de celles-ci est de leur permettre d'apprendre de façon autonome à agir et interagir avec leur environnement en partant d'une connaissance très limitée à priori de cet environnement. Ces nouvelles aptitudes cognitives doivent permettre de créer des robots dépassant les contraintes liées à la programmation des connaissances. Par ailleurs, les robots à venir devront pouvoir s'adapter aux changements et situations inconnues dans leur environnement. Pour cela, il est nécessaire que ces robots soient capables de construire d'autres connaissances à partir de celles précédemment acquises.

L'apprentissage par un robot des actions réalisables se fait au travers des observations et interactions avec l'environnement. Les interactions avec différents objets et les observations sont importantes pour apprendre ce que l'on nomme les *affordances*. D'autre part, connaître l'ensemble des actions possibles dans son environnement permet à un robot de pouvoir prédire les conséquences de ses actions. Cela lui permet également de planifier des actions pour atteindre un objectif défini.

La problématique de mon stage s'intéressait à réaliser une discréétisation adaptative de l'espace d'effet des actions. Dans le système d'apprentissage des effets des mouvements actuellement conçu et utilisé, la discréétisation de l'environnement et des propriétés des objets est donnée par l'expérimentateur. Afin de s'adapter à différents jeux de données d'apprentissage correspondant à des objets, environnements et processus d'exploration divers, cette discréétisation doit être apprise automatiquement par le système dans un sens qui permette que les propriétés pertinentes de chaque objet et de l'environnement soient représentées de façon adéquate pour le système d'apprentissage.

Ce rapport de stage est divisé de la façon suivante : tout d'abord, nous présentons le laboratoire et ses domaines de recherche, puis nous abordons le projet DREAM dans le cadre duquel notre stage a été réalisé. Une section est ensuite consacrée à présenter les différents sujets relatifs à la robotique développementale. Le cœur de notre travail est présenté dans les sections 4 et 5 avec l'approche théorique et l'expérimentation de validation associée. Enfin, nous abordons les perspectives et travaux futurs dans la section suivante avant de conclure dans la dernière partie.

2 Le laboratoire

L'ISIR¹ (Institut des Systèmes Intelligents et de Robotique) est un laboratoire de recherche multi-disciplinaire créé en 2007. C'est une Unité Mixte de Recherche entre l'UPMC et le CNRS, localisée sur le campus de Jussieu. Il compte environ 130 personnes et rassemble des chercheurs de différents domaines : Sciences de l'Ingénieur et de l'Information ainsi que des Sciences du Vivant (sciences du mouvement, neurosciences et sciences cognitives) incluant également des personnels hospitalo-universitaires. Par ailleurs, l'ISIR est membre du réseau national Robotex² qui regroupe des plates-formes expérimentales de robotique en France et ses équipes sont actives au sein du GDR Robotique.

Les travaux de recherche à l'ISIR sont principalement centrés sur la modélisation et l'analyse des systèmes dynamiques artificiels et naturels, la conception optimale de systèmes robotiques interactifs, la commande des systèmes interactifs, la conception et le traitement du signal de systèmes perceptifs multimodaux, la modélisation des interactions Homme-Système, les modèles neuro-computationnels, l'apprentissage artificiel ainsi que l'adaptation bio-inspirée des systèmes et de leur commande.

Ses activités sont situées au cœur d'un certain nombre d'enjeux relatifs aux systèmes robotiques autonomes et/ou interactifs concernant notamment le comportement dynamique robuste et sûr des systèmes, la perception artificielle, les interactions multimodales, l'apprentissage et l'adaptation. Elles visent en particulier à développer de nouvelles approches en matière de conception et de commande des systèmes robotiques fertilisées par un dialogue permanent entre les sciences de l'ingénieur et de l'information, la biologie et les sciences cognitives.

Dans cette perspective, le laboratoire est constitué en 4 équipes :

- AGATHE (Assistance aux Gestes et Applications THERapeutiques) : conception de systèmes robotiques synergétiques en particulier pour le guidage des gestes dans des applications thérapeutiques ;
- AMAC (Architectures et Modèles pour l'Adaptation et la Cognition) : apprentissage artificiel et adaptation du comportement à l'environnement ;
- INTERACTION : processus d'interaction avec des mondes physiques et virtuels où les agents en interaction peuvent être des personnes, des machines, des robots, des avatars virtuels ou des dispositifs physiques ;
- SYROCCO (SYstèmes RObotiques COMplexes) : commande des systèmes robotiques dynamiques adaptatifs et leur conception par notamment des méthodes d'optimisation et de synthèse de fonctions bio-inspirées.

Nous avons réalisé notre stage de fin d'études au sein de l'équipe AMAC³ dirigée par Stéphane Doncieux. L'équipe est composée de 13 membres permanents et 10 membres non-permanents. Cette équipe est multidisciplinaire, entre neurosciences computationnelles et robotique/informatique. Les travaux réalisés au sein de cette équipe visent à atteindre deux objectifs. Le premier est de comprendre le vivant au travers d'une démarche de modélisation mathématique et informatique des fonctions cognitives et motrices. Le second est de synthétiser des architectures de contrôle robotiques pour doter les robots de capacités cognitives et motrices intégrant la décision et l'apprentissage.

L'équipe AMAC comporte 2 axes principaux, l'axe *Neurosciences computationnelles* dont la finalité est la compréhension du vivant et qui vise à étudier et à modéliser sous forme mathématique et informatique les fonctions cognitives. Un second axe *Sciences de l'ingénierie* vise à développer des méthodes et des algorithmes pour doter les robots de capacités cognitives et motrices augmentant leur efficacité. Son objectif est la conception de mécanismes d'adaptation de systèmes robotiques complexes.

L'équipe s'intéresse à l'élaboration de modèles des fonctions perceptives, cognitives et motrices ainsi qu'à la synthèse d'architectures de contrôle pour les robots dans une perspective intégrative.

1. <http://www.isir.upmc.fr>
2. <http://equipex-robotex.fr>
3. http://www.isir.upmc.fr/telechargements/Rapport_activite_18_10_2012.pdf

3 Le projet DREAM

3.1 Présentation générale

NOTRE stage a été financé financé par le Labex SMART⁴ et s'insère scientifiquement dans le cadre du projet DREAM⁵. TODO SMART. DREAM (Deferred Restructuring of Experience in Autonomous Machines), soit *Restructuration différée de l'expérience dans les machines autonomes*, est un projet financé par le programme de recherche et d'innovation H2020 de l'Union Européenne. Ce projet a pour objectif d'incorporer un processus de sommeil au sein d'une architecture cognitive implanté dans un robot, dans le but de consolider son expérience acquise et donc d'améliorer sa capacité à apprendre et à s'adapter.

La robotique change de paradigme : partant d'une situation où les conditions sont contrôlées et où tout est connu à l'avance (c'est-à dire qu'il est possible de programmer explicitement le robot à résoudre une tâche), la robotique évolue vers des environnements ouverts où les concepts peuvent changer au cours du temps et où de nouveaux concepts peuvent apparaître. Faire apprendre un modèle de l'environnement à un robot nécessite une très grande quantité de données représentant les propriétés physiques des objets à son voisinage. Comme expliqué en introduction, cela n'est pas possible pour des raisons de coût et cela incite à concevoir des robots autonomes et adaptables.

L'autonomie et l'adaptabilité des robots vont de pair avec des jeux de données gigantesques qu'il faut pouvoir traiter au mieux. Cette accumulation de connaissances nécessite des processus de consolidation afin d'éviter d'être submergé par les informations. Chez l'être humain, cette consolidation est réalisée durant le sommeil. Celui-ci est crucial pour restructurer les représentations, maintenir l'intégration et la cohérence de la connaissance, ainsi que pour améliorer l'apprentissage et la formation d'abstraction. D'autre part, le sommeil consolide les souvenirs récents et, de manière concomitante, pourrait permettre de comprendre en modifiant leur structure de représentation [1].

Le projet REAM s'inspire donc du fonctionnement du sommeil chez l'animal pour faire mieux apprendre les robots avec une alternance entre des phases actives d'interaction et de collecte de données, et des phases passives de restructuration, réinterprétation et modélisation des connaissances.

La motivation principale de ce projet vise à concevoir un système développemental capable de démarrer depuis une représentation agnostique puis de bifurquer, de façon progressive et autonome, vers des représentations plus adaptées.

L'objectif de DREAM est de permettre aux robots d'acquérir une compréhension ouverte de leur environnement au cours de longues périodes de temps. Cet objectif principal peut être décomposé en 4 sous-objectifs : ① faire évoluer et décomposer de nouvelles valeurs et motivations, ② restructurer les représentations et modèles, ③ consolider la connaissance et ④ étendre la connaissance à travers les interactions sociales.

Les mécanismes à développer pour le système sont constitués d'une architecture cognitive reposant sur des algorithmes évolutionnaires. En terme d'informations, celles qui satisfont aux motivations lors de l'exploration de comportements possibles, sont considérées comme pertinentes. Cette information pertinente nécessite d'être stockée et utilisée afin de permettre au robot d'être plus efficient le lendemain. Cela est rendu possible en construisant des modèles et en redéfinissant leur représentation pour les rendre plus compacte et plus précis durant les *rêves* des robots.

Pour cela, le point de départ du projet repose sur :

- Un système agnostique aux tâches où il n'y a pas de représentation spécifique imposée ;
- Des flux sensorimoteurs bruts ;
- Un système motivationnel basique.

Ce projet implique 5 partenaires académiques européens ayant chacun des champs de compétences différents :

- UMPC/CNRS (coordinateur) : exploration de compétences et neurosciences ;
- Armines/ENSTA Paristech : compétences motrices et représentations sensorielles ;

4. <http://www.smart-labex.fr/>

5. <http://www.robotsthatdream.eu>

- GII - Universidade da Coruña : architecture cognitive basée sur des algorithmes évolutionnaires, motivations extrinsèques ;
- Queen Mary University of London : Apprentissage automatique et neurosciences computationnelles ;
- VU Amsterdam : partage de la connaissance entre robots.

Les différents partenaires disposent de 5 modèles de robots pour mener à bien ce projet : le robot Meka⁶, PR2 de Willow Garage, plusieurs Thymio-2⁷, le bras robotique CrustCrawler⁸ ainsi que le robot Baxter de Rethink Robotics. La majorité des robots est présente au sein de l'ISIR.

À l'issue de ce projet, plusieurs résultats sont espérés. Dans le domaine de la robotique, de nouveaux frameworks pour le bootstrapping cognitif sont attendus. Ceux-ci pourront se baser sur des algorithmes évolutionnaires et pourront modéliser l'alternance entre des phases actives et passives (jour/nuit). Pour cela, des connaissances approfondies en psychologie et neurosciences sont requises. Du point de vue des neurosciences, sont attendus de nouveaux modèles pour la restructuration de connaissance et le processus exploratoire (*bootstrapping*) cognitif.

Par ailleurs, au terme de ce projet, les travaux de recherche des différents membres ont vocation à s'imbriquer les uns avec les autres dans le but de fournir une architecture complète, allant de la détection et le suivi d'objet, à l'apprentissage d'affordances en passant par l'apprentissage de lancers de balle.

Le projet DREAM est réparti au sein de 8 Work Packages. La figure 1 représente l'architecture des différents Work Packages au sein du projet DREAM.

Plusieurs expériences sont prévues afin de valider les travaux effectués : une expérience de manipulation d'objets par un bras robotique et une de collaboration entre robots mobiles.

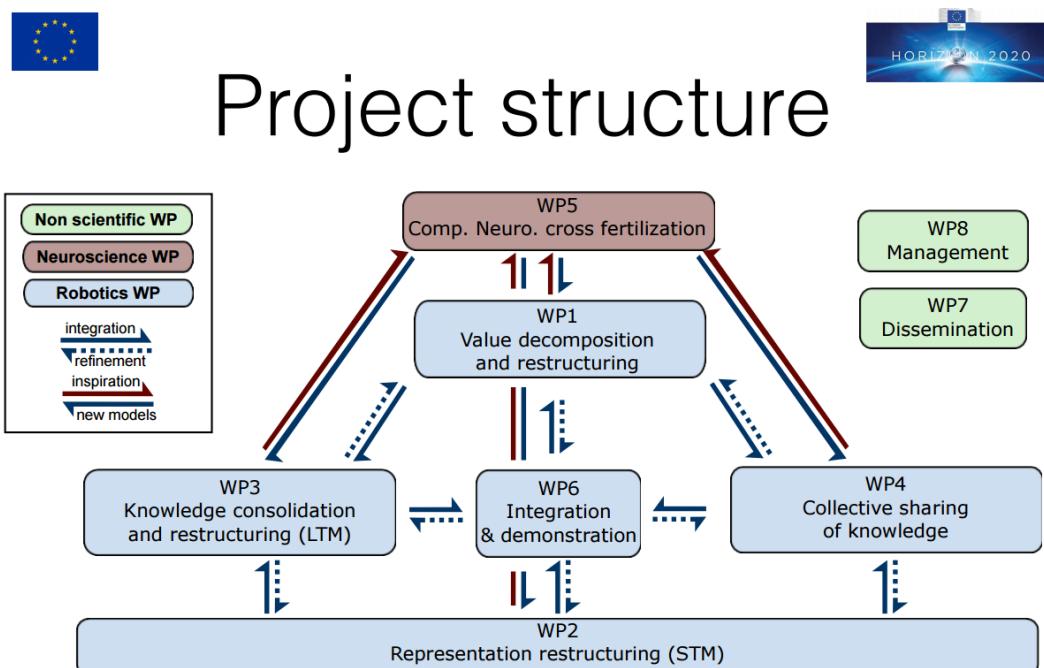


FIGURE 1 : Architecture du projet Dream

6. http://en.wikipedia.org/wiki/Meka_Robotics

7. <http://www.thymio.org/>

8. <http://www.crustcrawler.com>

3.2 Travaux de l'équipe AMAC au sein du projet DREAM

Une partie des membres de l'équipe AMAC travaille sur le projet DREAM (4 chercheurs, 2 post-doctorants, 3 doctorants et plusieurs stagiaires) et abordent différents thèmes.

Un premier thème concerne la manipulation d'objets. Plusieurs personnes travaillent sur la perception interactive et le suivi d'objets. L'idée principale de ce sujet de recherche est que pour segmenter des éléments (en l'occurrence des objets manipulables) dans une scène quelconque, il n'est pas possible de se baser sur un à priori. Pour compenser cela, le robot va interagir avec l'environnement afin d'acquérir de l'information. Une première étape consiste à segmenter l'espace perceptif au moment de l'interaction du robot sur les objets, tandis qu'une autre étape vise à segmenter chaque objet. Ce babillage est orienté objet, c'est-à-dire qu'il consiste à récupérer des données à propos de chaque objet pour produire une segmentation d'objets.

D'autres travaux se concentrent sur le lancer de balle dans un seau, effectué par un robot. Pour cela, des algorithmes de *Quality Diversity Search* sont employés. Ces algorithmes permettent des mouvements de lancer plus riches, plus divers et de plus haute qualité en minimisant l'énergie nécessaire au lancer.

La modélisation de l'apprentissage biologique constitue un second thème des travaux de l'équipe AMAC dans DREAM. Un autre sujet de recherche se focalise notamment sur l'étude du chant chez le diamant mandarin. Cet oiseau a la particularité de pouvoir apprendre le chant de son tuteur. Il est par ailleurs communément employé en tant que modèle de l'acquisition de la parole. Plus particulièrement, ce sujet traite de l'impact du sommeil sur l'apprentissage du chant. Les résultats préliminaires ont montré que l'on pouvait observer une baisse de performance durant le sommeil, à court terme donc, mais qu'il y avait en revanche une augmentation de la performance en toute fin d'apprentissage.

Enfin, un troisième thème aborde l'apprentissage par renforcement. Les travaux afférents sont effectuées sur les replays (forward et backward) et se basent sur l'algorithme Dyna-Q et sur l'apprentissage par renforcement (notamment le Q-learning) pour résoudre des tâches de navigation.

4 État de l'art

4.1 Affordances

L'apprentissage des affordances est une étape cruciale dans l'approche de la robotique développementale.

Le terme a été introduit en 1966 dans le domaine de la psychologie par J.J. Gibson comme *the affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill*. [2]. Gibson définit les affordances comme les possibilités d'interaction offertes à l'animal par son environnement. Cette définition établit une relation forte entre un objet, une action et l'effet de cette action sur l'objet dans un environnement donné (voir figure 2). En d'autres termes, une affordance est la connaissance qu'une action donnée sur un objet donné produit un certain effet. L'idée principale derrière ce terme est d'être capable de découvrir les différentes actions qui peuvent être réalisées sur un objet, en considérant les capacités disponibles et l'environnement dans lequel s'effectue ces actions. Les trois composants d'une affordance sont la perception de l'objet, l'action réalisée par l'agent et l'effet résultant de cette action sur l'objet [3].

Ce concept a influencé toutes sortes de domaines, de la robotique à l'interaction Homme-Machine en passant par le design. Les chercheurs se sont inspirés de ce concept d'affordances pour le transposer dans le domaine de la robotique afin d'améliorer les capacités perceptives des robots [4]. En effet, les affordances sont importantes en robotique dans la mesure où elles donnent une représentation des propriétés des objets, nécessaires pour acquérir une connaissance des actions réalisables [5].

Dans [6], Fitzpatrick et al. proposent une méthode permettant à un robot de découvrir et d'apprendre des affordances. Celle-ci consiste à permettre au robot d'effectuer des actions sur un objet et d'en observer le résultat.

Les affordances permettent de découvrir les différentes actions qui peuvent être réalisées sur un objet en fonction des capacités de l'acteur. Par exemple, une bouteille et une tasse avec une anse peuvent toutes les deux être saisies, mais

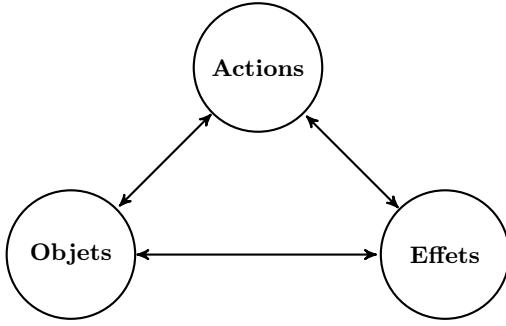


FIGURE 2 : Affordances : relation entre **Objets**, **Actions** et **Effets**.

cette action nécessite différentes compétences sensorimotrices pour être réalisée, en fonction des caractéristiques de l'objet. Autre exemple, pour un robot avec des pinces, une boîte offre les affordances *déplaçable* et *levable*, parmi d'autres.

Deux approches coexistent pour détecter les affordances. La première est une méthode analytique qui repose sur l'analyse du modèle géométrique de l'objet considéré. La seconde est une approche basée sur le traitement des données : les caractéristiques de l'objet considéré sont comparées à celles d'objets dont on connaît les affordances. La détection d'affordances est un préalable à l'apprentissage d'affordances [4] ainsi qu'à la prédiction d'effets d'une action réalisée.

Dans la majorité des travaux mentionnés dans la littérature scientifique, les affordances d'un objet sont apprises par un robot disposant d'un bras suivant trois étapes : exploration de l'environnement qui aboutit à la génération d'un jeu de données, apprentissage des affordances à partir du jeux de données et basé sur différentes techniques, et enfin, la validation de l'apprentissage en faisant exécuter une série de tâches au robot.

Par ailleurs, les affordances peuvent également être utilisées pour identifier ou reproduire des actions, tel qu'expliqué dans [7]. Les affordances fournissent des informations à propos des conséquences d'une action. En les identifiant et en les enregistrant, elles peuvent être réutilisées dans un ensemble de tâches devant être apprises et réalisées par le robot.

Les affordances peuvent être utilisées pour prédire les effets d'une action ou pour planifier les actions à effectuer. Elles permettent également la reconnaissance d'actions et l'imitation d'actions. En planification, les affordances sont capitales pour réaliser des actions. Grâce à cette connaissance des affordances, le robot peut choisir la prochaine action à effectuer sur un objet afin d'accomplir une tâche.

Toutefois, il est nécessaire de spécifier au préalable un certain nombre d'actions prédéfinies élémentaires afin d'être capable de découvrir des affordances [5]. Par exemple, cela peut concerner des actions du type *Pousser* ou *Soulever*.

4.2 Boostapping

Meltzoff et al. expliquent dans [8] que les enfants ne savent pas à priori quels mouvements musculaires permettent d'atteindre un état particulier. Les auteurs appellent cela le babillage corporel, directement inspiré du babillage vocal lorsque les enfants expérimentent des sons vocaux. Ils expliquent que les enfants font des mouvements répétés, comme dans un jeu. Finalement, après un certain temps et de répétitions, les enfants apprennent la relation entre les mouvements qu'ils effectuent et leurs effets. Ce concept a été réutilisé par la suite dans le domaine de la robotique développementale. En effet, pour un robot, apprendre des affordances nécessite d'interagir avec l'environnement à travers une phase d'exploration, ce qui correspond au babillage. Différentes heuristiques existent pour effectuer cette exploration.

Babillage sensorimoteur Le babillage sensorimoteur constitue une première heuristique. Il existe de multiples façons de réaliser ce babillage. Une première approche, qualifiée de naïve, consiste à utiliser des mouvements

aléatoires. Cette approche est la plus simple à implémenter mais elle présente un défaut majeur : elle peut générer un nombre important de mouvements qui ne produisent pas de contact avec les objets.

Une seconde approche est présentée par Maestre et al. dans [9] où ils décrivent l'utilisation de l'algorithme évolutionnaire *Novelty Search*. Cet algorithme permet de générer des trajectoires permettant de maximiser l'exploration de l'environnement. Les résultats de cet article montrent que les trajectoires permettant de toucher les objets sont de plus en plus privilégiées au fur et à mesure des générations.

Motivations - TODO Une seconde heuristique réside dans les motivations intrinsèques. Ce concept a été décrit par Ryan et Deci dans [10], puis formalisé plus tard en modèles computationnels par Oudeyer et Kaplan dans [11]. Les motivations intrinsèques permettent d'explorer progressivement l'environnement en favorisant une spécialisation progressive. Cette heuristique est utilisé en robotique et spécifiquement en robotique développementale.

Ryan et Deci définissent les motivations intrinsèques comme des activités amusantes ou comprenant du défi. D'autre part, ils définissent les motivations extrinsèques par des activités réalisées pour atteindre un résultat.

Pour Oudeyer et al., les notions de « amusant », « défi » ou de « nouveauté » n'ont pas de définitions claires et explicites dans le domaine de l'informatique. Ces définitions sont vagues et peuvent être interprétées de différentes façons. Par ailleurs, ils considèrent que le concept de résultat est ambigu. Oudeyer et al. rapprochent la motivation intrinsèque de la curiosité qui est le besoin naturel de construire des modèles de notre environnement. Dans cette optique, le concept de curiosité est davantage favorisé au profit de la recherche de la performance.

Selon ces mêmes auteurs, les motivations intrinsèques et extrinsèques ne sont pas exclusives : une même tâche peut concilier ces deux types de motivations. Ils définissent par ailleurs deux modèles de motivations intrinsèques. Le premier est basé sur les connaissances. Il vise à mesurer les dissonances entre la situation expérimentée par le robot et les connaissances et attente que le robot a de ces situations. Le deuxième est basé sur les compétences. Il mesure les compétences qu'un agent a pour réaliser des résultats ou objectifs fixé par lui-même.

ils considèrent les notions comme ambiguës, donc ils proposent leurs modèles)

4.3 Réseaux bayésiens

Une fois que le babillage sensorimoteur a été réalisé, les relations entre objets, actions et effets nécessitent d'être modélisées. Les réseaux bayésiens constituent une méthode efficace pour réaliser cela. Les réseaux bayésiens sont des modèles graphiques probabilistes représentés sous la forme de graphe dirigés et acycliques. Les noeuds représentent les variables aléatoires et les liens représentent l'influence entre les variables. Leur intérêt réside dans la possibilité offerte de combiner les connaissances des experts, contenues dans le graphe, et l'expérience, contenue dans les données.

Dans [5], Montesano et al. présentent un modèle de réseau bayésien dans lequel les dépendances entre objets, actions et effets sont modélisées. Une des motivations des auteurs réside dans le fait de pouvoir permettre à un robot d'apprendre des actions par imitation. Ce modèle de réseau bayésien permet donc d'apprendre des affordances concernant des objets mais il permet également de prédire des actions en fonction des effets obtenus. En effet, ce type de modèle est capable de réaliser des inférences et sa représentation des affordances comme dépendances probabilistes permet d'analyser et de comprendre les résultats d'apprentissage.

La planification de tâches est une autre utilisation possible des réseaux bayésiens modélisant des affordances. Gonçalves et al. présentent un autre framework utilisant les réseaux bayésiens [12] pour planifier une tâche en fonction des effets attendus. Le robot considéré dans l'article dispose de différents objets utilisables en tant qu'outils et différentes actions dans son répertoire. Il peut raisonner à partir de ceux-ci pour atteindre les effets désirés sur un objet grâce à des objets devant lui dont certains lui sont peut-être inconnus. Par exemple, à partir d'un ensemble d'objets disponibles de type *outil* ainsi que d'un effet désiré (amener plus près) sur un objet, un robot disposant d'un réseau bayésien de ce type peut raisonner pour trouver quel outil et quelle action disponibles sont les plus efficaces pour réaliser l'effet désiré.

4.4 Réseaux de neurones

Les réseaux de neurones constituent une autre approche pour détecter les affordances. Nguyen et al. proposent dans [13] un nouveau framework basé sur l'utilisation de réseaux de neurones convolutionnels profonds pour détecter des affordances. Une forme 3D englobant les objets est générée et permet à ce réseau de neurones d'apprendre ses caractéristiques de profondeur de ces formes englobantes. L'expérimentation décrite dans cet article est réalisée sur des images d'images du monde réel contenant des objets usuels. Cette méthode est initialement grossière mais devient de plus en plus fine au cours du temps.

4.5 Utilisation d'outils

Au-delà des affordances, des recherches prometteuses se concentrent sur la manipulation d'outils. Dans [5], Montesano et al. proposent une approche pour apprendre les affordances visuelles d'objets ou d'outils. Dans cet article, un robot apprend les caractéristiques visuelles d'objets et d'outils à l'aide de descripteurs visuels. De cette façon, ils montrent que le robot est capable de réaliser une action spécifique ou d'obtenir un résultat désiré. L'utilisation des descripteurs visuels permet au robot de généraliser avec des objets inconnus.

D'autre part, Gonçalves et al. [12] propose un modèle probabiliste capable d'apprendre les interactions mutuelles entre objets dans des tâches complexes qui impliquent une manipulation. Dans ces tâches, l'un des objets devient un outil une fois saisi et utilisé, tandis que l'autre objet est celui sur lequel l'outil agit. Avec ce modèle, le robot considéré est capable d'apprendre des relations porteuses de sens entre actions, outils, d'autres objets et les effets associés, ainsi que d'exploiter la connaissance acquise pour faire des prédictions et prendre des décisions optimales.

4.6 Algorithmes de partitionnement (clustering)

Le clustering ou partitionnement de données fait partie du domaine de l'analyse de données. Cette technique a été conçue pour traiter des jeux de données trop massifs et trop complexes pour être traités manuellement. L'idée est de grouper un ensemble d'objets de telle manière à ce que les objets présents au sein d'un même cluster soit le plus similaire possible tandis que les différences entre les différents clusters soient les plus grandes possibles afin d'obtenir une séparation nette entre clusters. Le clustering est considéré comme une des questions les plus importantes en apprentissage non supervisé, c'est-à-dire sans connaissance préalable sur la classification des objets [14], [15], [16], [17] et [18].

Les variables utilisées dans le clustering utilisé durant notre stage sont des points dans un espace continu. L'objectif consiste à grouper ces points de manière à ce qu'ils puissent constituer des ensembles de point cohérents entre eux et que ces ensembles soient distinguables.

Différentes notions de mesure de distance (ou dissimilarité) existent. Parmi celles-ci, il est possible de citer :

1. Distance de Minkowski
2. Distance euclidienne
3. Distance cosine
4. Distance de Mahalanobis
5. Distance moyenne

La qualité d'un clustering dépend donc de la mesure de distance utilisée.

Différentes typologies de clustering existent. Voici quelques exemples :

1. Partition : initialement, les éléments sont considérés comme faisant partie d'un seul cluster. Puis les éléments initiaux sont groupés de façon itérative en différentes partitions dont le nombre est donné en paramètre en entrée ;

2. Connectivité (hiérarchique) : ce type de clustering partitionne les éléments dans un arbre où chaque noeud de l'arbre représente un cluster ;
3. Distribution : les clusters sont modélisés en utilisant une distribution statistique ;
4. Densité : un cluster est une région dense d'objets entourée d'une région de faible densité. Les éléments présents dans les zones de faible densité sont considérés comme du bruit ;
5. Grille : les données sont partitionnées en un nombre de cellules formant une structure de grille.

4.7 Algorithmes génétiques

L'évolution biologique peut être considérée comme un mécanisme d'optimisation qui vise à permettre aux meilleurs individus de survivre dans leur environnement. Afin de résoudre des problèmes d'optimisation difficiles, le concept d'algorithme génétique a été proposé par John Holland et son équipe [19] dans les années 1970. Les algorithmes génétiques s'inspirent donc de l'évolution biologique et se basent sur des mécanismes similaires d'évolution. Ils ne visent pas à trouver une solution exacte au problème posé mais plutôt à trouver une solution qui satisfasse le plus aux différents critères. Par ailleurs, ils s'appliquent surtout à des problèmes qui ne sont pas résolvables en temps raisonnable avec des méthodes analytiques ou algorithmiques classiques. Le calcul s'effectue en parallèle sur une population d'individus initialisée aléatoirement qui correspondent aux solutions potentielles au problème posé. Un individu contient un génotype qui est encodé sous la forme de valeurs réelles ou entières. Il existe d'autres encodages possibles et fréquemment utilisés, notamment des graphes. La génération de nouveaux individus s'effectue grâce à trois opérateurs d'exploration de l'espace de recherche. Le croisement consiste en la création d'un nouvel individu à partir de plusieurs autres individus. La mutation vise à la modification aléatoire d'une partie de l'individu. Enfin, chaque individu est évalué à l'aide d'une fonction coût et les individus les plus performants sont sélectionnés parmi la population courante d'individus. Ces cycles d'évolutions sont répétés tant qu'une solution correcte n'a pas été trouvée.

Les algorithmes génétiques multi-objectifs sont une variante des algorithmes génétiques classiques. Ils sont utilisés lorsque le problème posé comporte plusieurs objectifs à optimiser. Les solutions à ce problème ne sont pas uniques mais sont définies en un ensemble de solutions optimales, localisées sur le front de Pareto de l'espace des solutions. Elles constituent donc un ensemble de solutions optimales au regard des différents paramètres optimisés. L'algorithme NSGA-II[20] est une version améliorée de NSGA et repose sur trois principes. Tout d'abord, il favorise l'élitisme : seuls les meilleurs individus parmi les parents et les enfants sont gardés. Il favorise par ailleurs les solutions non dominées et enfin, il utilise une variété explicite de solutions.

Ces algorithmes génétiques ont des domaines d'applications très variés : optimisation, apprentissage ou étude du vivant.

5 Notre travail

5.1 Environnement

AFIN de développer des algorithmes d'intelligence artificielle dans le domaine de la robotique, il est nécessaire de disposer d'une multitude d'outils tels qu'un simulateur et un framework pour communiquer avec les différents composants, en plus de disposer, tout naturellement, d'un robot.

Dans le cadre du projet DREAM, l'ISIR utilise le robot Baxter⁹ (voir figure 3). C'est un robot créé par Rethink Robotique et utilisé à la fois à des fins industrielles et de recherche. Ce robot anthropomorphique dispose d'un « visage animé » et de deux bras articulés possédant chacun sept degrés de liberté et de multiples senseurs¹⁰. Le robot peut être contrôlé par un programme écrit par l'utilisateur, mais il est aussi possible de lui faire apprendre la réalisation de tâches en guidant ses bras et effecteurs lors d'une phase de programmation gestuelle.

9. <http://www.rethinkrobotics.com/baxter/tech-specs/>

10. http://sdk.rethinkrobotics.com/wiki/Hardware_Specifications



FIGURE 3 : Le robot Baxter de Rethink Robotics utilisé à l'ISIR.

est un simulateur nécessaire pour valider le bon comportement du robot et éviter ainsi tout problème en situation réelle. La figure 8 donne un aperçu d'une simulation lancée avec Gazebo.

Un modèle du Baxter est fourni et le simulateur Gazebo¹¹ permet de faire des expériences en simulation. Travailler en simulation procure un certain nombre d'avantages, notamment : coût moindre, facilité de test, vitesse d'exécution, robot est toujours disponible.

MoveIt!¹² est un framework open source conçu pour la manipulation de robots et inclut de nombreux composants de planification de mouvements, manipulation, perception 3D, cinématique, contrôle et navigation. Ce framework permet de développer facilement des applications robotiques complexes.

ROS¹³ est un framework flexible pour contrôler un robot. Il comporte différents outils, bibliothèques de fonctions et permet l'interaction et la communication de tous les composants, notamment sous la forme de *service* (client/serveur) ou de *topics* qui exposent des données (subscriber/publisher).

5.2 Discrétisation adaptative

Travaux relatifs TODO

- On a un Baxter, devant une table sur laquelle il y a un objet (un cube)
- On s'intéresse aux déplacements de l'objet par le robot
- Le robot peut pousser l'objet dans différentes directions en déplaçant son end effector parallèlement au plan de la table
- Avant le stage, le robot utilisait des mouvements prédéfinies pour réaliser un nombre réduit d'effets connus (haut/bas/gauche/droite, fig.4)

11. <http://gazebosim.org>

12. <http://moveit.ros.org>

13. <http://www.ros.org/>

- Maintenant on utilise un algorithme de babillage qui génère plein de mouvements qui ont plein d'effets différents, le but est de trouver automatiquement comment bien clusteriser cela pour obtenir un nombre réduit de clusters d'effets pertinents
- N'hésite pas à t'appuyer sur ta partie 4 en y faisant référence.

Notre travail se base en partie sur les travaux existants d'un doctorant, intégré également dans le projet DREAM, et portant sur la découverte des affordances. Dans ces travaux, le nombre d'actions disponibles et connues par le robot est arbitrairement paramétré (par exemple : **Gauche**, **Droite**, **Haut**, **Bas**). La figure 4 montre ces quatre effets réalisables par le robot en simulation.

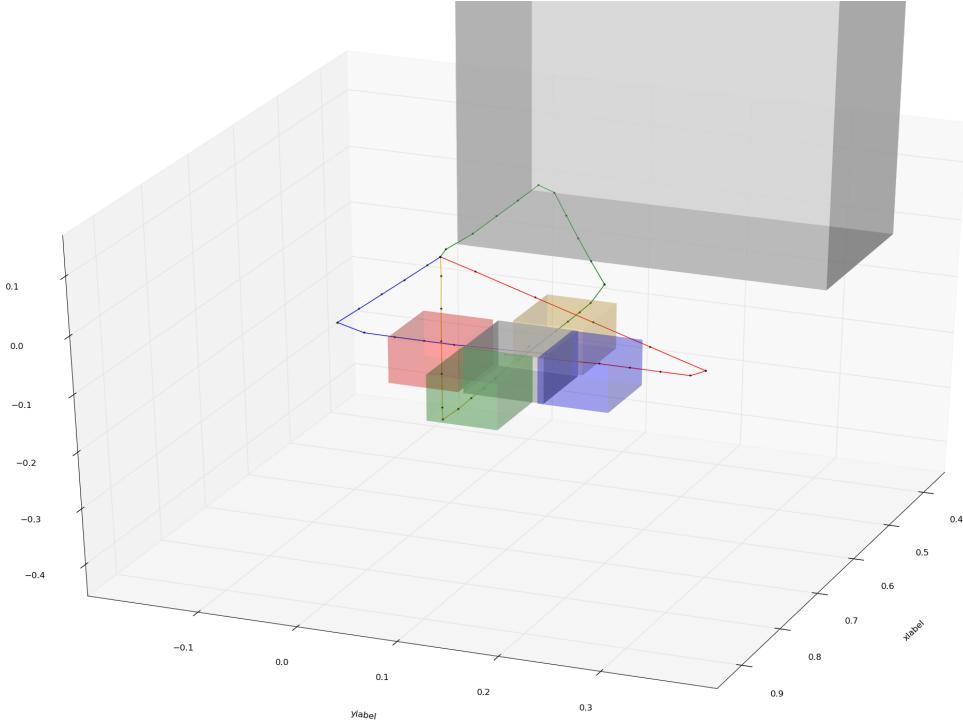


FIGURE 4 : Différentes trajectoires réalisées par le robot (représenté par le volume gris). La position initiale de l'objet est représentée par le cube gris au milieu. Chaque cube coloré correspond au mouvement de la même couleur (par exemple, le cube rouge est la position finale du cube gris après avoir poussé par l'effecteur du robot suivant la trajectoire rouge).

En raison de la nécessité de généraliser dans un contexte où le nombre d'actions est potentiellement différent et surtout inconnu, il n'est pas possible de laisser une telle valeur arbitraire. Pour pallier à ce problème, il est nécessaire de procéder à une discréétisation adaptative de l'environnement, ce qui constitue le sujet principal de notre stage. L'idée générale est de laisser le robot découvrir lui-même les différentes actions réalisables sans en spécifier le nombre. Pour arriver à cet objectif, la méthode choisie lors de la définition du sujet est le clustering qui doit permettre de distinguer les différentes actions.

Algorithmes de clustering En nous basant sur la littérature existante ([14], [15], [16] et [18]), nous avons réalisé en début de stage une synthèse des algorithmes de clustering les plus communément utilisés. La production scientifique est riche et il existe aujourd'hui plus d'une centaine d'algorithmes de clustering, groupés en différentes catégories (voir section 4.6). Afin de filtrer et sélectionner les algorithmes les plus pertinents pour le problème posé dans notre stage, nous avons choisi plusieurs critères non arbitraires. En se référant à la généralisation indispensable mentionnée ci-dessus, un premier critère concerne la nécessité de ne pas fournir le nombre attendu de clusters en paramètre. En effet, le système doit pouvoir de trouver de façon autonome un nombre d'actions permettant un clustering efficace. Un second critère concerne la possibilité de travailler avec de nombreuses dimensions. Au départ,

seules deux dimensions sont prises en compte, les coordonnées x et y . Cependant, une évolution envisagée dès le départ concerne l'ajout de nouvelles dimensions pour le clustering telles que les trois dimensions concernant la rotation de l'objet. Le troisième et dernier critère concerne la disponibilité d'une implémentation en Python ou C++. En effet, l'objectif du stage n'était pas d'implémenter un algorithme en particulier mais de réutiliser facilement des implémentations déjà existantes. Les algorithmes ne satisfaisant pas à ces critères ont été rejettés.

Sur la centaine d'algorithmes de clustering existants, sept ont été retenus lors de cette première phase : X-means¹, Affinity Propagation¹, DBSCAN², HDBSCAN², OPTICS², Mean-Shift² et Level Set Tree. (1) correspondent à des algorithmes de partitionnement et (2) correspondent à des algorithmes basés sur la densité.

Intuition L'enjeu consistait à mettre au point une méthode adaptative permettant de découvrir automatiquement le meilleur algorithme de clustering et les meilleurs paramètres associés pour tout jeu de données considéré.

Pour cela, notre intuition a été de réfléchir à une manière d'optimiser les paramètres des différents algorithmes jusqu'à obtenir une convergence vers un ensemble de paramètres optimaux relativement aux données expérimentales considérées. Les algorithmes génétiques sont par nature efficaces pour obtenir des résultats bons à défaut d'être exacts. Plusieurs critères étant à optimiser, nous avons donc fait le choix d'un algorithme génétique multi-objectifs.

Implémentation Décrire plus généralement ta fonction d'évaluation : on considère les clusters générés par le clustering comme autant d'actions élémentaires, et on cherche à déplacer (en simulation) l'objet vers une zone cible en enchaînant ces actions élémentaires en utilisant un algorithme de planification simple. Et on définit "exécuter une action" comme exécuter un mouvement au hasard dans le cluster correspondant, donc les actions sont stochastiques, donc on répète plusieurs fois l'expérience pour chaque individu et on fait des moyennes. N'hésite pas à faire un petit schéma (semblable à ce qu'on voit figure 7 mais "à la main") pour aider à expliquer éventuellement.TODO

Pour implémenter notre algorithme, nous avons utilisé la bibliothèque C++ open source Sferes [21] créée au sein de l'équipe AMAC et dédié à l'optimisation évolutionniste. Ce framework a pour avantages d'être léger, multi-coeur et facilement extensible. Sa rapidité est comparable à celle de codes dédiés. De nombreux algorithmes de l'état de l'art y sont inclus (par exemple CMA-ES ou NSGA-II). La mise en place est rapide dans la mesure où il ne reste qu'à rédiger le code de la fonction coût pour obtenir un algorithme évolutionnaire complet.

L'algorithme ci-dessous est utilisé par la fonction coût au sein de l'algorithme génétique NSGA-II :

Algorithm 1 Evaluation algorithm for fitness function

```

1:  $E = \{C_{e1}, \dots, C_{en}\}$                                      ▷ Set of clusters containing effects
2:  $F = \{f_1, \dots, f_n\}$                                      ▷ Set of final points
3:  $I = (0;0)$                                                  ▷ Initial point
4:  $t = 0$                                                        ▷ Number of tries
5: Compute the set of effects  $E$  from genotype
6: for all final point  $f_i$  in  $F$  do
7:   for  $j = 1$  to  $nb\_repeat$  do
8:     while  $dist(I, f_i) > \varepsilon$  and  $t < t_{max}$  do
9:        $M = \{m_i \mid m_i = \text{rand}(C_{ei}) \forall C_{ei} \in E\}$ 
10:       $m = \underset{m_i \in M}{\text{argmin}}(dist(I + m_i, f_i))$ 
11:      Apply movement  $m$  to  $I$ 
12:       $t = t + 1$ 
13:    end while
14:  end for
15: end for
16: return (nb clusters, nb movements, final distance)

```

Le principe de cet algorithme itératif au sein duquel un algorithme de clustering est réglé et contenu dans la fonction coût est expliqué ci-après. Le génotype d'un individu contient le type d'algorithme de clustering à utiliser et un ou plusieurs paramètres associés à cet algorithme (générés par l'algorithme NSGA-II). Ces paramètres permettent de constituer un clustering où les clusters représentent des ensembles d'effets. Pour chaque point final, un processus itératif est initié dans lequel, à chaque étape, l'objectif est d'atteindre le point final. Ce processus pour chaque point final est répété *nb_repeat* fois. Pour cela, à chaque itération, un effet est sélectionné aléatoirement dans chaque cluster. L'effet gardé et ajouté au point courant est celui qui permet de se rapprocher le plus possible du point final. L'itération est répétée tant que la distance entre le dernier vecteur utilisé et le point final est supérieure ou égale à *epsilon* et que le nombre d'itérations ne dépasse pas une certaine limite *t_max*. Enfin, le nombre de clusters générés avec les paramètres d'entrée ainsi que les valeurs moyennes du nombre de mouvements requis pour atteindre les points finaux et la distance finale sont retournés à la fin de la fonction coût.

Les trois objectifs devant être satisfait correspondent à : ① le nombre de mouvements pour atteindre la cible, devant être minimisé pour faire en sorte que le robot soit le plus rapide possible, ② le nombre de clusters, devant être minimisé pour simplifier le calcul et ③ la distance finale, à minimiser pour la précision du contrôle. Dans la mesure où plusieurs objectifs sont à minimiser, nous avons sélectionné l'algorithme génétique NSGA-II[20] pour remplir cette tâche.

>L'algorithme de clustering *X-means* [22] est un algorithme utilisé dans différents travaux de recherche sur les affordances pour clusteriser les effets [5], [23].

Sa principale caractéristique est d'être basé sur l'algorithme *K-means* et de le généraliser en tentant de déterminer automatiquement le nombre *k* de clusters. La validation expérimentale de notre méthode a été effectuée en utilisant l'algorithme *Mean Shift*. En effet, au travers de différents tests sur des jeux de données et des paramètres différents, cet algorithme se classait parmi les meilleurs. Cet algorithme a révélé empiriquement par la suite la génération de clusters plus divers que *X-means* lorsque ses paramètres ont été changés.<

TODO : tu as commencé par X-means pour raisons de simplicité, et tu as constaté que Mean Shift marchait mieux donc tu es passé à Mean Shift. Tu peux changer de paragraphe quand tu commences à décrire le génotype.

Le génotype est constitué par la valeur d'un seul paramètre décimal associé à cet algorithme. La population initiale est constituée de 32 individus, évalués durant 25 générations. Les taux de croisement et de mutation pour NSGA-II sont respectivement de 0.8 et 0.1. Nous avons utilisé un jeu de données généré par un babillage orienté-objet simulé (voir figure 5). Un unique point final a été aléatoirement généré au début de l'expérience. La fonction coût a été évaluée 30 fois pour chaque individu, les valeurs moyennes des objectifs étant ensuite reportées.

Les résultats ont été comparés avec une discréétisation obtenue par une recherche de type *Grid Search* dans l'espace de paramètres. Le résultat obtenu est une succession de graphiques représentant les différents clusters obtenus avec les différentes valeurs de paramètres. Des exemples de clusters obtenus pour différentes valeurs du paramètre *quantile* de *Mean Shift* sont listés dans la figure 5.

NSGA-II étant un algorithme multi-objectif, plusieurs meilleures solutions peuvent être trouvées, correspondants aux individus dominants le front de Pareto. La figure 6 montre la valeur du paramètre (de l'algorithme de clustering) correspondant aux individus dominants. Différentes valeurs peuvent être relevées, par exemple 0.05, 0.12 ou 0.31.

À partir de cet ensemble de meilleures valeurs de paramètres, nous avons généré (voir figure 7) deux séries de graphiques. Les deux ensembles de clusters optimisent différents objectifs : vitesse et précision pour celui de gauche contre coût computationnel pour celui de droite. La colonne de gauche contient davantage de clusters et permet un contrôle rapide et précis. À l'inverse, l'autre solution repose sur un nombre moins important de clusters.

Cette première partie de notre stage, comprenant la proposition d'un algorithme de clustering d'effets, a fait l'objet d'une soumission d'un abstract au workshop *Developmental learning and representation building in human-robots and ambient intelligence systems* de la conférence ECAL2017¹⁴ qui se tiendra à Lyon en septembre 2017 (voir Annexe).

14. <https://project.inria.fr/ecal2017/>

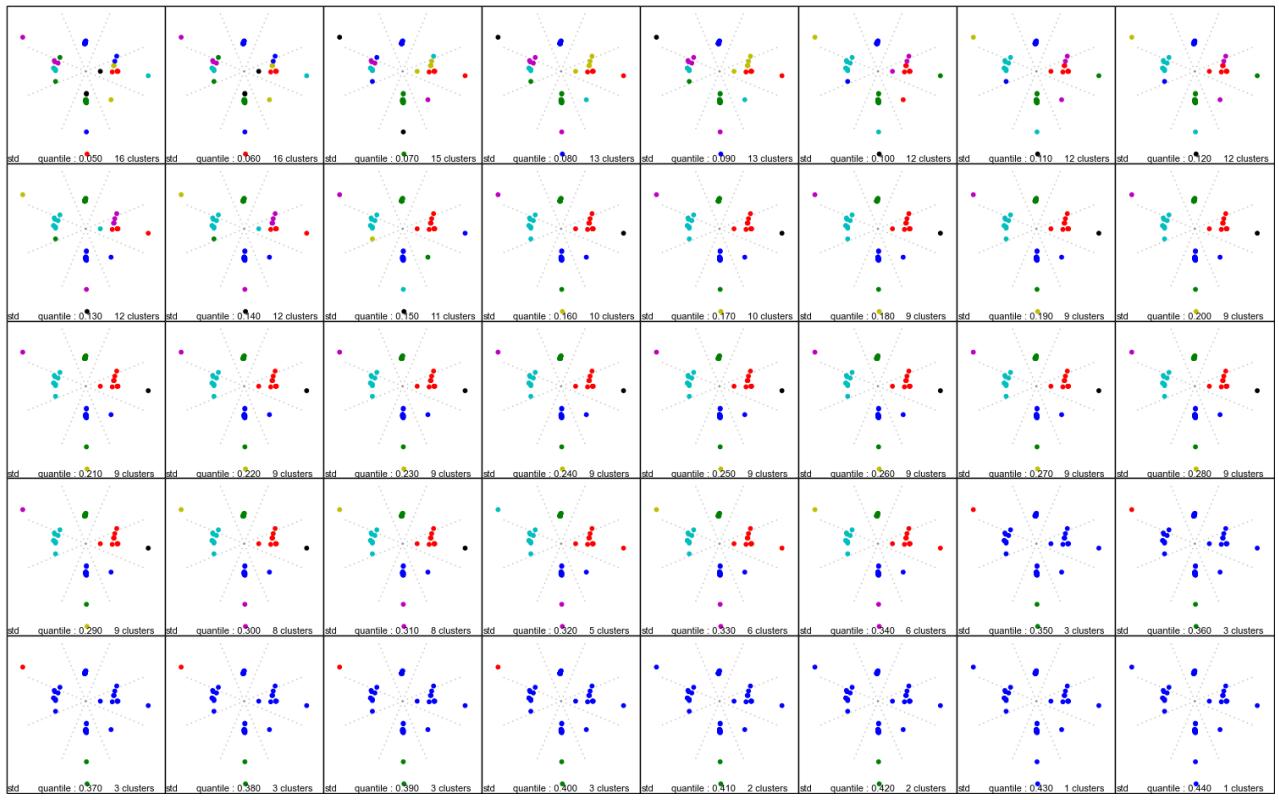


FIGURE 5 : Clusters en fonction de l'évolution du paramètre *quantile* de l'algorithme *Mean Shift*. Les valeurs sont comprises dans l'intervalle [0.050 ; 0.44].

5.3 Expérience de validation

La seconde partie de notre stage a consisté à mener une expérience pour valider l'algorithme de clustering d'effets défini dans la première partie de ce mémoire.

Dans un premier temps, l'expérimentation s'est déroulée en simulation. Une fois cette simulation validée (non encore réalisé à ce jour), il est prévu d'utiliser le robot afin de tester notre algorithme dans un environnement réel.

Cette expérience peut être divisée en 3 étapes : tout d'abord la génération du babillage sensorimoteur et l'enregistrement des effets pour les différents objets, puis la réalisation du lustering des effets et enfin le test de l'apprentissage. La figure 8 représente la mise en place initiale de l'expérimentation.

Babillage Durant la première étape de cette expérimentation, un objet (par exemple un cube) est placé sur la table à une position initiale. Un babillage sur l'objet est réalisé afin d'obtenir des jeux de données contenant les différents effets découverts à traiter. Ce babillage est effectué à l'aide de l'algorithme *NovEB* (Novelty-driven Evolutionary Babbling) décrit par Maestre et al. [9]. Cet algorithme utilise l'algorithme évolutionnaire *Novelty Search* [24] pour générer des trajectoires sans connaissance de l'environnement et des objets à priori. Il est capable de générer

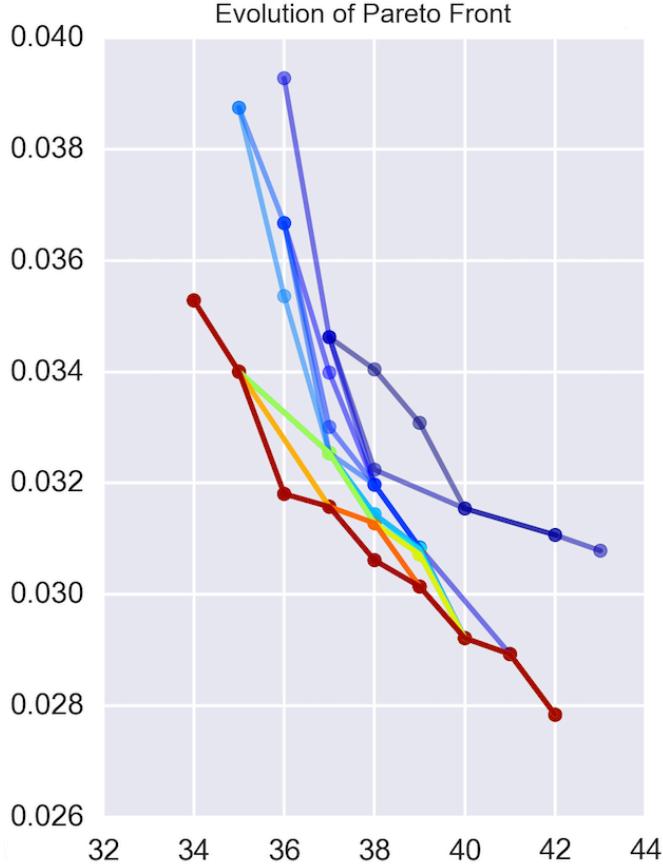


FIGURE 6 : Valeur du paramètre de l’algorithme de clustering Mean Shift correspondant aux individus dominants le front de Pareto. Plusieurs valeurs se détachent, par exemple 0.05, 0.11 ou 0.3. TODO

plusieurs milliers d’interactions différentes avec l’environnement, ce qui représente davantage de trajectoires qu’avec un babillage aléatoire. Pour cela, les trajectoires sont générées en se basant sur la maximisation d’un critère de nouveauté. Ce critère est la distance moyenne entre la trajectoire courante et ses k plus proches voisins dans la population courante et dans une archive de trajectoires précédemment explorées. Les trajectoires générées sont celles qui augmentent le plus de nouveauté. L’algorithme *NovEB* génère un dataset qui peut être traité ultérieurement par des algorithmes d’apprentissage automatique. La figure 9 montre un exemple de trajectoire touchant l’objet considéré.

Clustering d’effets La seconde étape de l’expérience de validation implique l’algorithme que nous avons présenté dans la première partie et vise à clusteriser les effets présents dans les jeux de données. Il en résulte une variété de paramètres de l’algorithme de clustering considéré qui peuvent être exploités durant la troisième étape. Les meilleurs paramètres trouvés dans l’étape précédente sont utilisés pour pousser les différents objets vers la zone cible. Dans notre expérience, une valeur du paramètre *quantile* est sélectionnée (par exemple 0.18) et utilisée pour générer le clustering utilisé par le robot.

Test de l’apprentissage Enfin, dans un troisième temps, l’objectif du robot est de pousser l’objet présenté devant lui vers une zone cible (dont la position est variable), en utilisant les différents effets à sa disposition trouvés lors de l’étape précédente. Cette dernière étape soulève plusieurs problèmes qu’il a fallu ou qu’il faut résoudre.

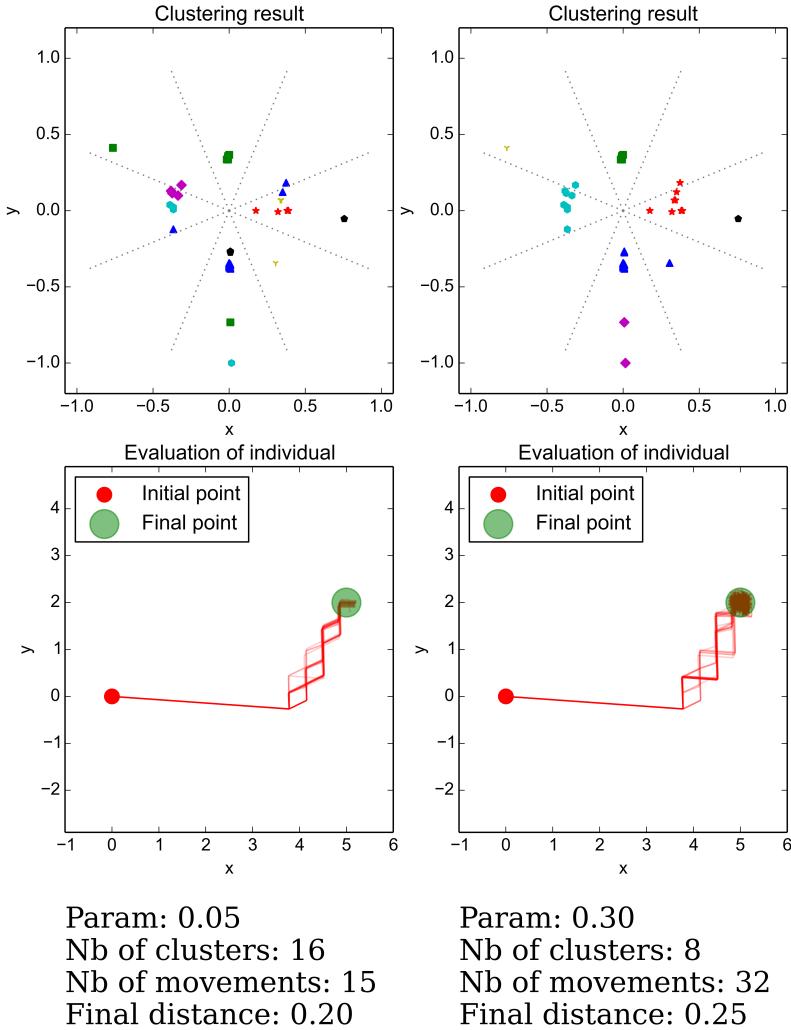


FIGURE 7 : Les deux colonnes montrent différents résultats de clustering et de trajectoires d'évaluations basés sur les valeurs de paramètres appartenant aux individus dominants le front de Pareto.

Le premier problème provient du fait que les différents mouvements réalisés par le robot ne sont pas tous identiques. Certains effectuent une poussée au milieu de la face considérée tandis que d'autres vont toucher cette même face à des positions très variées, parfois même aux extrémités de celle-ci. Un mouvement au centre de la face a pour effet de pousser de manière rectiligne l'objet tandis qu'un mouvement en dehors de ce centre peut provoquer une rotation de l'objet sur l'axe z en plus d'une trajectoire rectiligne. Il convenait donc d'orienter l'effecteur pour qu'il se retrouve dans une configuration idéale, c'est-à-dire parallèle à la face considérée, avant tout mouvement.

Le second problème a trait avec la tâche elle-même. En effet, dans un second temps de l'expérimentation, l'objet doit être poussé dans une zone inaccessible au bras du robot. Cela signifie que l'objectif visé n'est pas simplement de pousser un objet jusqu'à une zone particulière, mais d'effectuer un geste avec une vitesse élevée afin de pousser fortement l'objet jusqu'à cette zone. L'utilisation d'algorithme de *Quality Diversity* doivent permettre d'effectuer un babillage relatif à la vitesse des lancers.

En raison de cette dernière difficulté, l'expérimentation a consisté initialement à pousser un objet dans une zone éloignée mais restant accessible aux bras du robot.

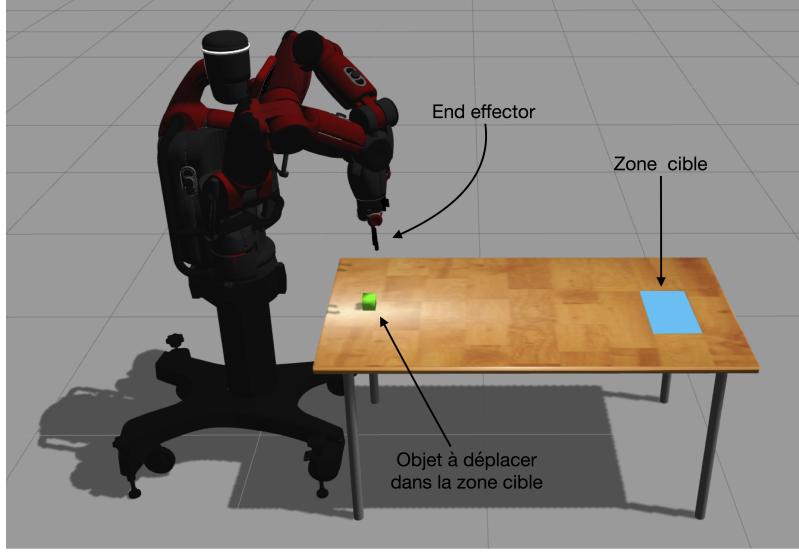


FIGURE 8 : L'objectif du robot est d'apprendre à pousser différents objets (par exemple, le cube vert) depuis sa position initiale jusqu'à la zone cible en bleu, en dehors de la portée du bras du robot. Une première étape de babillage permet de découvrir les effets qui sont clusterisés à l'aide de l'algorithme de clustering. Enfin, dans une troisième étape, le robot tente d'atteindre la zone cible.

Résultats Les premiers résultats provenant de cette expérimentation en simulation montrent que le robot est capable de pousser un objet vers une zone particulière accessible par le robot en se basant sur la discréétisation d'effets. Afin de pousser l'objet dans la direction adéquate, le robot positionne son bras dans le sens opposé au déplacement. Le robot effectue ensuite un déplacement correspondant à l'effet choisi et continue jusqu'à ce que l'objet se trouve dans la zone souhaitée.

Une vidéo de ces résultats est disponible en ligne : <https://youtu.be/bKCBSDx8L1I>

6 Travaux futurs et perspectives

PLUSIEURS axes d'améliorations peuvent être proposés. Durant ce stage, nous avons testé l'algorithme présenté ci-dessus avec un seul algorithme de clustering, *Mean Shift*. Le premier axe d'amélioration consisterait à tester davantage d'algorithmes de clustering, tels qu'*Affinity Propagation*, *Birch* ou *DBSCAN*.

Un second axe d'amélioration pourrait être de tester l'algorithme avec des tâches plus complexes. Par exemple, l'expérience pourrait consister à pousser l'objet dans une zone hors de portée du bras du robot. Cette expérimentation nécessiterait l'utilisation d'algorithme de *Quality Diversity* [25] pour générer un ensemble complet de déplacements. Au sein de l'équipe AMAC, ce type particulier d'algorithmes est utilisé pour générer des trajectoires qui permettent au Baxter de lancer une balle dans un seau [26].

Un troisième axe concerne l'expérience utilisant le clustering d'effets. Il pourrait être pertinent de la réaliser en utilisant d'autres objets ayant différentes propriétés : un cylindre et un cône. En effet, un cylindre a pour particularité de rouler sur une grande distance rectiligne lorsqu'il est poussé depuis sa face courbe alors que ce mouvement est plus réduit lorsqu'il est poussé depuis l'une de ses bases. À l'inverse, un cône décrit une courbe s'il est poussé depuis sa face courbe et nécessite d'être poussé depuis sa base pour parcourir une grande distance rectiligne. Un babillage avec ces deux objets pourrait permettre au robot d'apprendre à repositionner les objets afin d'exposer leur face appropriée aux différentes actions (face courbe pour le cylindre et base pour le cône).

L'algorithme décrit dans ce mémoire a été utilisé pour une expérience réalisée en collaboration avec UDC et ISIR en vue de la réunion d'évaluation du projet DREAM qui se tient mi-septembre à Paris. Le but de cette expérience consiste à faire en sorte qu'un robot mobile puisse atteindre successivement différentes zones matérialisées par des

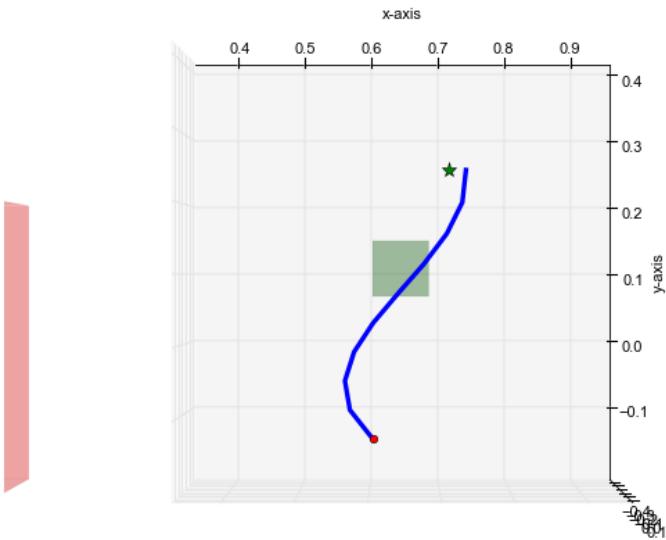


FIGURE 9 : Génération de trajectoire avec NovEB. Le rectangle rouge représente l'emplacement du robot. Le point rouge correspond à la position initiale de l'effecteur du robot. Le tracé en bleu correspond à la trajectoire de cet effecteur vers l'objet. L'étoile verte indique la position de l'objet à la fin de l'exécution de la trajectoire de l'effecteur.

puces RFID. Pour cela, des techniques de Reinforcement Learning ont été utilisées conjointement avec l'algorithme de clustering d'effets. Par ailleurs, cet algorithme peut être utilisé en amont du projet DREAM dans des travaux nécessitant une discréétisation de l'environnement.

7 Conclusion

À PRÈS l'introduction donnée durant mon semestre de cours à Lyon, ce stage de Master 2 m'a permis de davantage découvrir le domaine de la robotique développementale. Ce domaine de recherche, à la croisée de la robotique, de l'informatique et des neurosciences est un domaine riche, en plus d'être fascinant et particulièrement intéressant. Notamment parce que les chercheurs s'emploient à créer des robots qui pourraient booster à partir d'une connaissance presque nulle de l'environnement. Ou parce que le lien étroit entre neurosciences (et spécialement neurosciences cognitives) permet d'implémenter des modèles présents chez l'être humain dans des robots. Toutefois, le sujet abordé dans ce stage pointe la difficulté que représente la conception d'un robot autonome et capable d'apprendre en permanence dans son environnement. Ce stage de six mois m'a également permis de côtoyer concrètement le monde de la recherche, ses méthodes de travail, le rythme nécessaire des publications ou encore les nombreux défis qu'elle implique.

Durant ce stage, nous avons proposé une approche qui permet à un robot d'apprendre depuis ses propres expériences et d'exploiter les données issues du babillage afin d'identifier des ensembles d'effets qui définissent des actions pertinentes pour un contexte dépendant de la tâche à accomplir. Cet algorithme constitue un élément du projet DREAM qui pourra être utilisé par les différents partenaires et l'équipe AMAC.

Tout au long de ces six mois, j'ai acquis un vaste ensemble de compétences nécessaires à la programmation de robots, notamment grâce à l'utilisation de frameworks et d'outils open source. Le thème de mon stage m'a également amené

à acquérir des connaissances solides dans le domaine de l'apprentissage d'affordances, tant d'un point de vue des neurosciences que du point vue de la robotique.

L'avancement de mon stage, se terminant fin septembre, est conforme avec les jalons posés en début de stage. La partie théorique contenant l'algorithme de clustering d'effets est achevée tout comme la partie concernant l'expérimentation en simulation. Le mois restant sera consacrée à mise en place de l'expérimentation avec le robot réel et la constitution d'un benchmark en vue de la publication d'un nouvel article.

Enfin, ce stage a confirmé mon choix de poursuivre en thèse après mon Master 2. La recherche m'attire parce qu'elle est riche de savoirs et en constante progression et c'est un domaine exigeant qui nécessite d'être curieux et ouvert. Le domaine de la robotique développementale m'intéresse beaucoup parce qu'il est prometteur. Cependant, je discerne que je souhaiterais davantage travailler dans un domaine moins fondamental et empirique, et avec davantage de modèles théoriques mathématiques.

Références

1. U. Wagner, S. Gais, H. Haider, R. Verleger, and J. Born. Sleep inspires insight. *Nature*, 427(6972) :352–355, Jan 2004.
2. James Jerome Gibson. *The ecological approach to visual perception*. Houghton Mifflin, Boston, 1979. Includes indexes.
3. Erol Şahin, Maya Çakmak, Mehmet R. Doğar, Emre Uğur, and Göktürk Üçoluk. To afford or not to afford : A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 15(4) :447–472, December 2007.
4. L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor. Affordances in psychology, neuroscience and robotics : a survey. *IEEE Transactions on Cognitive and Developmental Systems*, PP(99) :1–1, 2016.
5. L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances : From sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1) :15–26, Feb 2008.
6. Paul Fitzpatrick and Giorgio Metta. Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society of London A : Mathematical, Physical and Engineering Sciences*, 361(1811) :2165–2185, 2003.
7. M. Lopes, F. S. Melo, and L. Montesano. Affordance-based imitation learning in robots. In *2007 IEEE/RSJ ICIRS*, pages 1015–1021, Oct 2007.
8. Andrew N. Meltzoff and M. Keith Moore. Explaining facial imitation : a theoretical model. *Early Development and Parenting*, 6(3-4) :179–192, 1997.
9. Carlos Maestre, Antoine Cully, Christophe Gonzales, and Stephane Doncieux. Bootstrapping interactions with objects from raw sensorimotor data : A novelty search based approach. *5th Joint International Conference on Development and Learning and Epigenetic Robotics, ICDL-EpiRob 2015*, pages 7–12, 2015.
10. Richard M. Ryan and Edward L. Deci. Intrinsic and extrinsic motivations : Classic definitions and new directions. *Contemporary Educational Psychology*, 25(1) :54 – 67, 2000.
11. Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation ? a typology of computational approaches. *Frontiers in Neurorobotics*, 1 :6, 2009.
12. A. Gonçalves, G. Saponaro, L. Jamone, and A. Bernardino. Learning visual affordances of objects and tools through autonomous robot exploration. In *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 128–133, May 2014.
13. Anh Nguyen, Dimitrios Kanoulas, Darwin G Caldwell, and Nikos G Tsagarakis. Object-Based Affordances Detection with Convolutional Neural Networks and Dense Conditional Random Fields. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3260–3267, 2017.
14. Dongkuan Xu and Yingjie Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2) :165–193, 2015.
15. Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. A roadmap of clustering algorithms : Finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3) :297–314, 2009.
16. Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras. A survey of clustering algorithms for big data : Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3) :267–279, 2014.
17. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Chap 8 : Cluster Analysis : Basic Concepts and Algorithms. *Introduction to Data Mining*, page Chapter 8, 2005.

18. T. Sajana, C. M. Sheela Rani, and K. V. Narayana. A survey on clustering techniques for big data mining. *Indian Journal of Science and Technology*, 9(3) :1–12, 2016.
19. J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
20. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *Trans. Evol. Comp*, 6(2) :182–197, April 2002.
21. J.-B. Mouret and S. Doncieux. SFERESv2 : Evolvin’ in the multi-core world. In *Proc. of Congress on Evolutionary Computation (CEC)*, pages 4079–4086, 2010.
22. Dan Pelleg and Andrew W. Moore. X-means : Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML ’00, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
23. Self-discovery of motor primitives and learning grasp affordances. *IEEE International Conference on Intelligent Robots and Systems*, pages 3260–3267, 2012.
24. J. Lehman and K. O. Stanley. Improving evolvability through novelty search and self-adaptation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2693–2700, June 2011.
25. Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality diversity : A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3 :40, 2016.
26. Seungsu Kim and Stéphane Doncieux. Learning highly diverse robot throwing movements through quality diversity search. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO ’17, pages 1177–1178, New York, NY, USA, 2017. ACM.

Unsupervised effect clusterization

Stanislas Leroy^{1,2}, Carlos Maestre¹, Alexandre Coninx¹, Stéphane Doncieux¹

¹UMR 7222, ISIR, Sorbonne Universités, UPMC Univ Paris 06, Paris, France

²Université Claude Bernard Lyon 1, Villeurbanne, France

Learning affordances is a critical step in a developmental robotics approach. The data required for such learning can be provided by a babbling approach consisting in applying random action and observing the corresponding effects, but this approach raises specific challenges. Defining a priori a limited set of possible actions and corresponding effects is a strong limitation to what the robot can achieve. On the contrary, applying completely random actions results in a large set of possible effects that may be hard to exploit, in particular if the affordance is represented by discrete structures such as Bayesian networks (Montesano et al. (2008)).

In this work, a clustering method is proposed to automatically build a limited set of effects on the basis of data generated by a babbling method relying on random actions and on a given task.

Several methods can be used to perform the aforementioned effect clustering. In the related literature the most popular method is *X-means* (Pelleg and Moore, 2000) that relies on the *k-means* algorithm to define the clustering. Its main feature is to determine an optimal value k , i.e., an optimal number of clusters, based on the dataset used as input. This method has been used multiple times in developmental robotics to clusterize features and effects of objects (Montesano et al., 2008) or to distinguish categories of effects (Ugur et al., 2012). There is a tension between the precision of the representation, that may push towards a large number of effects in order to precisely act and understand what happens, and its simplicity, to reduce the memory and computational cost. The number of clusters therefore depends on a context that cannot be inferred by such an algorithm. It is proposed here to take the context into account to automatically tune the clustering algorithm in order to discover clusters of effects that make sense for the robot given this context.

The task given to the robot provides a mean to evaluate the relevance of a particular discretization. The main hypothesis made in this work is thus that the discretization depends on the context defined by the task to be achieved. The task is defined as a particular position to reach in the effect space associated to an object, for instance moving an object to a particular position. The data generated by the babbling tells

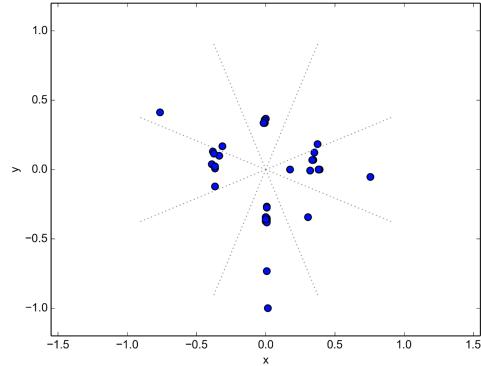


Figure 1: An example of dataset where effects in blue need to be grouped into clusters.

the possible movements in this space. The proposed method consists in an iterative algorithm in which a clustering algorithm is tuned in order to satisfy three different objectives: (1) the number of moves to reach the goal, to be minimized to make the robot as fast as possible, (2) the number of clusters, to be minimized in order to simplify computations and (3) the final distance to the goal, to be minimized for the accuracy of the control.

NSGA-II, a multi-objective evolutionary algorithm (Deb et al., 2002), was used to generate the set of non-dominated parameters of the clustering algorithm optimizing these objectives. The algorithm used to compute the fitness function is described thereafter (Algorithm 1).

An experimental validation of our method was performed using the *Mean Shift* clustering algorithm (Fukunaga and Hostetler, 1975), which empirically revealed to generate more diverse clusters than X-means when their parameters were changed. The genotype was constituted by the value of the single float parameter associated with this algorithm. Ini-

Algorithm 1 Evaluation algorithm for fitness function

```

1:  $E = \{C_{e1}, \dots, C_{en}\}$   $\triangleright$  Set of clusters containing effects
2:  $F = \{f_1, \dots, f_n\}$   $\triangleright$  Set of final points
3:  $I = (0;0)$   $\triangleright$  Initial point
4:  $t = 0$   $\triangleright$  Number of tries
5: Compute the set of effects  $E$  from genotype
6: for all final point  $f_i$  in  $F$  do
7:   for  $j = 1$  to  $nb\_repeat$  do
8:     while  $dist(I, f_i) > \varepsilon$  and  $t < t_{max}$  do
9:        $M = \{m_i \mid m_i = \text{rand}(C_{ei}) \forall C_{ei} \in E\}$ 
10:       $m = \text{argmin}_{m_i \in M} (dist(I + m_i, f_i))$ 
11:      Apply movement  $m$  to  $I$ 
12:       $t = t + 1$ 
13:    end while
14:  end for
15: end for
16: return (nb movements, nb clusters, final distance)

```

tial population was made of 32 individuals, evaluated during 25 generations. Crossover and mutation rates for NSGA-II were respectively set to 0.8 and 0.1. We used a movement dataset generated by a simulated object-oriented babbling (Figure 1). One final point was randomly generated at the beginning of the experiment. The fitness was evaluated 30 times for each individual, the average objective values being reported.

The results were compared to discretizations obtained by a simple grid search over the parameter space.

Figure 2 shows some of the non-dominated points that have been generated. It contains the clustering results and the trajectories generated in the fitness function. The two clusterings optimize different objectives: speed and accuracy versus computational cost. The parameters shown on the left contain more clusters and allow a fast and accurate control. The other solution found relies on less clusters.

In that paper, we have proposed an approach to allow a robot learn from its own experiments and exploit babbling data in order to identify sets of reachable effects that define actions relevant to a task-dependant context. In the future, we will extend that approach by using more clustering algorithms and parameters, and investigate its application to more complex and varied tasks.

Acknowledgements

This research is sponsored in part by the DREAM project¹. This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 640891. This work was also funded by the Labex SMART (ANR-11-LABX-65) supported by French state funds managed by the ANR within the In-

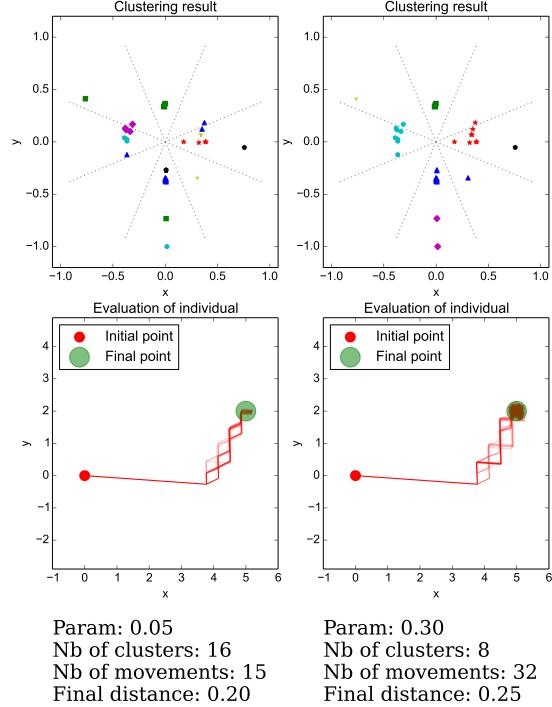


Figure 2: The two columns show different clustering results and evaluation trajectories based on parameter values belonging to individuals on Pareto Front.

vestissements d’Avenir programme under reference ANR-11-IDEX-0004-02.

References

- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE TEC*, 6(2):182–197.
- Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE TIT*, 21(1):32–40.
- Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: From sensory-motor coordination to imitation. *IEEE TR*, 24(1):15–26.
- Pelleg, D. and Moore, A. W. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth ICML*, ICML ’00, pages 727–734, San Francisco, CA, USA.
- Ugur, E., Sahin, E., and Oztop, E. (2012). Self-discovery of motor primitives and learning grasp affordances. In *2012 IEEE/RSJ ICIRS*, pages 3260–3267.

¹<http://www.robotsthatdream.eu/>