LEARNING THE MANIPULATION OF FLEXIBLE TOOLS IN DEVELOPMENTAL ROBOTICS: A FISHING ROBOT

REPORT OF L3 INTERNSHIP

MARIE-MORGANE PAUMARD

FLOWERS TEAM

May 12th – August 1st

INRIA BORDEAUX

SUPERVISORS

Clément Moulin-Frier

Pierre Rouanet

Pierre-Yves Oudeyer

INSTITUTIONS

ABSTRACT

Learning how to manipulate flexible tools is an harsh issue in robotics, since there is generally no analytical model of the system dynamics available. Learning algorithms are therefore a pivotal tool to control such systems. In this report, I present the results of my experiment on the manipulation of a fishing rod by a 2-arm robot equipped with a movement generation and perceptual systems. I study how an optimization algorithm allows the robot to reach particular position of the hook on the floor. Then, I analyze the distribution of effects (i.e. final fishhook position) in different contexts as well as optimization performances for particular goals. The obtained results are encouraging to pursue this work and I provide future directions in the conclusion.

KEYWORDS Developmental Robotics, Dynamical Movement Primitives, Covariance Matrix Adaptation, Fishing robot, Optimization strategies.

ACKNOWLEDGEMENT

I would like to thank my two supervisors Clément and Pierre for their valuable advices and their precious help for my experiments. They also spend a considerable amount of time to reread this rapport so I am very grateful to them.

I also would like to thank all the Flowers members, because they shared with me their passions for their researchs, and especially John for his brilliant idea about the initial position of the fishhook and Alexandre for its design and printing.

CONTENTS

1 INTRODUCTION

Developmental robotics is a scientific field which models the development of cognitive processes in natural and artificial systems. It has close links to machine learning[3], neurosciences and developmental psychology with a dual aim: first to design better autonomous machines able to efficiently learn complex tasks in an open environment and second, to use these machines to better understand child development. Flowers[1], the group where I did my internship, specifically aims at creating curious machines capable of progressing and learning by themselves.

The goal of my internship is making Baxter Robot[2] learn how to manipulate flexible tools. Baxter is a recent commercial robot equipped with two arms (7 degrees of freedom each) and various sensors. The flexible tool I considered is a fishing rod from which we can track the position of the fishhook. The aim is that the robot find accurate motor commands for the arm in order to reach desired positions on the floor. It is a complex issue because there is no analytical model available for such a system, due to the flexible nature of the fishing line.

Hence, the only way to achieve our aim is to provide the robot with learning and exploration abilities, with which it builds its own model of the situation by experience. To do so, the robot can try various motor commands and observe the corresponding effects in term of final hook position on the floor. At the beginning of my internship, the aim was to adapt existing autonomous exploration algorithms, which have recently been released on an open-source library, to achieve this. These strategies, inspired by child development, allow the robot to learn efficient sensorimotor models. I will not detail these strategies here because the topic of my internship has slightly changed and I actually used other types of algorithms.

I rather focused on how to reach a particular hook position—instead of learning how to reach any possible hook position. For this aim, I used an optimization algorithm allowing the robot to iteratively execute arm movement in order to get closer to a desired hook position. I realized various experiments, in simulation and on the Baxter robot, and the results are described in detail in this report.

***

This report is organized as follows. First and foremost, I expose tools and major concepts for movement generation, perception and optimization. Secondly, I summarize my work on the computer simulation of the fishing rod. Thirdly, I describe the robotics experimentation I carried out. Finally, I discuss the results I got in terms of reproductibility and optimization performance.

2 OVERVIEW OF THE SYSTEM

In this section, I present the robot, the movement generation system, the tracking system and the optimization process, that will be used for my experiment.

2.1 *Earlier work*

A previous experiment of fishing robots[10]was done by Sao Mai Ngyuen during her thesis at Flowers. She presented an architecture of learning from imitation and intrinsic motivation. The fishing experiment she conducted intended to demonstate the efficiency of her algorithm. Flowers team gets better means, like Baxter robot or Optitrack's tracker and wants to use other algorithms for movement generation,

hook position tracking and optimization, and this is why I worked on the same experiment.

2.2 *Robot: Baxter*

Baxter is an adaptable robot that is produced by RethinkRobobtics[2], which has two 7-degrees of freedom arms. Safe for humans, it does not require any safety cage, even when one plays violent movements.

I worked with its left arm, on which I attached a tiny fishrod. For security reasons, the degree of freedom of the closest end effector was disabled. This robot allows for accurate reproduction of trajectory in joint space. We choose dynamic movement primitives to generate various and yet smooth movements.

2.3 *Movement generation system: Dynamic Movement Primitives*

Using the robotic system I just presented, reaching various hook positions necessitates complex arm movement generation. Dynamic Movement Primitives (DMPs)[5] have drawn our attention; they can be modelled by a sping-damper equation to which a disturbance may be added to create more complex moves. These complex movements have long been thought to be composed of sets of primitive action building blocks, and DMPs are a proposed mathematical formalization of these primitives. Besides, they guarantee convergence towards the target and allow for spatially and temporally scaling. A DMP is expressed as the sum of two components:

— An *attractor* term allowing the robot actuators to smoothly move from a predifined initial to a final position, following the trajectory of an overdamped spring-damper system (Fig. 1).

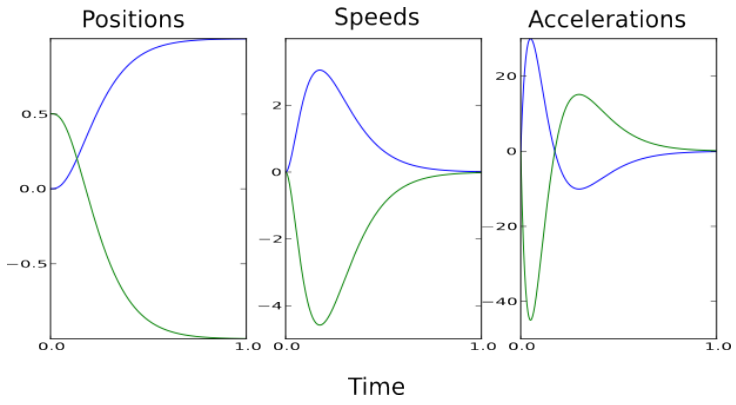


Figure 1: Attractor term: exemple of position, speed and acceleration profiles, generated by two different DMPs. The blue one moves from 0 to 1, and the green one, from 0.5 to -1. Movement dynamics is the same as an overdamped spring-damper system.

— A *forcing* term adding pertubations to the trajectory of the attractor, allowing the generation of more complex movements. This forcing term f is modelled

as a weighted sum of normalized gaussian basis functions and is added to the acceleration profile of the attractor. It is defined by:

$$f(x,g) = \frac{\sum_{i=1}^{N} \psi_i(x)\omega_i}{\sum_{i=1}^{N} \psi_i(x)}$$

where ψ_i is a gaussian function, known as basis function, and ω_i its weighting. In brief, it is a set of Gaussians that are selectively activated during the movement (Fig. 2).

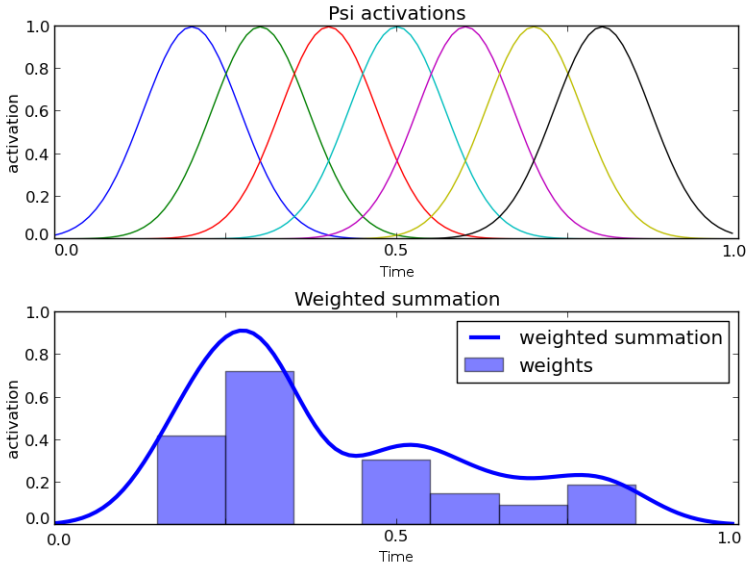


Figure 2: Example of forcing function, composed of 10 basic functions with different weights. Source: Travis DeWolf, Dynamic movement primitives part 1: The basics

Then, the equation of the movement is:

$$\ddot{y} = \alpha_y(\beta_y(g - y) - \dot{y}) + f$$

where y is the state of the system, g the goal, f the forcing term, and α and β two gains. Fig. 3 shows the acceleration profile; positions are obtained by double differentiation, knowing the starting position and velocity integration.

For my experiment, I used Stulp's library[11], binded from C++ to Python. I fixed initial and final positions of the arms movement. The initial position was chosen to limit hook oscillations before starting the movement (Fig. 4). The final position was chosen to allow the hook to systematically touch the floor at the end of the movements, as well as to reach various hook positions in front of the robot. Since those positions are quite close, almost half orders m do not produce movements of the hook. This make the reaching of particular hook positions even more challenging.

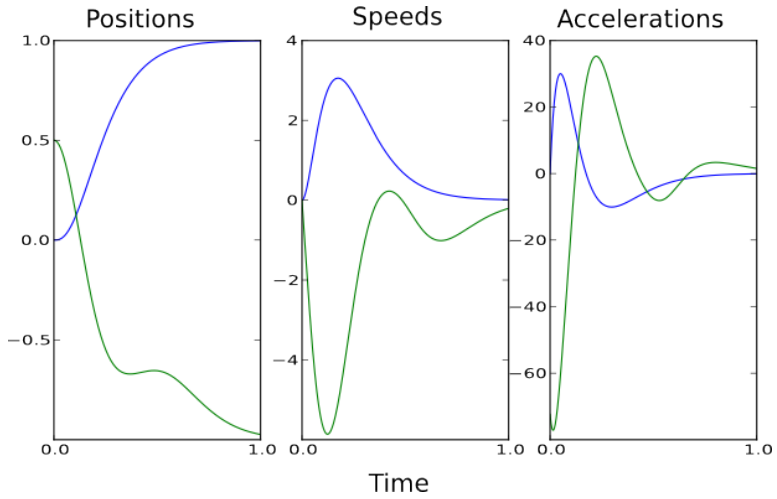


Figure 3: Two DMPs with the same attractor term as in figure 1 but with a forcing term (as illustrated in figure 2) added to the green one, allowing to generate a more complex movement.

Many extensions for DMPs exist, e. g. imitation of demonstration trajectories, but we do not use them in this work.

Figure 4: Initial position: the hook is stopped by the ground.

Thus, a robot arm movement is only parametrized by the weights of the basis functions (inital and final positions, as well as constants of the spring-damper system, being fixed). Them movement of each arm joint is modelled using a DMP. I used a few basis functions per joint perturbating the attractor trajectory during the first half of the movement. In fact, perturbating it near the final position does generally not influence the final hook position. As the robot arm has 6 degrees of freedom, we obtain a 12-dimensional movement representation. This space is called M, for Motor space.

2.4 *Perceptual system: Optitrack*

To record the hook position, I used Optitrack's V120:Trio camera, which provides position and orientation tracking. It produces and receives infrared rays that are reflected by the Optiwand markers, which are infrared-reflecting spheres. It allows to track the 2-D final positions of the hook on the floor. We call S, for Sensory, this space.

IMPROVEMENT OF THE TRACKING First, I consider using the same fishing rod as Mai did, in her experiment. The fishhook consisted in a sphere where I stucked the OptiWands. Due to its shape, the recognition system did not find it most of the time: approximately, only 30% of the final positions are detected. Too close, the reflectors are confused with each other. Too spaced, they are not trackable. Then, I designed and 3-D-printed a flat form, with which I get almost 95% of detection.

Such system is more workable than Mai's, who used a simple camera to find the hook. Optitrack is simpler, more accurate and extensible.

2.5 *Optimization process: Covariance Matrix Adaptation*

In robotics, finding a movement allowing to reach a desired goal is generally expressed as an optimization problem, i. e. the problem of finding a value minimizing a function $\texttt{cost} : M \to \mathbb{R}$. `cost` is called the cost function and maps movement representations (Section 2.3) to their respective performances in reaching a particular goal. The main issue is that I cannot express this cost function analytically because the robotic setup contains flexible parts. Given a goal $s_g \in S$ to be attained (i. e. a particular hook position on the floor), the cost function is therefore only accessible by executing a particular movement $m \in M$ with the robot and observing the corresponding effect $s \in S$. The cost of m is then defined as the distance between the goal s_g and s, the reached position of the hook when executing m.

Previous works (e. g.[12]) have successfully solved such optimization problems in robotics setups by applying the CMA-ES optimization algorithm to DMPs-based movement generation. CMA-ES stands for Covariance Matrix Adaptation Evolution Strategy; I used this state-of-the-art algorithm for my experiments and describe it below.

In the main lines, the algorithm (Alg. 1) first randomly generates motor samples in M from an initial gaussian distribution, using DMPs. This initiated distribution is chosen to generate sample with small weigths in the forcing term. Then, these samples are evaluated with respect to the cost function and the gaussian distribution is updated such that low-cost samples will be more likely to be chosen in the future. This process is repeated until convergence (Fig. 5). CMA-ES actually performs a stochastic gradient descent. Applied on a robotic framework, this allows to find by experience a motor command accurate to reach predefined goal.

Algorithm 1 CMA-ES

```
set λ                                   ▷ number of sample per iteration
initiate the mean and covariance matrices
while not terminate do
    for i ← 1 to λ do
        sample λ new solutions and evaluate them
    order of the sampled solutions
    update the mean and covariance matrices by giving more weight to low-cost samples
```

For my experiment (Section 4.2), I rather choose to terminate after 220 iterations.

CMA-ES can be conceive as an evolutionary algorithm, were variations are generated from the current solution (i. e. mutation), evaluated (i. e. fitness) and weighted (i. e. selection).

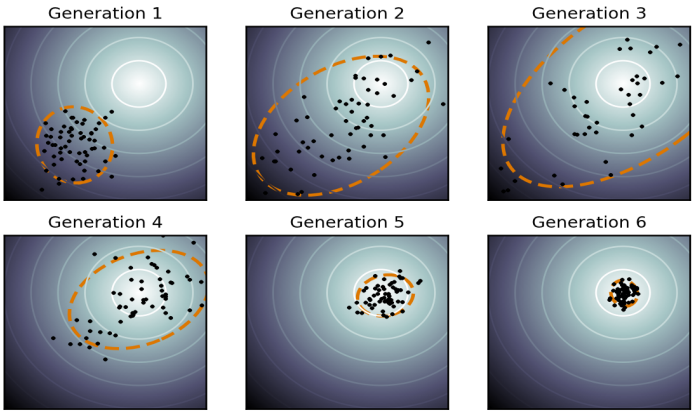


Figure 5: Illustration of an optimization run with CMA-ES on a simple two-dimensional problem. Each generation represents an iteration of the while loop of Alg. A low cost function correspond to the center of the circles. 1.
Source: Wikipedia, CMA-ES, File: Concept of directional optimization in CMA-ES algorithm.png

Henceforth, with those methods and tools, I can run some simulations and carry out experiments.

3 SIMULATION

First, I was asked to produce a simulation sharing properties with my experiment. It prepares me on working with Baxter, that was inaccessible at the beginning of my internship. Moreover, it familiarizes me with the algorithmic tools I will use.

I used a model of multiple pendulum [4], coded with PyDy, which I adapted to my experiment. Such a system can be conceived as a simplified fishing line. I produced a python code, a few plotting functions and a module for a Flower's software library: Explauto[9]. A multiple pendulum consists in few points with given masses beneath one another, that react to the input order applied horizontally on the first one (Fig. 6). The input order correspond to a force trajectory expressed as a serie of step functions or a DMP. This simulation is based on `odeint` and takes a few seconds to solve differential equations, which is quite long. Indeed, the purpose of a simulation is to go faster than an experiment, producing more data.

Figure 6: Exemple of a five joints pendulum exposed to a force.

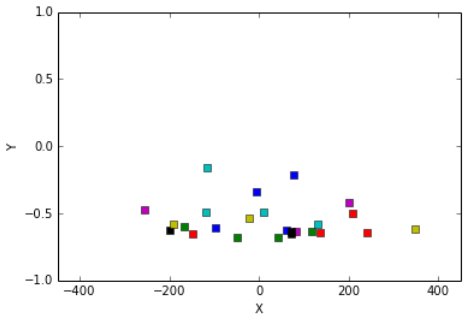


Figure 7: Points reached by the last point for uniformly sempled random orders after ten-seconds simulations

On figure 7, we observe that, even if possible, the reaching of positive value on the y-axis is scarce: producing random movements rarely allows the last point to reach high y-position at the end. We applied an optimization algorithm to make the system learning how to hit the uppest point.

EXPERIMENT To try reaching high y-values, I used CMA-ES. I defined the cost as the distance between the height of the last particule and the topmost value. To limit the horizontal space, I add a term to the cost functions when the pendulum goes away.

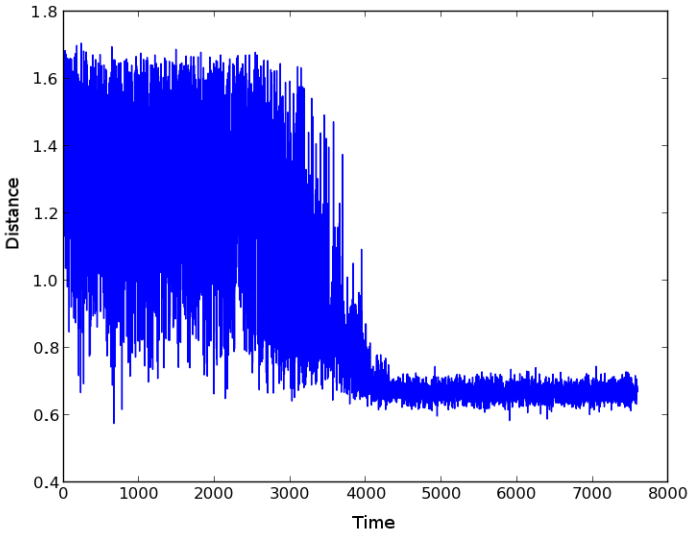


Figure 8: Evolution of the cost function during an optimization run. The cost function is defined as the observed distance between the height of the last point of the pendulum and the uppest reachable point ($y = 1$). The resting position is $y = -1$ so the cost varies from 0 to 2.

On figure 8, we can notice that an acceptable solution is found after 4000 iterations. This high number of iterations is probably due to the complexity of the problem. It is like if a human was doing the same experiment with closed eyes, only acknowledging the final position.

FURTHER WORK I am producing a second simulation, based on PyDy, which I build from scratch. In addition to the n-pendulum, it embedded an articulated arm, which allows us to better manage inertial references. At present, it is not part of Explauto; however I plan to include it as soon as possible.

***

This simulation gives me a general approach of the results I can expect with CMA-ES.

4 EXPERIMENTATION

I will now explain the experiment I carried out with the robot.

4.1 *Basic setup*

Figure 9 shows the experimental setup.

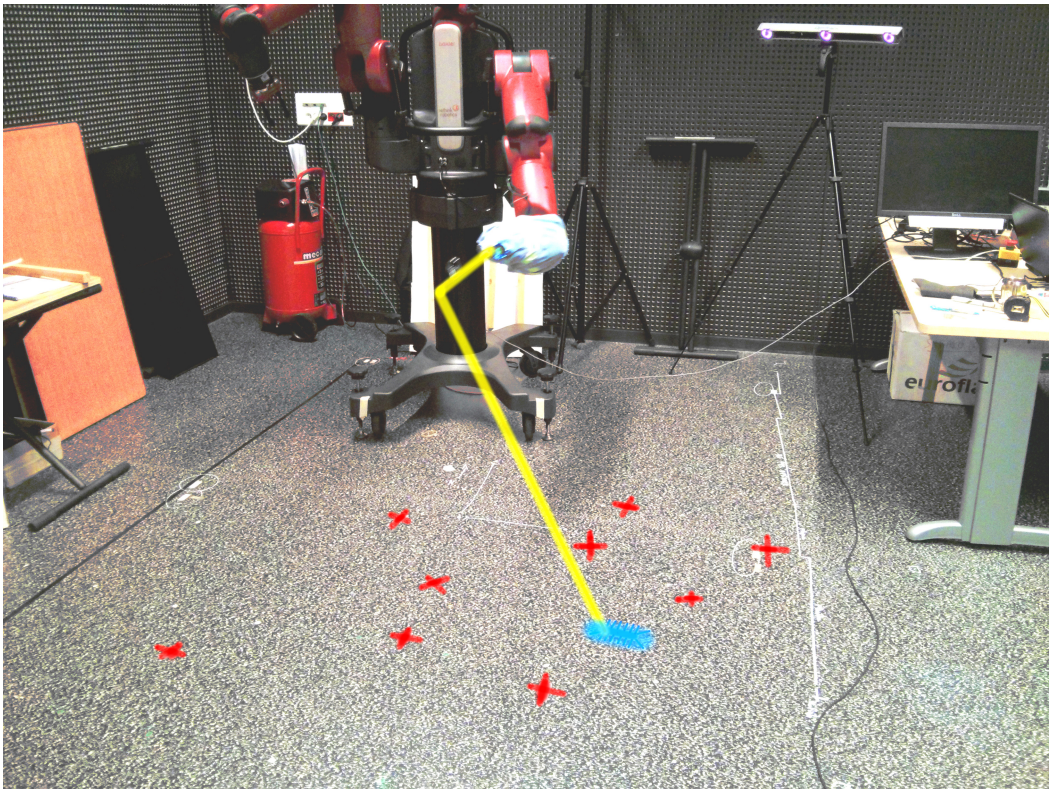


Figure 9: Experimental setup. The hook appears in blue, when the rod is yellow. Red markers are disposed on the floor—they will be the targets to achieve.

What the robot is able to perceive is the final position of the hook in the 2-D space of the floor. However, this former setup provides very unprecise results, this is why I applied some enhancements.

IMPROVEMENT OF PRECISION As the hook is very light, I wondered how it can affect the precision of the casts. I tested the use of plumb on the fishing line. The experiment consisted in picking ten random DMPs and applying each one fivefold.

Figure 10, illustrates the reliability of a few casts. As expected, with the plumbs, tracked points of the same movement are less spaced. However, I found out that returning to the initial position was not a very stable behavior and that the final position of the hook after a particular movement was quite sensitive to the initial condition. This is the reason of the bipolar distribution of the cyan points.

In the final experiments, I fixed the plumbs to the hook.

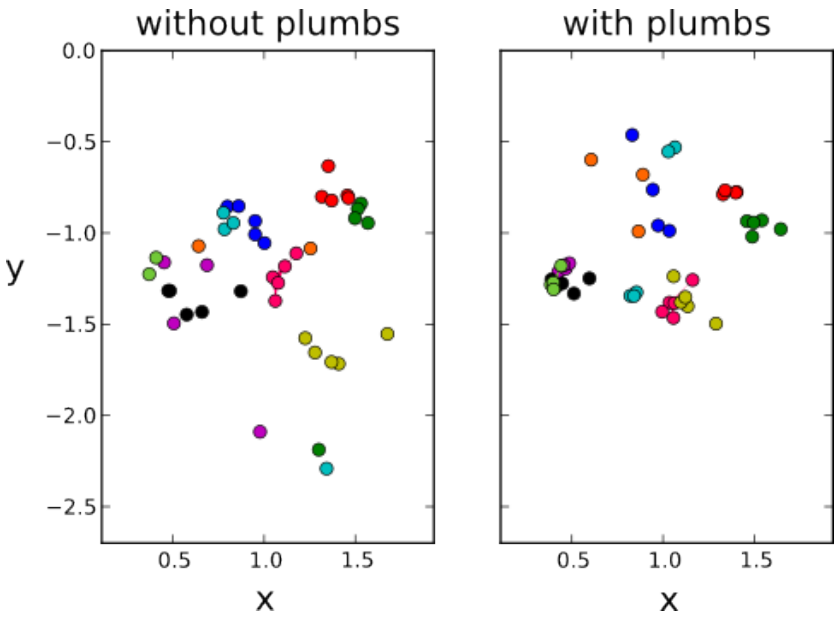


Figure 10: Comparison of precision casts between weighted and not weighted hook. I randomly pick 10 DMPs and play them five times. Each color represents the result of a given command.

IMPROVEMENT OF THE REPRODUCTIBILITY To oppose multipolar distribution of the results, I chose an initial position that stops the fishhook: I was suggested to stop it with the ground, when the end effector is right above it, straightening the line.

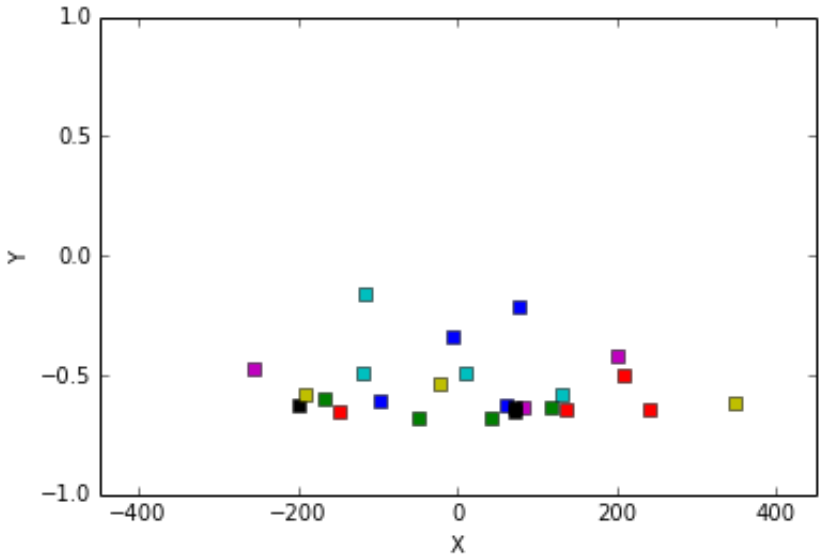


Figure 11: Same experiment as figure 10, with a different initial position.

As assessed by plot 11, there is no multipolar effect anymore, the effects are even closer. The cluster at (1.5, -2.0) is surrounding the initial position because half of the orders does not produce movement.

4.2 *Experiment one: reaching multiple goals*

In this experiment, the robot attempts to reach many goals. For each one, a independent CMA-ES optimization is performed (see section 2.5). The cost function is defined as the distance between the hook position at the end of the movement and the goal (Fig.12). The optimization procedure generates 220 movements for each goal.

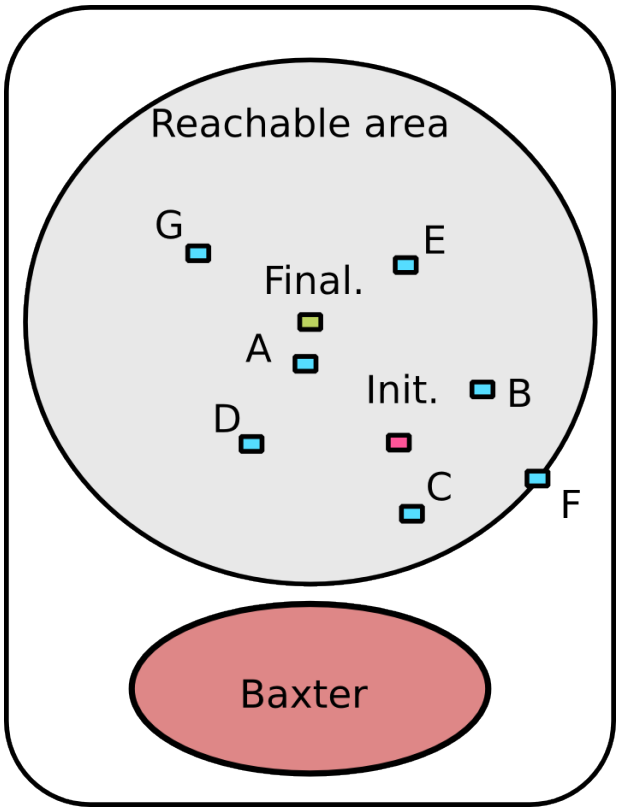


Figure 12: Goals are labelled from A to G. The initial position is pink. The reachable area is the space of the ground that can be reached at the end of the movement. The final position is the projection on the floor of the end of the rod; it is the center of the reachable positions.

***

This experiment provides me with many data to analyse.

5 RESULTS

In this part, I analyse and discuss the results I got.

5.1 *Reachable points*

First, I want to point out the non-uniformity of the effect space. As shown on figure 14, some points are way easier to reach. By easy, I mean that they ar more likely to be reached by a random movement. As expected, the farthest to the initial position

the point is, the hardest to reach it become. Furthermore, the points between the robot and the initial position are laborious to achieve.

Based on those results, goals A, B and E are the simpliest to reach, when goals C, D, F and G are more arduous.

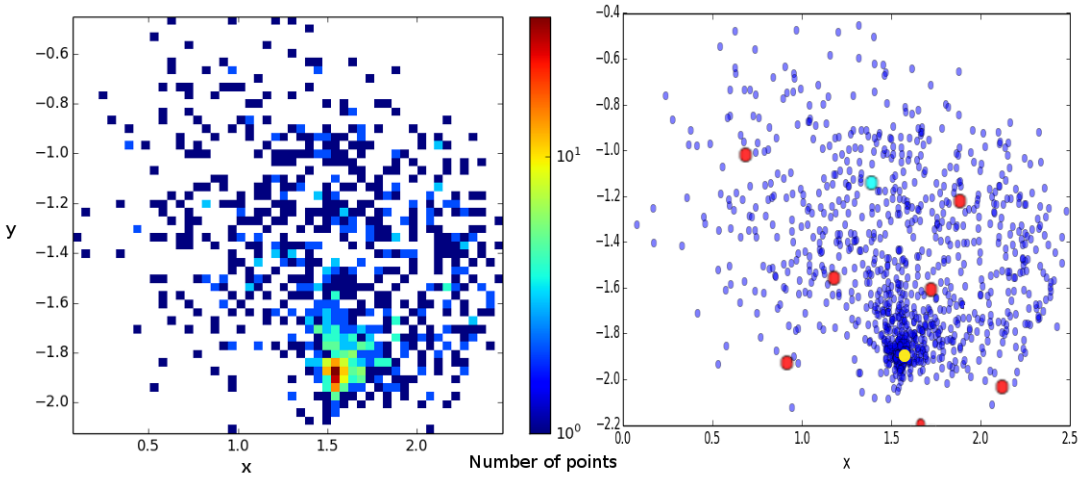


Figure 13: Left: reached points from 1000 random casts. Right: scatter plot of the reached points (blue), the goals (red), the initial (yellow) and final (cyan) positions.

5.2 *Experiment one*

The results of the experiment one are displayed on table 1 and plotted on appendix A.

The table 1 confirms my hypothesis on the difficulty of the goals: goals A, B and E are almost reached, when the others are not approached within 0.1 meters.

GOAL	20CM	15CM	10CM	BEST COST (METER)	BEST COST (ITERATION)
A	37	56	56	0.02	56
B	26	28	28	0.03	86
C	28	n/a	n/a	0.17	168
D	68	149	n/a	0.11	149
E	20	35	47	0.03	216
F	86	113	113	0.08	113
G	87	113	113	0.08	139

Table 1: Experiment one: number of iterations needed to approach each goal, best attained distance and number of iterations needed to reach it

I want to highlight some interesting behaviors.

Goal C was unachievable (Appendix A). The process converges quickly to a local minimum and CMA-ES stops trying to escapes from it: the mean distance is 0.35 meters for the last 50 iterations.

Goal A was early reached (after 56 iterations), but the cost function does not fall. Such attitude is surprising because, when a good movement is found, the robot

should explore around it and, if not able to find a better move, the covariance matrix will shrink to stay close to the goal (same as goal C). We identify few plausible causes of this stagnation, but still should investigate:

— The unstability of some properties of our setup: the hook may get tangled in the line and the rod fixation may create a slack. Those modify the length of the line and the inclinaison of the rod;

— The initial position is still unstable, and little variations when the hook stops can produce great divergences;

— CMA-ES is vast and may include some mecanisms we are not fully aware of, e. g. trying to escape local minima by increasing exploration range.

In all cases, I wanted to know if the robot was able to replay the best moves or if, for some unknown reason, the result become very different, which will explains why the optimization algorithm becomes uninterested from those movements. For this reason, I wanted to verify the consistency of the best movements for each goal. I selected the top-five moves and replayed them fivefold:

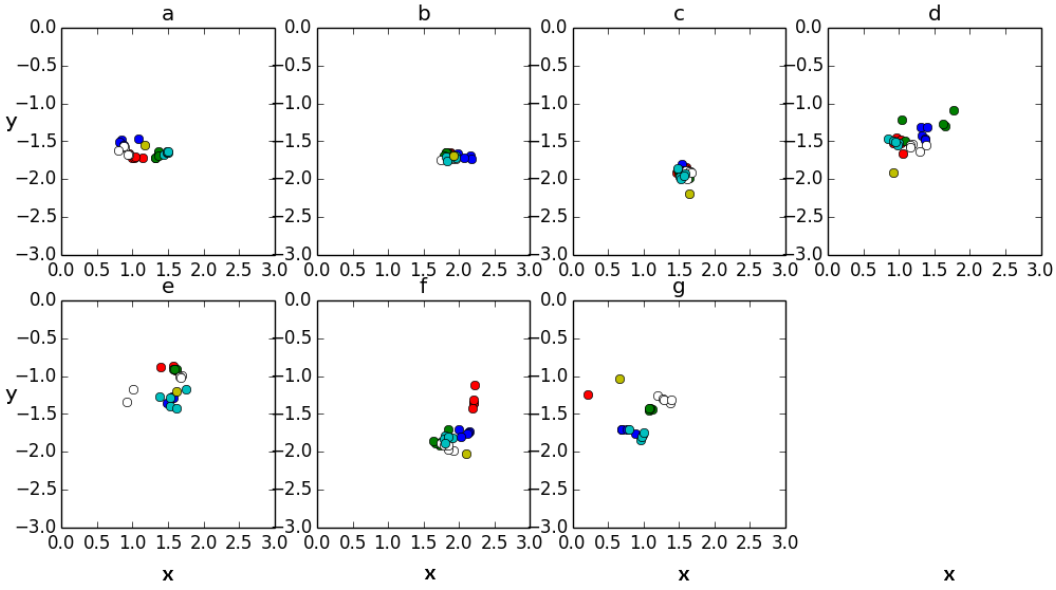


Figure 14: Distribution of the five best moves for each goal. Each panel displays results fir a particular one : each goal is played fivefold and has a specific color. The goals are yellow.

Figure 14 gives an overall picture of the distribution around goal I got. The mean distance for each goal are displayed in table 2.

GOALS	A	B	C	D	E	F	G
MEAN DISTANCE (METERS)	0.27	0.09	0.31	0.56	0.26	0.41	0.69

Table 2: Mean distance (meters) for the five best moves of each goal.

This table shows that only the best moves of the easiest goals are reproductible. B is the more stable one, which can be correlated with the fact the associated cost

function decreases more than for the others (Appendix A). Figure 15 depicts the progression of CMA-ES algorithm for goal B; the oldest points are transparent. We can observe that, apart the movements that does not move the hook, the clusters become increasingly close to the goal. As a result, we can say the sluggishness of the costs evaluation plot is mainly caused by the commands that do not change the hook position.

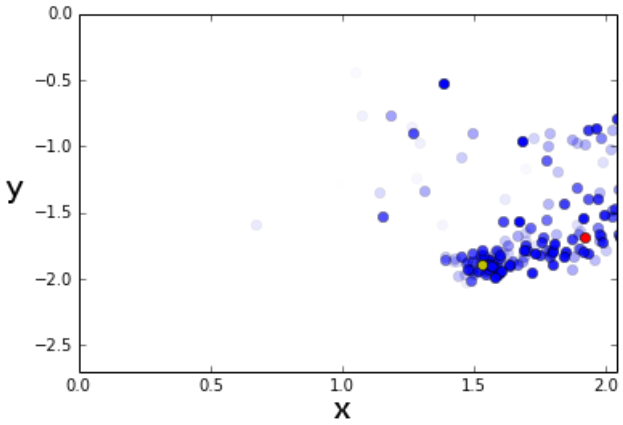


Figure 15: Progression of CMA-ES algorithm. Oldest samples are transparent blue. Goal B is red and the initial position is yellow.

6 CONCLUSION

During these fruitful weeks, I discovered the stimulating field of developmental robotics. I have modeled and simulated a robot learning how to manipulate a fishing rod. Such simulation allowed me to apply an optimization algorithm on a large dataset and predict the main lines of the robot behavior. Then I applied this optimization strategy to the Baxter robot equiped with a line and a hook. It generates trajectories through dynamic movement primitives and perceive the hook final position with an Optitrack system.

The results show that some movement are more reproductible than others in term of the produced effects. Time needed to reach a goal using the optimization procedure seems to be correlated with the ease of reachings and the reproductibility.

I have set all the foundations needed for the upcoming experiments, described in the last section, that I will conduct next weeks to bypass the limitations I faced.

6.1 *Limitations*

The carried experiment shows a few limits.

First, I do not record the initial position for each single move, although it has an huge influence on the hook position (Fig. 10). Currently, I am not able to evaluate its stability.

Second, I need to investigate more on the reasons why the distance towards the goal does not diminish even when it was reached once. I exposed my hypothesis in section 5.2.

Finally, I do not use Explauto, the Flowers' library for autonomous exploration[9], with Baxter, since I was only interested in optimization rather than exploration.

However, studying exploration was the initial aim of my internship, and I want to benefit from the two remaining weeks to apply exploration strategies.

6.2 *Further experiments*

To override those restraints, I have a few experiments I would like to carry out before the end of my internship.

First, I want to compare earlier results with those in case the robot retains all the effects associated with every command. The purpose of this comparison is to see how prior knowledge can affect the time needed to reach a goal. We could for instance compare the performances according to different orderings of the goals. Studies on autonomous exploration and curiosity([6], [7], [8]) show that some orders may ameliorate the learning performance. Thus, I want to apply exploration strategies embedded in Explauto. In this experiment, there will be no specific goal and the robot will explore by itself the whole reachable area.

Another test I should perform is the motor analysis in order to find some structuration of the learning. However, since the motor space is highly dimensionnal, such study will be hard to conduct.

REFERENCES

[1] http://flowers.inria.fr/.

[2] http://www.rethinkrobotics.com/.

[3] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[4] Gilbert Gede, Dale L. Peterson, Angadh S. Nanjangud, Jason K. Moore, and Mont Hubbard. Constrained multibody dynamics with python: From symbolic equation generation to publication. volume 7B, 2013.

[5] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[6] Manuel Lopes and Luis Montesano. Active Learning for Autonomous Intelligent Agents: Exploration, Curiosity, and Interaction. *ArXiv e-prints*, March 2014.

[7] Clément Moulin-Frier, Sao Mai Nguyen, and Pierre-Yves Oudeyer. Self-organization of early vocal development in infants and machines: the role of intrinsic motivation. *Frontiers in Psychology*, January 2014.

[8] Clément Moulin-Frier and Pierre-Yves Oudeyer. Exploration strategies in developmental robotics: a unified probabilistic framework. *Development and Learning and Epigenetic Robotics (ICDL)*, August.

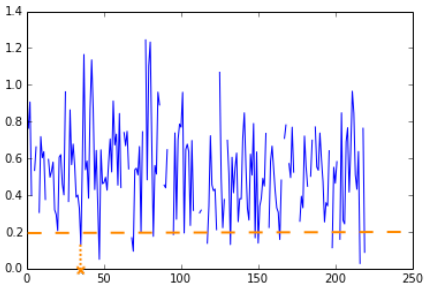
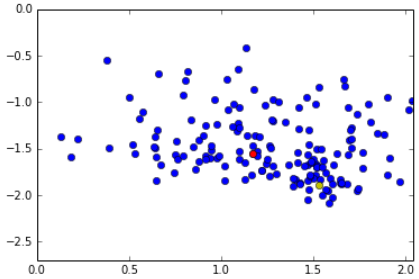
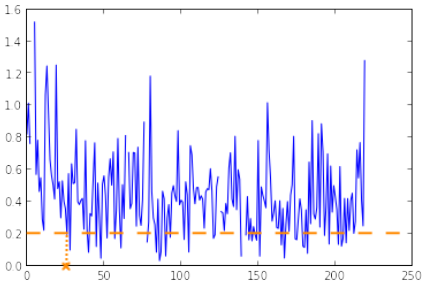
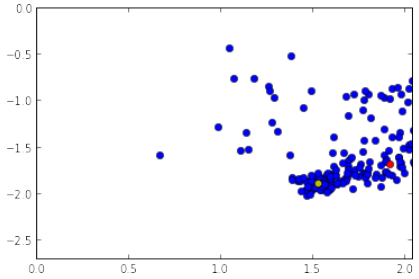
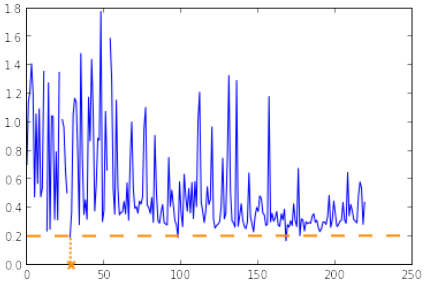
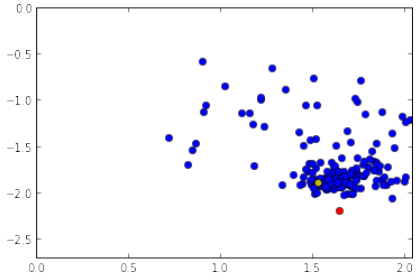
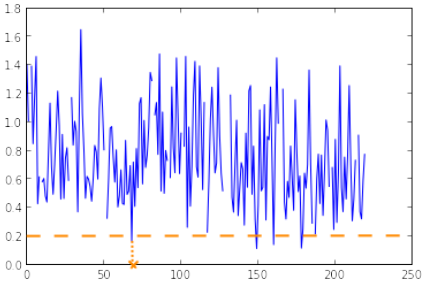
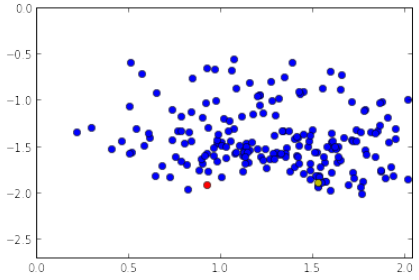
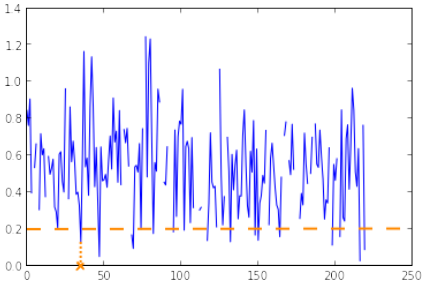
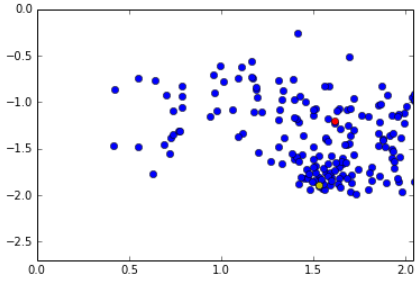
[9] Clément Moulin-Frier and Pierre Rouanet. https://github.com/flowersteam/explauto.

[10] Sao Mai Nguyen and Pierre-Yves Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, 36(3):273–294, March 2014.

[11] Freek Stulp. https://github.com/stulp/dmpbbo.

[12] Freek Stulp and Pierre-Yves Oudeyer. Emergent proximo-distal maturation through adaptive exploration. In *International Conference on Development and Learning (ICDL)*, 2012. **Paper of Excellence Award**.

A RESULTS OF EXPERIMENT ONE

GOAL	COST EVOLUTION (TIME, COSTS)	REACHED POINTS (X,Y)
A		
B		
C		
D		
E		

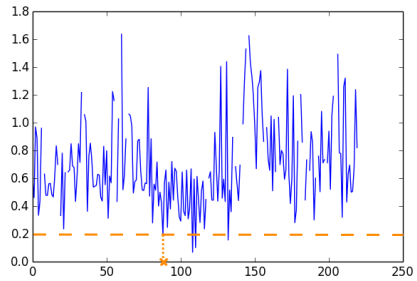
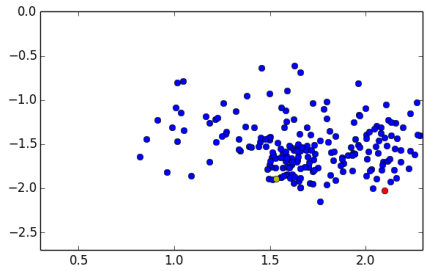
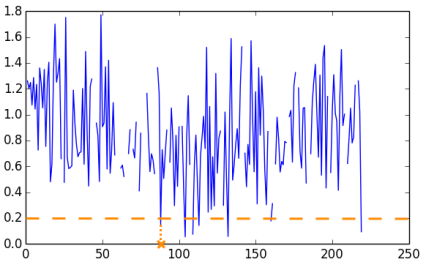
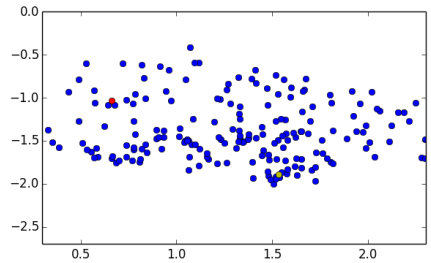
GOAL	COST EVOLUTION (TIME, COSTS)	REACHED POINTS (X,Y)
F		
G		

Table 3: Experiment one: comparison of costs evolution and reached points for each goal. The goals are red and the initial position are yellow. For each one, 220 movements are generated. Orange line represent a cost of 0.2 meters.