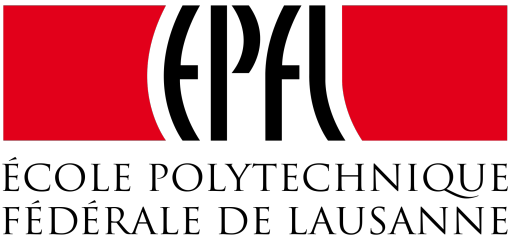


ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Semester project
Microengineering department

Adaptive Human-Robot Interaction: From Human Intention to Motion Adaptation Using Parameterized Dynamical Systems

Autumn semester at **LASA** - EPFL

By :
Antoine Laurens

Assistant :
Mahdi Khoramshahi

Professor:
Aude Billard

Contents

Chapter 1

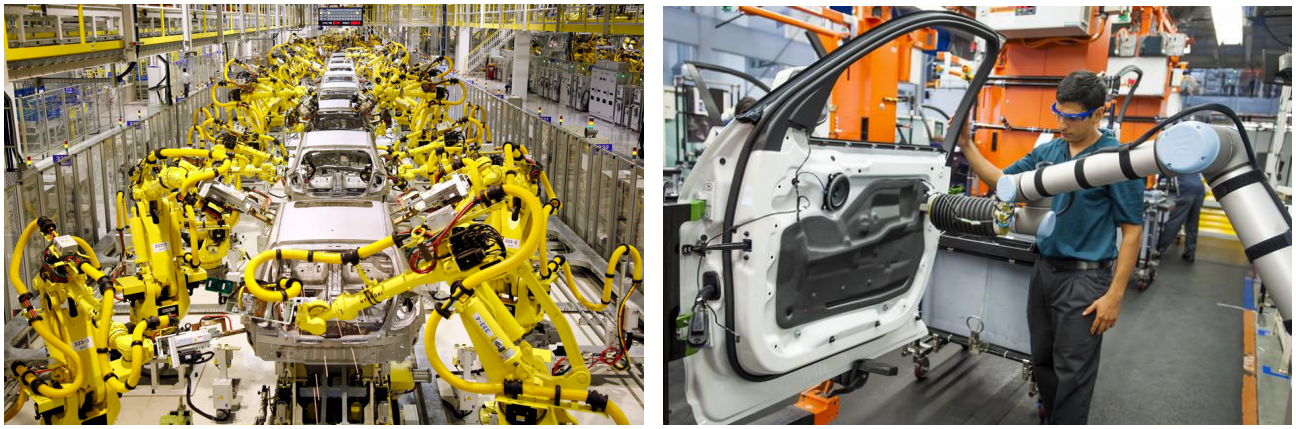
Introduction

1.1 Objectives and motivations of the project

Robots are mainly here to assist us with tasks that are repetitive and burdensome. Up to now, they have mainly been used in industry on large production lines where the robots accomplish in many cases a specific task. Fig.1a shows a production line with robots operating on car frames gives an example of a fully automatized production. These applications require high safety as a human could easily get hurt by one of the moving arms. Therefore, human interaction is usually limited to an operator controlling and configuring the robot through a control panel and at a safe distance to avoid all harm. However, robots are to set to become available for a much larger range of applications of which are included:

- Rehabilitation robotics that can, for example, allow people with disabilities to walk again.
- Service robots to assist people in their daily lives.
- Highly modular robots for small to medium size industry where one robot should accomplish a large variety of tasks. An example is shown in Fig.1b

These three applications all require an understanding of user intention in order to adapt the motion; for example, a highly adaptive robot that would be used for a wide range of applications in a small industry needs to understand for which job it is needed for now and adapt its behavior accordingly. Different approaches exist to endow robots with such capability. Table 1.1 summarize three general categories of solutions with their pros and cons to modify a robotic task. Among these three solutions, two of them have added-hardware which can be a burden to transport or to learn how to use them. The goal is for the user to communicate its desired behavior in an intuitive way without needing to learn technical details for multiple hours. For this reason, our focus will be on the third method; i.e., direct physical interaction where the robot adapts its motion according to the intentional forces of the human user.



(a) (b)

Figure 1: (a) Industrial Robots on a car production line [1] (b) Human working on a car with the help of a robot [2]

Solution	Pros	Cons	Requirements
Control Panel	• High precision • Large adaptation possibilities	• Unintuitive	• Extra hardware
Remote control	• Good for low degree of freedom (DOF) robots	• Unintuitive for high DOF robots such as the Kuka arm • Limited Precision	• Extra hardware
Physical interaction	• Highly intuitive • No added hardware • Large adaptation possibilities	• Limited Precision	• Safety • Compliant interaction

Table 1.1: Different general approaches for task modification of robots

Machine learning and control theory provided us with a variety of techniques to teach our robots how to perform such tasks [3]. For instance, encoding tasks (a set of desirable motions) using dynamical system [4] provides us with stable and smooth motion generation and consequently passive interaction with environment [5]. However, the ability of robots to adapt their tasks to their environment or to the intention of their human-user is limited. In this project, we focus on motion-adaptation of robot task to the intention of the human-user thought the physical interaction where the motions are generated by a dynamical system. A recent attempt has been done to propose an adaptive mechanism for dynamical systems in a previous semester project which was only limited to simulation [6]. In this project, we provide an implementation of such adaptive mechanism considering the physical interaction between a human and a real robot (i.e., Kuka LWR 4+, a 7-DOF robotic arm). Moreover, we provide solutions for issues that are arising from the real-world interaction such as uncertainties and imprecision in control loop.

1.2 Previous semester project and challenges to overcome

This project is done in continuation of a previous semester ([6]). In that project, a method has been proposed for adapting the robot's behavior to the intention of a human-user (details in Section 2.3.2). This method is implemented and tested in simulation, and well-documented. The goal is now to take this method and implement it in reality. Going from simulation to reality comes with two main challenges. The first one is the fact that, in simulation, the human performing the motion on the robot is simulated. In reality, human intention is noisy and possibly inconsistent. Thus, it is is a difficult task to simulate human behavior. Simple approaches such as adding noise to a deterministic set of parameters that models the human intention are yet unrealistic. The second challenge comes from the fact that in simulation, the desired output is known exactly and we can easily determine metrics to measure the error between the desired output and the real output. In reality, however, the desired output is determined by the human's desires and therefore there is no absolute truth on the desired output and we can not measure an error to evaluate different approaches. To evaluate properly this problem, the robot should be tested by a set of people. We must submit subjects to different adaptation methods so that they can evaluate subjectively which one has adapted best to their intentions. However, this would require the setup of an experimental protocol as well as the approval from an ethics board. This work is outside of the scope of this project and therefore, there will be no rigorously prepared human testing of the adaptation methods.

Chapter 2

Robot control

2.1 General robot control

In this chapter, we present our mathematical formulation for the control of robot (using DS-based impedance control), motion generation (using parameterized dynamical systems), and motion adaptation (using proposed gradient-based methods).
Let's consider the general dynamics of a N-DOF robotic manipulator as follows.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u_i + u_{ext} \tag{2.1}$$

where $q \in \mathbb{R}^N$ is the joint configuration, $u_i \in \mathbb{R}^N$ is the input torque and $u_{ext} \in \mathbb{R}^N$ is the external torques applied to the robot (including human interation forces). $M \in \mathbb{R}^{N\times N}$ denotes the mass matrix, $C \in \mathbb{R}^{N\times N}$ and $G \in \mathbb{R}^N$ accounts for centrifugal forces and and gravity. Moreover, let's consider the position of end-effector of the robot for the task-space as $x \in \mathbb{R}^6$ where the Jacobin matrix ($J \in \mathbb{R}^{6\times N}$) is as follows.

$$\dot{x} = J(q)\dot{q} \tag{2.2}$$

Therefore, the control of the robot in the task-space can be formulated as

$$\bar{M}(x_r)\ddot{x}_r + \bar{C}(x_r,\dot{x}_r)\dot{x}_r + \bar{G}(x_r) = F_c + F_{ext} \tag{2.3}$$

where $\bar{M}, \bar{C}$ and $\bar{G}$ are respectively M, C and G transformed from the joint space to the task space. $x_r \in \mathbb{R}^6$ is the real position of the end-effector in the task space. $F_c \in \mathbb{R}^6$ is the force applied to the end-effector by the controller and $F_{ext} \in \mathbb{R}^6$ are the external forces applied to the robot. Given a control force for the end-effector of the robot $F_c \in \mathbb{R}^6$, we can compute the torque commands for the joint as follows:

$$u_i = J^T(q)F_c \tag{2.4}$$

The general scheme of a robot controller is composed of a motion planner (that computes the desired motion based on the current state of the robot) and a controller (that executes the desired motion with a satisfactory behavior such as stability, optimally, or convergence behavior). In this project, we consider a velocity-based impedance controller as proposed in [5] and illustrated in Fig.2. This controller can be expressed as

$$F_{imp} = -D(\dot{x}_r - \dot{x}_d) \tag{2.5}$$

$$F_{inv} = \bar{C}(x_r,\dot{x}_r)\dot{x}_r + \bar{G}(x_r) \tag{2.6}$$

$$F_c = F_{imp} + F_{inv} \tag{2.7}$$

where $F_{imp} \in \mathbb{R}^6$ is the outptut of the impedance controller used to move the robot, $D \in \mathbb{R}^{6\times 6}$ is a Diagonal matrix used as the gain for the impedance control and $\dot{x}_d \in \mathbb{R}^6$ is the desired velocity computed with equation 2.9. $F_{inv} \in \mathbb{R}^6$ is the force required to compensate for both the centrifugal force and gravity.
The combination of equations 2.3 and 2.7 leads to

$$\bar{M}(x_r)\ddot{x}_r = -D(\dot{x}_r - \dot{x}_d) + F_{ext} \tag{2.8}$$

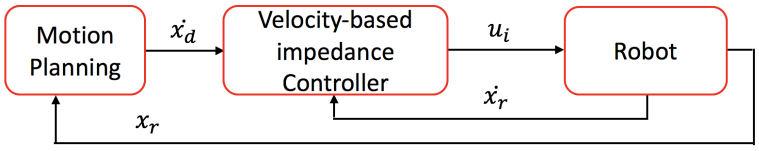


Figure 2: Velocity-based impedance controller used in this project

2.2 Dynamical systems (DS)

Thus far, the motion planner is considered as a general mapping from the state of the robot to the desired motion. In our approach, we use time-indented dynamical systems as our motion planner as follows:

$$\dot{x}_d = f(x_r;\Theta) \tag{2.9}$$

where f is a globally asymptotically stable dynamical system under a Lyapunov function (for an attractor or a limit cycle). Moreover, Θ is a set of parameters for f. Applying this to equation 2.8 leads to:

$$\bar{M}(x_r)\ddot{x}_r = -D(\dot{x}_r - f(x_r;\Theta)) + F_{ext} \tag{2.10}$$

In the framework of learning by demonstration, [7] proposed a method to learn such dynamical system in a stable manner. Motion planning using dynamical system is a practical approach since it

- generates smooth and stable motion for the robot.
- reacts to or re-plans according to changes in the state of robot.
- generalizes the desired behavior in the face of unseen contexts.
- provides a solid foundation for passive and compliant interaction.
- has potential for adaptability and re-learning.

2.2.1 Cyclic motion

As a simple example, let's consider a cyclic and circular motions generated by the following dynamical system.

$$\dot{\vec{x}} = \begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = f_c(\vec{x}) = \begin{pmatrix}\dot{r}cos(\phi) - r\dot{\phi}sin(\phi)\\ \dot{r}sin(\phi) + r\dot{\phi}cos(\phi)\end{pmatrix} \tag{2.11}$$

where:

$$r = \sqrt{x^2 + y^2} \qquad \dot{r} = -\alpha(r - r_0)$$
$$\phi = \text{atan2}(y, x) \qquad \dot{\phi} = \omega$$

Where $\vec{x} \in \mathbb{R}^2$ is the state of our system, $r \in \mathbb{R}$ and $\phi \in \mathbb{R}$ are the coordinates in the polar space. $\omega \in \mathbb{R}$ is the desired angular velocity, $\alpha \in \mathbb{R}$ is the desired radial velocity and $r_0 \in \mathbb{R}$ is the desired radius of rotation. The phase portrait of this dynamical system is illustrated in Fig. 3a where all motion converges to the desired limit cycle. Moreover, Fig.3b shows how such dynamical react to sudden perturbations of the system. This characteristic allows the robot to be compliant and enables human interaction which is crucial in this work.

As explained in [6] a dynamical system can be modified in the following way:

$$\dot{x}_d = T^{-1} f(T(x_r))$$

where $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is a diffeomorphism that preserves the smoothness of the resulted dynamical system. In our example of a cyclic motion we can have for example:

$$T(\vec{x}) = H\vec{x} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \vec{x}$$

where $H \in \mathbb{R}^{2\times 2}$ is an invertible linear transformation which preserves the stability of the system. This particular example allows to scale the 2 semi-axes of the circle to transform it into an ellipse. An illustration of this is given in Fig.3c with $a = 1.5$, $b = 0.8$. The linearity allows for the combination of a series of transformations. By combing

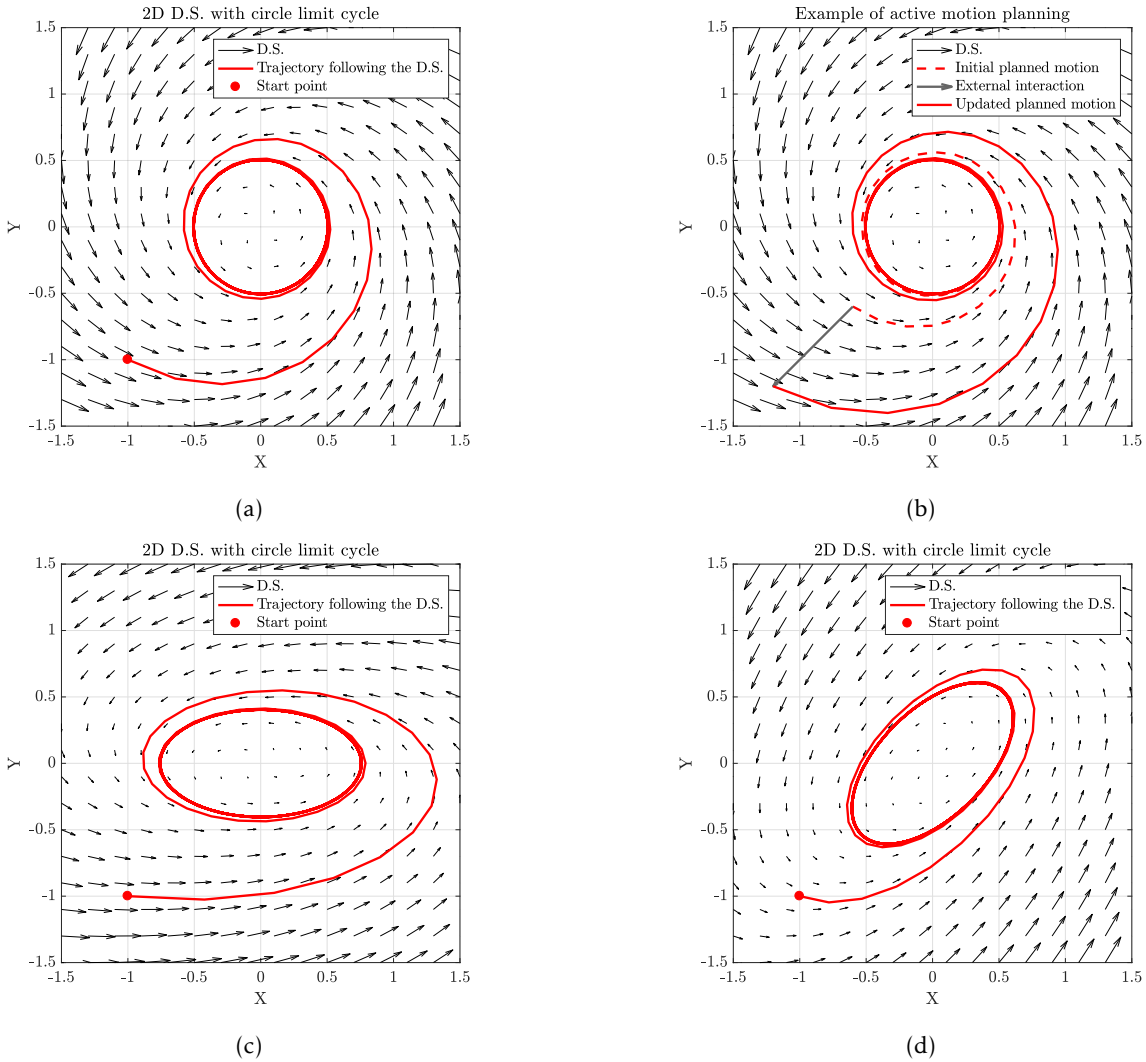


Figure 3: (a) Example of circular motion generated with a dynamical system. (b) Example of dynamical system re-planing it's motion after the state being changed. (c) Example of circular motion transformed into an ellipsoid movement through a linear transformation with the semi-axes scale by a factor 1.5 and 0.8 respectively. (d) Ellipsoid movement described in (c) rotated with a angle of 45^o

the previously described transformation with a linear rotation we can orientate the cycle.The rotation matrix of an angle $\alpha \in [-90^o\ 90^o]$ is given by:

$$R = \begin{pmatrix} cos(\alpha) & sin(\alpha) \\ -sin(\alpha) & cos(\alpha) \end{pmatrix}$$

An example of the 2 previously described transformations combined is illustrated in Fig.3d with $a = 1.5,\ b = 0.8$ and $\alpha = 45^o$. Finally, the center of rotation can also be changed in a cyclic motion changing where the rotation is executed. For this, we use the following function:

$$\dot{\vec{x}}_d = f(\vec{x}_r - \vec{x}_t)$$

where $\vec{x}_t \in \mathbb{R}^2$ is the target of the center of rotation. This principle is illustrated in the following Section.

2.2.2 Repetitive pick-and-place

Another example of dynamical system, that can lead to interesting applications such as pick-and-place, is a simple attractor. This dynamical system can be formulated as:

$$\dot{\vec{x}}_d = A\vec{x}_r \tag{2.12}$$

where A is a negative-definite matrix. An illustration of such a dynamical system is given in Fig.4a with:

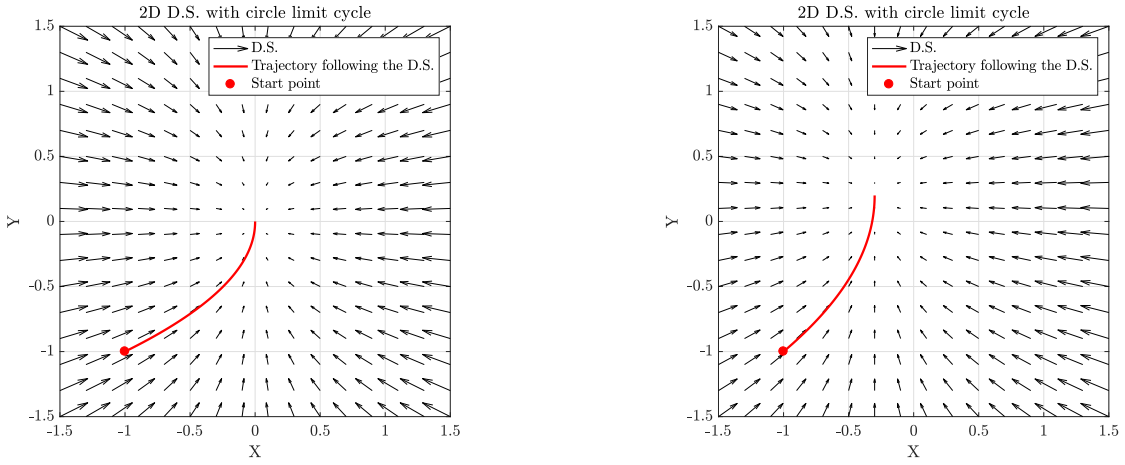
$$A = \begin{pmatrix} -1 & 0 \\ 0 & -1/2 \end{pmatrix}$$

This attractor can easily be modified in order to attract the system to any point with the following function:

$$\dot{\vec{x}}_d = A(\vec{x}_r - \vec{x}_t)$$

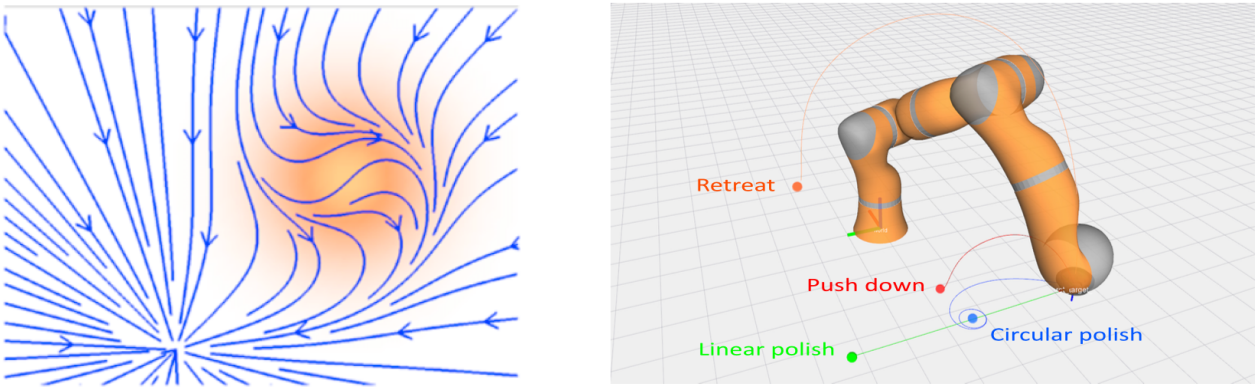
where $\vec{x}_t$ is the target we want to reach. Fig.4b shows an example of such a dynamical system with $\vec{x}_t = \begin{pmatrix} -0.3 \\ 0.2 \end{pmatrix}$.

A pick and place type of motions can be considered as a set of different attractors that, in turn, attract the robot to either pick, place, or go through a via point. Pick-and-place is widely used in industry, thus, providing an adaptive mechanism that allows the human-user to change the pick and place locations online would be a valuable tool in small industries.



(a) (b)

Figure 4: (a) Example of an attractor dynamical system. (b) Example of a shifted attractor
P



(a) (b)

Figure 5: (a) Example of a local adaptation(orange area) of a dynamical system[8]. (b) Different dynamical systems used to perform task adaptation in [9]

2.3 Motion adaption using dynamical systems

2.3.1 State of the art

Although numerous works have been done to perform motion-adaptation, here we only focus on those using dynamical systems. In [8], authors investigated local adaptation of dynamical systems for grasping objects depending on the shape of the object while avoiding obstacles of unknown size and shapes. The illustration of the local adaptation of a dynamical system is shown in Fig.5a. For instance, a local modulation can be applied to the DS to avoid an obstacle that would be along the initial trajectories.

A second example of adaptation of dynamical systems is illustrated in [9]. In this work, the adaptation is not directly performed on a single DS, but on a linear combination of several ones. The goal is to switch between different dynamical systems based on human interaction. A general motion planner measures the likelihood of different DS. When the human wants to switch motion, it can safely grab the robot and move it according to the desired motion. The planer measures the likelihood increase of the desired motion and smoothly switches from one stable DS to the other. The different DS used in the mentioned work are illustrated in Fig.5b.

In contrast to previous works, the adaptation in this work is done 1) globally (not just locally) which allows for higher level of generalization and 2) without switching which enables the robot to reshape one specific task according to the intention of the human.

2.3.2 Solution studied in previous semester project

As mentioned in Section 1.2, this work is done in the continuity of a previous semester project that simulated the adaptation of dynamical systems. In this approach, the goal is to adapt the set of parameters Θ upon which the DS depends. For example, in the circular motion described in Section 2.2, the parameter that describes the shape of the trajectory is the r_0 coefficient. By changing this parameter, one can modify the shape of the dynamical system and adapts it to the user's intentions.

In order to achieve this, optimization algorithms were used to minimize the error between the real velocity (which accounts for the human intention since it can be influenced by human interaction in our compliant architecture) and the desired velocity (which is generated by dynamical system given its parameters). This is described by the following cost function:

$$J(x_r, \dot{x}_r; \Theta) = \frac{1}{2} e^T e = \frac{1}{2} |\dot{x}_r - \dot{x}_d|^2 \tag{2.13}$$

where $e = \dot{x}_r - \dot{x}_d$ is the error between the real and the desired velocities, $|.|$ denotes the norm of a vector. In the previous work, the gradient of this function was used in order to update the parameters. The gradient of this function is defined as:

$$\nabla_\Theta J = \frac{\partial J}{\partial \Theta} \tag{2.14}$$

$$= \begin{pmatrix} \frac{\partial J}{\partial \theta_1} & \frac{\partial J}{\partial \theta_2} & \cdots & \frac{\partial J}{\partial \theta_n} \end{pmatrix}^T \tag{2.15}$$

where n is the number of parameters and θ_i is the i^{th} parameter of the dynamical system. To compute the gradient with respect to each parameter, we can write:

$$\frac{\partial J}{\partial \theta_i} = e^T \cdot \frac{\partial e}{\partial \theta_i} \tag{2.16}$$

$$= e^T \cdot \frac{\partial f(x_r;\Theta)}{\partial \theta_i} \tag{2.17}$$

The analytical formulation of $\frac{\partial f(x_r,\Theta)}{\partial \theta_i}$ for nontrivial DS is usually not available. Therefore, we use the centered differences approximation:

$$\frac{\partial J}{\partial \theta_i} = e^T \cdot \frac{f(x_r;\Theta_{\neq i},\theta_i + h) - f(x_r;\Theta_{\neq i},\theta_i - h)}{2h} \tag{2.18}$$

where h is the step size of the gradient, and $\Theta_{\neq i}$ denotes all other parameters (except θ_i) that are kept at their previous value.

The results of this project show that the best adaptation algorithms depends on the motion that is being adapted. In a fast evolving motion, such as point to point faster algorithms perform better. However, in the case of cyclic motions, the adaptation must be slow to avoid instability issues. The first motion that will be adapted to in reality will be the cyclic motion and therefore the adaptation method that will be implemented is the gradient descent method. In this method, the parameters are updated proportionally to the gradient of the cost function. This gives us:

$$\dot{\Theta} = -\epsilon \nabla_\Theta J \tag{2.19}$$

where ϵ is the rate of adaptation of the gradient descent which leads us to the following discrete formulation:

$$\theta_{new} = \theta_{old} - \epsilon e^T \cdot \frac{\partial f(x_r;\Theta)}{\partial \theta_i} \tag{2.20}$$

Therefore there are 2 hyperparameters for this method, the gradient step h and the rate of adaptation ϵ.

The control loop shown in Fig.2 can be modified to include this adaptation step as illustrated by Fig.6. An adaptation/optimization block is added to the system that takes as input the real velocity of the robot $\dot{x}_r$ and the desired velocity of the robot $\dot{x}_d$. The human is also added to the control loop where it applies an unknown force F_h to the robot in order to demonstrate his/her desired motions.

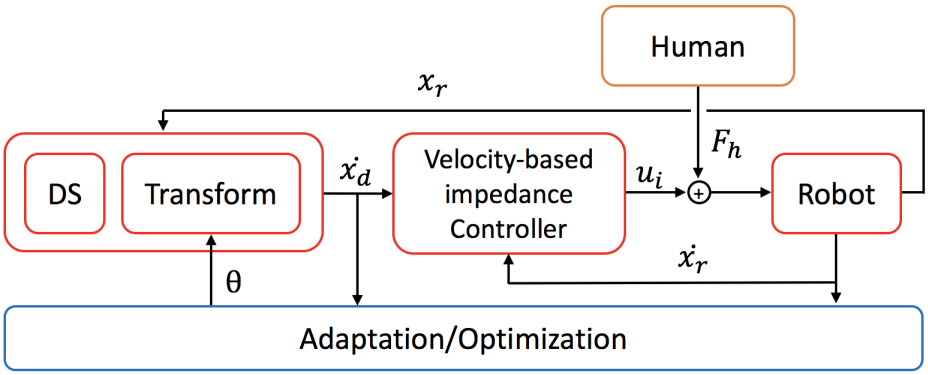


Figure 6: Control loop of an adaptable dynamical system

Chapter 3

Implementation of the adaptation

The adapatation algorithms are implemented and tested on the Kuka lwr 4+ (7-DOF robotic arm) robot illustrated in Fig.7. This robot is equipped with torque sensors allowing us to perform torque control and by extension impedance control allowing for compliant interaction and motion-adaptation.

Figure 7: Kuka robotic arm used to test the adapatation methods

3.1 ROS: Robot Operating System

The implementation of the adaptation methods is done using the Robot Operating System(ROS). This tool is a framework that provides different tools in order to communicate with the robot. Fig.8 shows the organization of a simple structure that uses ROS in order to communicate. A node is an entity that is interested in publishing and receiving messages from other subsystems (for example the robot or the computer controlling the robot). These nodes communicate through channels called topics and on which messages can be published, nodes can either subscribe(be notified every time a new message is published on the topic) or advertise(publish messages on the topic). The subscriptions and advertisements are managed by the ROS master unit.

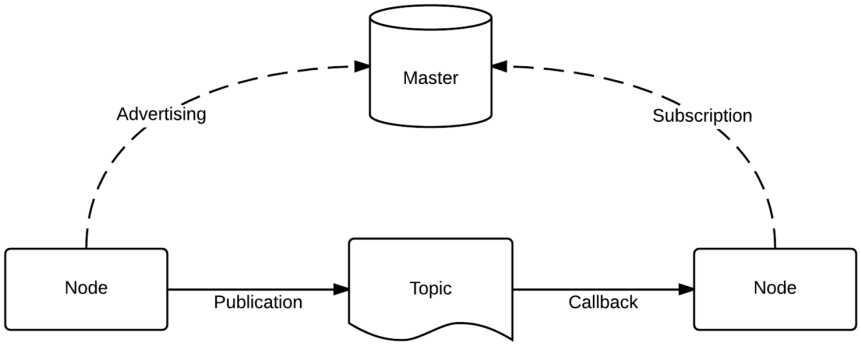


Figure 8: Organisation of a simple ROS system

Fig.9 shows the organization of the nodes and topics in this project.We distinguish 3 main nodes:

1. The FRI (`/lwr/lwr_fri` in Fig.9) node which collects information from the robot sensors and applies the desired command to the robot.
2. The rviz node(`/lwr_rviz` in Fig.9) which is a tool to visualize the robot in simulation.
3. The controller node(`/ds1_main_node` in Fig.9) is the one that is coded for this project, it collects the data from the FRI node through different topics and computes the desired velocity depending on the state of the robot and the parameters of the dynamical system.

The ROS package that is implemented in this project is called `adaptive_polishing` and the code for it can be found at [10]. This package is implemented using different packages coded by members of LASA. For controlling the robot, we use the ROS package provided by LASA [11].

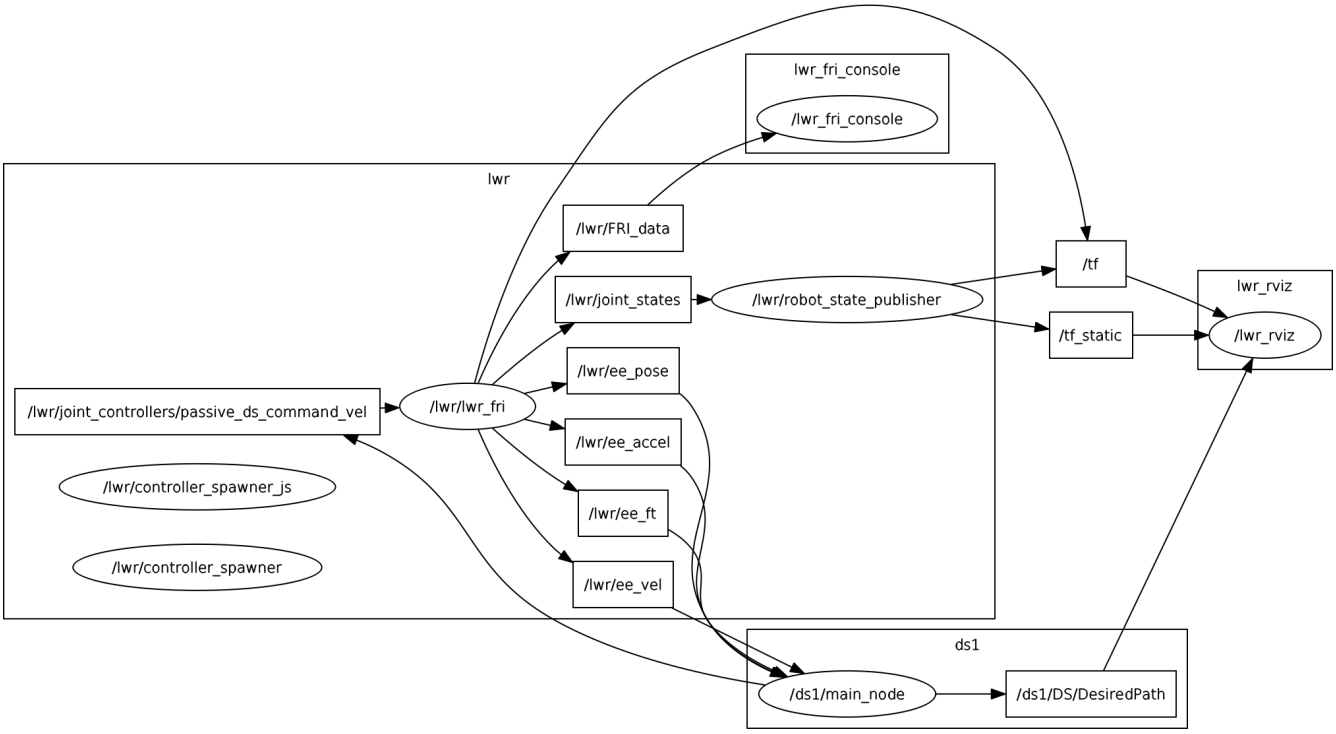


Figure 9: ROS nodes and topic organization in this project

3.2 Experimental setup

The implementation of the adaptation algorithms is done in multiple cases which are described in the following section. The results for every different implementation will be further discussed in chapter 4.

3.2.1 Cyclic motion

The first type of motion that is adapted is the cyclic motion. This continuous motion is simple but can be adapted in many different ways as described in Section 2.2.1. Also, it was well documented in the previous semester project and it was shown that it could be adapted in simulation.

Adaptation over low number of parameters

The first step is to test the adaptation for a lower number of parameters. Since, at each step time, we use two bits of information (errors on x and y axis), the adaptation of a high number of parameters would be slower and requires much richer and less noisy demonstrations. Therefore, for the first experiment, we adapt the center of rotation on the x and y axis using the adaptation law presented in Eq. 2.20.

Adaptation over high number of parameters

Achieving satisfactory results for a lower number of parameters, we investigate the performance of our adaptive mechanism for a higher number of parameters. The added parameters allow the scaling of both semi-axes to form an ellipse as explained in Section 2.2.1. In this experiment, the cost function is the same as in the previous experiment as well as the adaptation method.

Adaptation over multiple points along the trajectory

As detailed in Section 4, the performance of the proposed method is limited for a high number of parameters since information extracted only from one time-step seems to be insufficient for capturing the human intention and therefore adaptation. To improve this, we extended the cost function to include information from the past. The new cost function at time t is computed as:

$$J(x_r, \dot{x}_r, t; \Theta, K) = \frac{1}{2} \sum_{k=0}^{K-1} [\dot{x}_r(t - k\frac{T_s}{K-1}) - f(x_r(t - k\frac{T_s}{K-1}); \Theta)]^2 \tag{3.1}$$

where f is the DS generating the desired velocity, K is the number of data points taken over the past that are uniformly sampled from a time-window with length T_s; see Fig.10a. The idea is to adapt the parameters based on a range of data points along the trajectory in order to prevent one parameter updating to fast and not letting enough time for the other parameters to adapt. The adaptation algorithm used here is the gradient descent method:

$$\Delta\theta_i = -\epsilon \frac{\partial J}{\partial \theta_i} \tag{3.2}$$

$$= -\frac{\epsilon}{K} \sum_{k=0}^{K-1} e^T \cdot \frac{\partial f(x_r(t - k\frac{T_s}{K-1}); \Theta)}{\partial \theta_i} \tag{3.3}$$

We divide by the total number of points such that the gradient does not grow with the number of points.

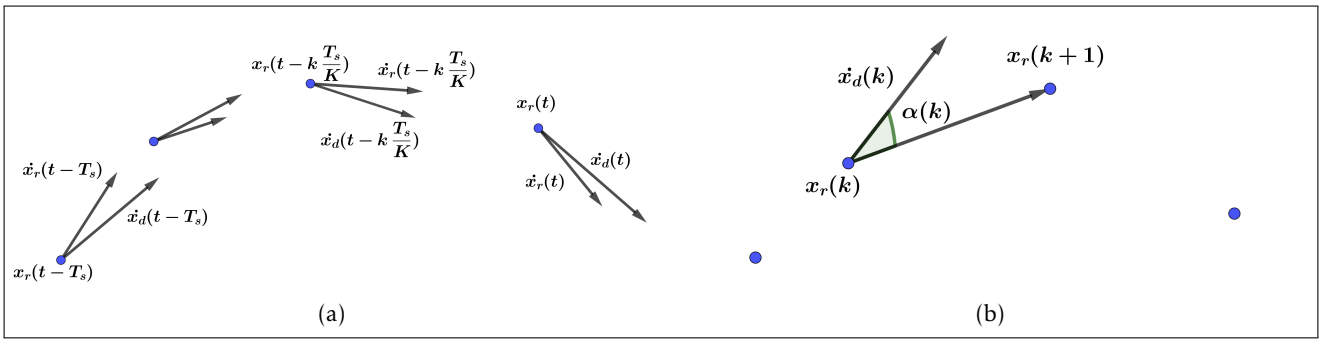


Figure 10: (a) Adaptation over multiple points along the trajectory. (b) Illustration of the alternative cost-function

Alternative cost-function

Due to a bug in the first version of the previously described method an alternative cost function is developed. The bug leads us to believe that the error comes from the fact that inconsistency in the speed norm does not allow to

properly adapt the system. Therefore in order to remove the dependency to the norm the following cost function described at time t step is proposed:

$$J(x_r, t; \Theta, K) = \frac{1}{2} \sum_{k=1}^{K} \alpha(x_r, k, t; \Theta)^2 \tag{3.4}$$

where:

$$\alpha(x_r, k, t; \Theta) = \text{acos}\left(\frac{f(x_r(t - k\frac{T_s}{K-1}); \Theta) \cdot [x_r(t - (k+1)\frac{T_s}{K-1}) - x_r(t - k\frac{T_s}{K-1})]}{|f(x_r(t - k\frac{T_s}{K-1}); \Theta)||x_r(t - (k+1)\frac{T_s}{K-1}) - x_r(t - k\frac{T_s}{K-1})|} \right) \tag{3.5}$$

This cost function is illustrated in Fig.10b. The idea is to minimize the angle α between $\dot{x}_d(k)$ and the direction of the vector $x_r(k+1) - x_r(k)$ over K data points along the trajectory. Similar to previous methods, the gradient of cost is used to update the parameters.

3.2.2 Pick and place motion

Another interesting task for motion-adaptation is the pick-and-place as it is a motion widely used in industry and could, therefore, the adaptation could be applied to many different domains. As mentioned previously a pick and place motion can be seen as a set of different attractors in order to follow a trajectory alternating between a pick location and a place location. In order to adapt these 2 locations, one must adapt the center of attraction of the attractor linked to both positions. This is similar to adapting the center of rotation of the cyclic motion as described previously. Therefore the adaptation algorithm that will be used is exactly the same as the one used in the simple parameter adaptation of the cyclic motion:

$$J(x, \dot{x}_r; \Theta) = \frac{(\dot{x}_r - \dot{x}_d)^2}{2} \tag{3.6}$$

$$= \frac{(\dot{x}_r - f(x_r; \Theta))^2}{2} \tag{3.7}$$

And the adaptation algorithm that will be used is the gradient descent method:

$$\Delta\theta_i = -\epsilon \frac{\partial J}{\partial \theta_i} \tag{3.8}$$

3.2.3 Implementation details

We want the robot to only be adapting when there is an actual human interaction with the robot and therefore we must detect such interactions. In the case of this project, we measure the equivalent force applied to the end-effector and only when the force goes above a certain threshold (experimentally chosen to be 10 N) do we start adapting the motion. This applies to both the pick and place and cycle motion.

When adapting multiple parameters at once, they often do not always have the same order of magnitude. And therefore the gradient step h should be adapted for each parameter depending on the magnitude. In order to keep a single h, when adapting the parameters we apply a normalization on each parameter to have an equivalent value between 0 and 1 and therefore make the step size meaningful.

The pick and place motion being a set of dynamical systems we must be careful at how the adaptation is done. In practice, 3 attractors were used. 2 for pick and place location and 1 attractor as a via-point to go above an imaginary obstacle. Both the pick and place locations are adaptable but not the checkpoint. The attractors can only be adapted when they are currently active. A motion planner node is used to choose which attractor should be active and pauses the other ones; it moves on to the next point when we have reached the location of the current attractor. This can cause a problem because if the user is currently adapting one attractor's location it could happen that the end-effector is near the current attractor and moves on to the next one, effectively stopping

the adaptation half-way through. To prevent this, the force applied to the end-effector must be below a certain threshold. This signal that the human has let go of the robot and is therefore satisfied with the new location for either place or pick. In short, the post-condition for sub-task is to reach the attractor and to have small interaction forces.

Chapter 4

Results

4.1 Cyclic motion

4.1.1 Adaptation over low number of parameters

Adaptation of the center of rotation alone showed very good results, the robot followed the humans desires in a smooth way. Fig.11, 12, 13 show three different moments in the experiment that is done on the robot. The value of the parameters in this case are:

- $\epsilon = 0.001$
- $h = 0.01$

The top left graph shows the norm of the force that is sensed at the end-effector of the robot along with the threshold that indicates the level at which the robot will start updating the parameters. The second plot on the left shows the parameters we are adapting, the coordinates of the center of rotation. The 3^{rd} plot on the left shows the value of the cost function, the error between the real velocity and the desired velocity. Finally, the plot on the right shows the current dynamical system along with the desired path of the robot as well as the real trajectory performed by the robot. We observe a smooth transition from one set of parameters to the other along with a quick reduction of the error whenever the applied force goes over the threshold.

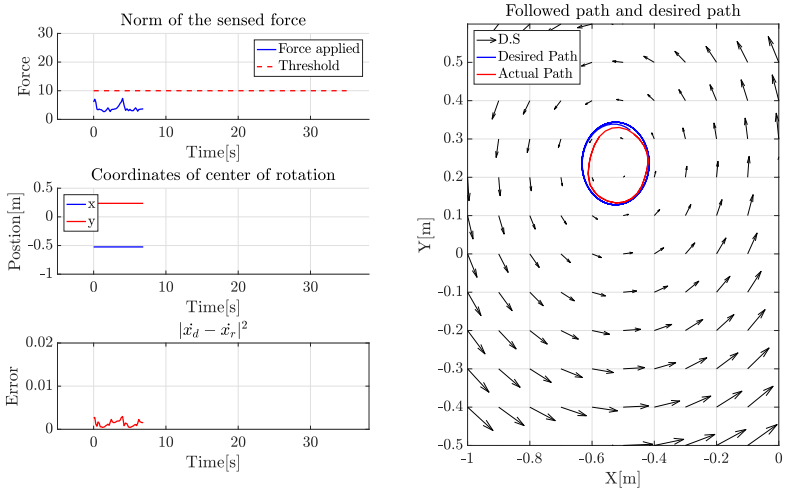


Figure 11: 1^{st} set of parameters in simple adaptation of the center of rotation

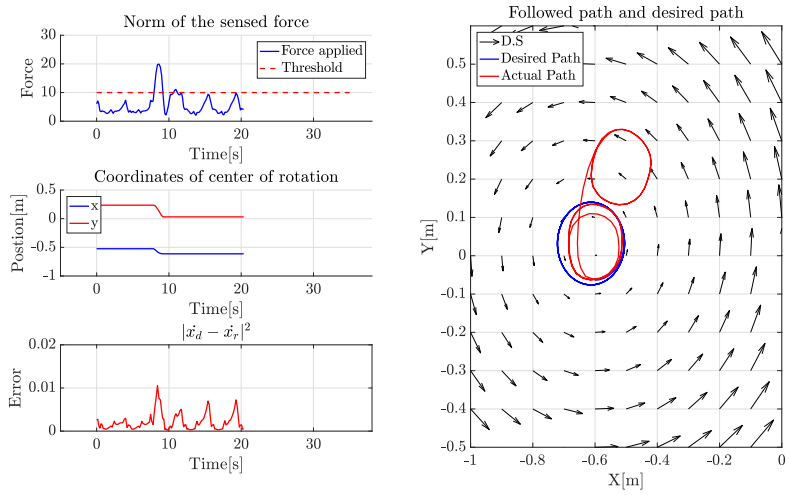


Figure 12: 2^{nd} set of parameters in simple adaptation of the center of rotation

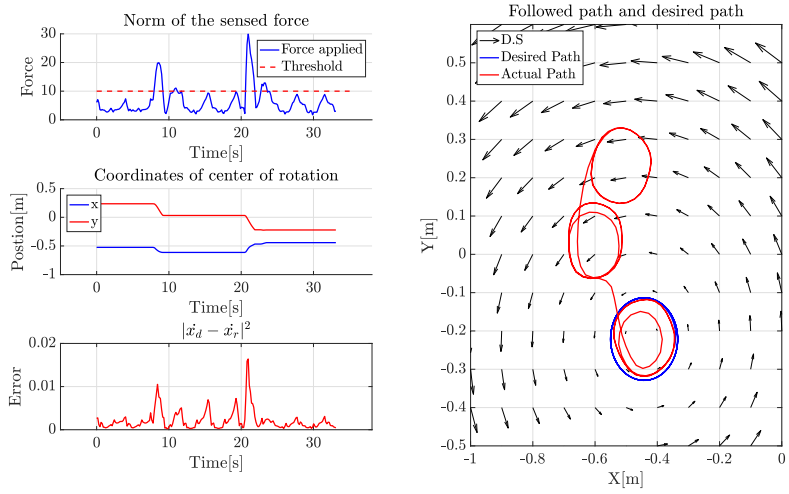


Figure 13: 3^{rd} set of parameters in simple adaptation of the center of rotation

4.1.2 Adaptation over high number of parameters

After successfully adapting the center of rotation of the cyclic motion, other parameters are added and tested. Noticeably the scales of the 2 semi-axes of the ellipse are added in order to modify the shape of our motion. When testing these parameters on there own, the adaptation is possible and the dynamical system adapts smoothly.This is illustrated in Fig.14 for which the parameter are:

- $\epsilon = 0.001$
- $h = 0.01$

The next step is to test all 4 parameters together. This, however, does not work as expected. The adaptation is still happening and everything is still smooth, however, it is very difficult for the human to modify all 4 parameters. The center of rotation is almost the only one adapting during most part of the trial. This is due to the fact that the

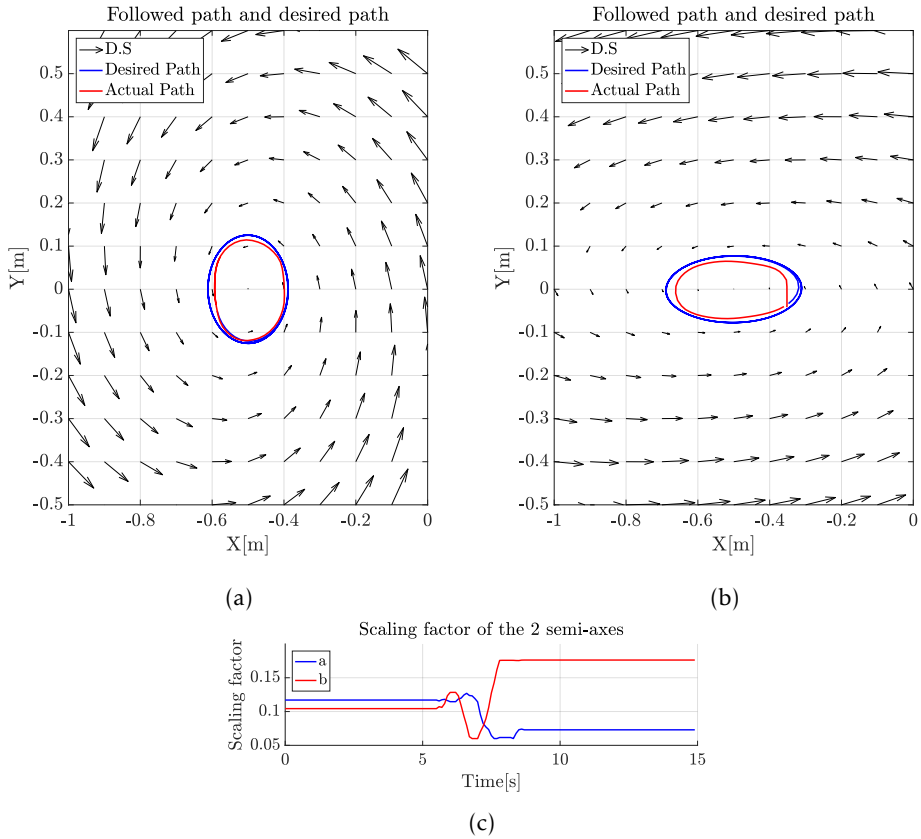


Figure 14: (a) Dynamical system at t = 0 (b) Dynamical system after adaptation at t = 10 (c) Evolution of the parameters during adaptation

center of rotation has a much quicker response to the adaptation and when the human is attempting to show an elliptical motion to the robot, the center of rotation simply moves along the ellipse, barely allowing the dynamical system to adapt the scale of the axes as we can observe in Fig.15. In this case, the parameters are:

- $\epsilon = 0.001$
- $h = 0.01$

4.1.3 Adaptation over multiple points along the trajectory

The goal of the second adaptation function is to prevent the center of rotation from simply following the trajectory when the user is attempting to demonstrate an elliptical motion. Both the center of rotation and the axes of the ellipse should adapt in order for all the data points to "fit" correctly to the proposed dynamical system. This new cost function allows the system to adapt efficiently to the demonstrated motion. When setting the parameter T_s (the length of the time window) to a low value we observe that the center of rotation is the only one adapting by following the movement shown by the human. When setting $T_s = 5$ seconds the dynamical system is able to rapidly find a fit to the demonstrated motion. Fig.16. In this example the parameters chosen are:

- $\epsilon = 0.1$
- $h = 0.01$
- $T_s = 5$ seconds

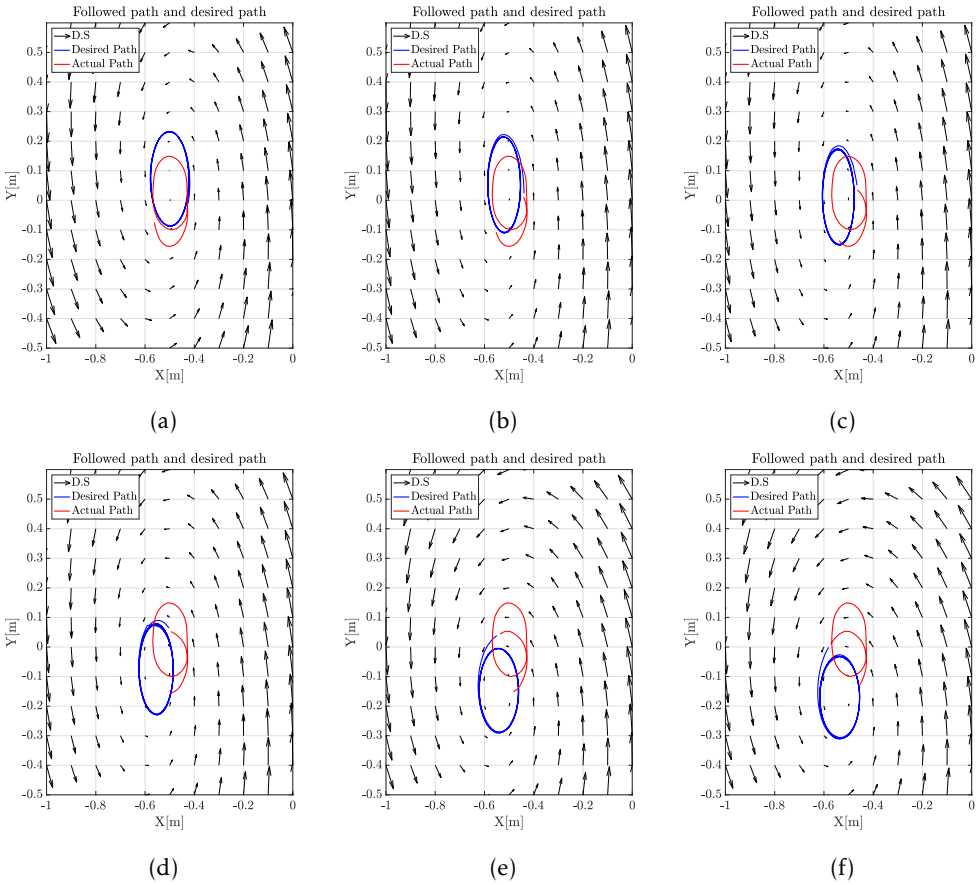


Figure 15: Evolution of the dynamical system with the first cost cost function while adapting multiple parameters, we can see the center of rotation moving along the trajectory

- $K = 10$ points

In Fig.17 we show the adaptation of the cyclic motion with the angle of the rotation as added parameter bring the total number of parameters to adapt up to 5. The set of parameters used are the same as for the previous example.

4.1.4 Alternative cost-function

Due to lack of time the alternative cost function is not tuned sufficiently. The initial results show that the adaptation isn't very consistent and doesn't always manage to converge to the ideal solution. Fig.18 shows an example in which the adaptation is successful, but later on when a new motion was shown isn't able to converge to the desired solution. In this case, the parameters used are:

- $\epsilon = 0.001$
- $h = 0.01$
- $T_s = 5$ seconds
- $K = 20$ points

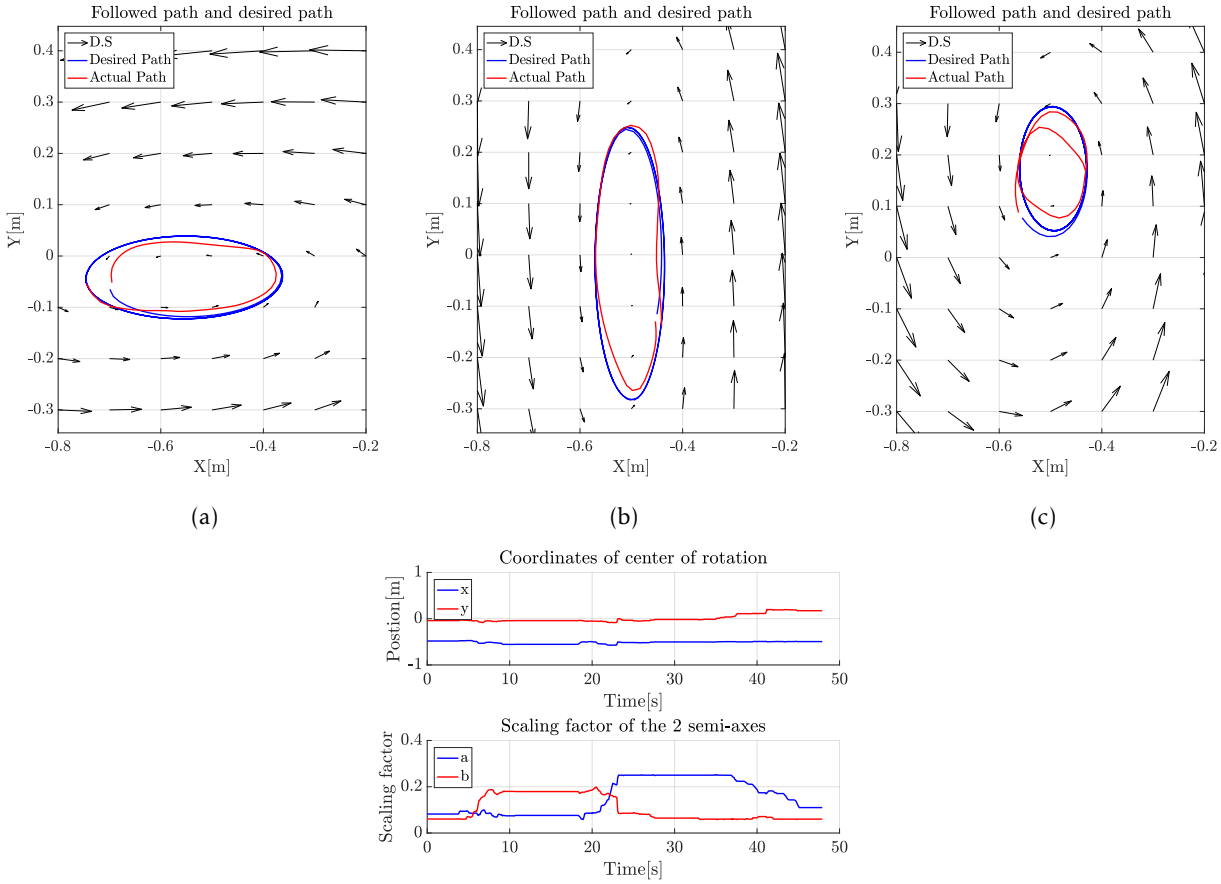


Figure 16: (a) & (b) & (c) Dynamical system at respectively $t = 15s$, $t = 30s$ and $t = 47s$ (d) Evolution of the parameters during adaptation

4.2 Pick and place motion

Adaptation of the attractors during the repetitive pick and place motion show that it is possible to adapt each attractor independently and therefore modify the pick and place locations with ease and very intuitively. However, it is noticed that there is sometimes not enough time to grab the robot before it reaches the attractor and therefore are not able to update it. The set of parameters that work for this adaptation are:

- $\epsilon = 0.001$
- $h = 0.01$

For a demonstration of the pick and place adaptation see [12].

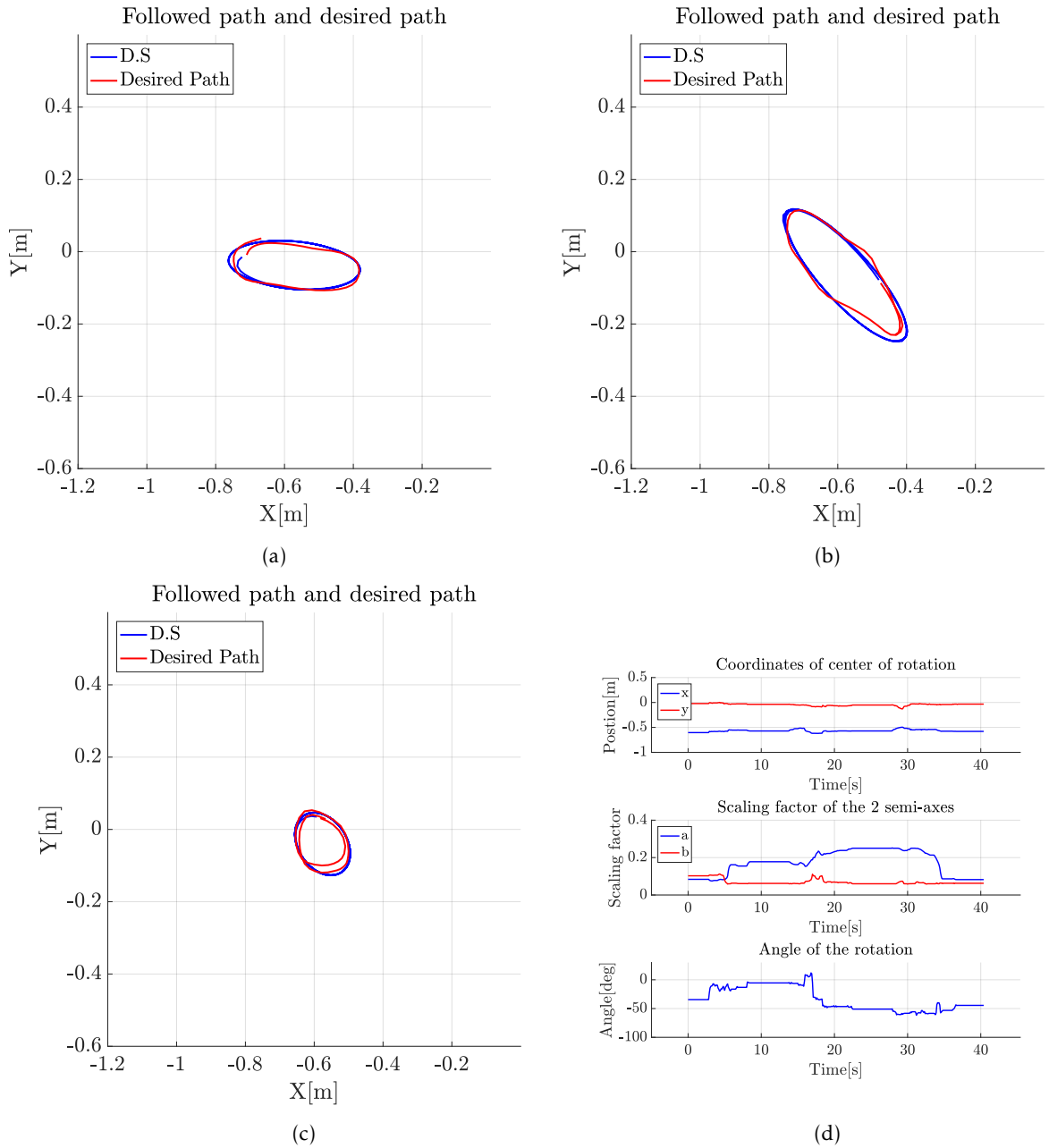


Figure 17: (a) & (b) & (c) Dynamical system at respectively $t = 10s$, $t = 25s$ and $t = 39s$ (d) Evolution of the parameters during adaptation

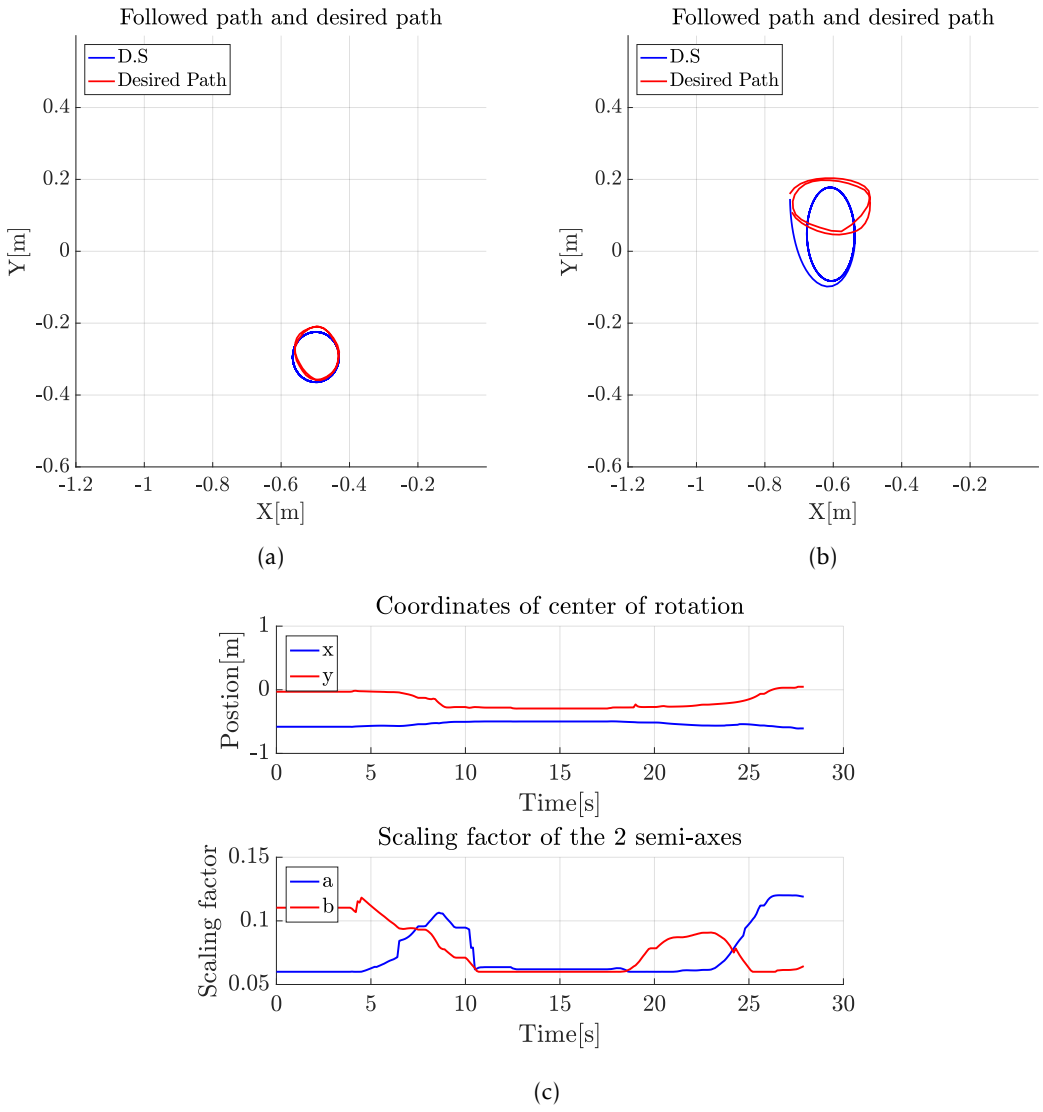


Figure 18: (a) Example of when the alternative cost function allows for good adaptation (t = 15) (b) Example of when the alternative cost function doesn't allows for good adaptation (t = 26) (c) Evolution of the parameters during adaptation

Chapter 5

Conclusion and future work

Conclusion

This project shows that motion-adaptation using parametric dynamical system is not only possible but it could be very effective for compliant human-robot interaction. We showed by sampling few data points form human behavior (captured by the real velocity) and motion-generator (the desired velocity), our simple adaptation mechanism is able to comply to the intention of the human. We observed that quality of the adaptation depends on the choice of the cost function, type of the motion, and hyperparameters. When adapting on a low number of parameters, a single data point can be taken in order to minimize the cost function. However, with the increase in the number of parameters to adapt, we need more information order to properly adapt the dynamical system to the demonstrated motion. The cost function 3.1 derived in this project allows such an adaptation by optimizing over a set of points along the trajectory instead of only the most recent point as proposed by [6]. With this modified cost function, we are able to successfully adapt cyclic dynamical systems with a set of 5 different parameters:

- The center of rotation of the rotation in both x and y.
- The angle of the orientation of the rotation of the cyclic motion.
- The scale of both semi-axes of the ellipse.

We have also shown that adaptation of a simple attractor is possible and that by extension this can allow to adapt more complex motions such as a repetitive pick and place which could be valuable for industrial applications.

Future work

Further attempts are required to reach a systematic approach for tuning the parameters to obtain a working solution based on the cost function 3.4. Also, a rigorous tuning of the hyper-parameters should be done in order to find the best working solution. Intuition on how the parameters were chosen for this project and where to start to optimize them further is the following:

- ϵ: When this parameter goes to 0, there is no more adaptation as the change in the parameters is directly proportional to the value of $epsilon$. However, if this parameter is set too high, then the parameters do not update smoothly and the dynamical system varies extremely quickly creating a very jerky motion that is very unpleasant to the user.
- h: The gradient step must be sufficiently small to have a precise approximation for the derivative of the cost function. However, since we are dealing with noisy observation, reducing this parameter would result in noisy approximation for the derivative.
- T_s and K: These two parameters go hand in hand. In order to adapt properly the dynamical system, the adaptation must be done on a set of points that both characterizes well the DS (from the desired velocity) and captures well the human intention (from the real velocity). Therefore, T_s must be sufficiently large to capture a sufficient portion of the cycle and K must allow the sampling of enough data points. On the other hand, increasing the number of data point increases leas to issues with the computational effort and including contradictory demonstrations form the past that needs to be forgotten.

Finally, future work can be done to improve the compliant behavior of the robot during the adaptation. The idea would be to lower the impedance gains (D in equation 2.5), thus, reducing the stiffness of the robot. This would bring 2 advantages: 1) it would reduce the human effort required to demonstrate his/her desired behavior, 2) the variation of the compliance serve as a haptic communication where human can infer the state of robot (e.g., the

robot's confidence with respect to the parameters that it is optimizing). However, the variation of the compliance should be done in a smooth and stable manner to avoid creating jerky and sudden motion while in contact with a human. Also, it is required to find a method where the impedance gains are updated based on the quality of the adaptation; e.g., based the value of the cost function. For example, if the human is unable to demonstrate a consist motion that the adaptation mechanism cannot converge to, the robot's stiffness should be decreased even further in order to allow the human to demonstrate the motion in a better way.

Antoine Laurens

Date and Signature

Bibliography

[1] *Industrial production line*. URL: http://imgur.com/CF8tEct.

[2] *Robot assisting human in industry*. URL: https://www.manufacturingtomorrow.com/article/2016/02/collaborative-robots-working-in-manufacturing/7672/.

[3] Aude Billard et al. "Robot programming by demonstration". In: *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[4] S Mohammad Khansari-Zadeh and Aude Billard. "Learning stable nonlinear dynamical systems with gaussian mixture models". In: *IEEE Transactions on Robotics* 27.5 (2011), pp. 943–957.

[5] Klas Kronander and Aude Billard. "Passive interaction control with dynamical systems". In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 106–113.

[6] Thomas Triquet, Mahdi Khoramshahi, and Aude Billard. *Task-adaptation for assistive robotics using switching dynamical systems*. Tech. rep. LASA - EPFL, 2017, p. 28. URL: https://www.dropbox.com/s/97hrwhsg2gz9c8g/Triquet_TaskAdaptationForAssitiveRoboticsUsingSwitchingDS.pdf?dl=0.

[7] Nicolas Sommer, Klas Kronander, and Aude Billard. "Learning Externally Modulated Dynamical Systems". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2017). URL: https://infoscience.epfl.ch/record/229361 (visited on 12/29/2017).

[8] K. Kronander, M. Khansari, and A. Billard. "Incremental motion learning with locally modulated dynamical systems". In: *Robotics and Autonomous Systems* 70 (Aug. 2015), pp. 52–62. ISSN: 0921-8890. DOI: 10.1016/j.robot.2015.03.010. URL: http://www.sciencedirect.com/science/article/pii/S0921889015000822.

[9] M. Khoramshahi and A. Billard. "A Dynamical System Approach to Task-Adaptation in Physical Human-Robot Interaction". In: *under review in Autonomous Robots* (2018).

[10] *GitHub repository of the adaptive polishing package*. URL: https://github.com/alaurens/adaptive_polishing.

[11] *GitHub repository of the ROS package to control KUKA LWR*. URL: https://github.com/epfl-lasa/kuka-lwr-ros.

[12] *Demonstration of pick and place adaptation*. URL: https://youtu.be/HLfZW6rPxYc.

The demonstrations of this work can be viewed here:

https://youtu.be/qIcOAtVMNgE

https://youtu.be/TGwNkSEMm0M