Master 2 - Semester Project

Learning Coupled Dynamical Systems for Adaptive Robots Coordination

Zeid Karim
karim.zeid@epfl.ch

Supervised by

Mahdi Khoramshahi
Prof. Aude Billard

Learning Algorithm and Systems Laboratory (LASA)

September 7, 2017

Contents

Chapter 1

Introduction

Improvement in robotics design (i.e., precision, energy-efficiency, cost) has transformed the industrial sector. From 2001 to 2015, the worldwide annual supply of industrial robots has increased from 81k to 254k [1]. With a number of robots increasing, the coordination between them is an important matter in order to not disturb the continuity of a production line.

In such industrial settings, conveyor belts are primarily used as a connection between different stations. They usually carry objects to different stations for different purposes such as assembling, packaging and manipulation. A good coordination between the conveyor belt and the station is necessary to obtain the expected results. Indeed, methods such as SEDS (Stable Estimator of Dynamical Systems)[2] are able to generate a dynamical system given a set of demonstrations. A dynamical system (DS) is a function that related the position vector to the velocity vector of an entity such as a robotic arm. SEDS[2] is a method that treats the target to reach as an attractor. However, the dynamic of a moving attractor is different from a fixed attractor. Despite SEDS being stable for a fixed attractor, the convergence is no longer guaranteed for a moving attractor. In order to solve this problem coupling the DS of the arm and the conveyor belt is necessary 1.1.

In [3], a method is presented to learn the coupling between different dynamical systems. However the coupling in this sense is different from ours as it looks for a way to synchronize different DS. For example if one DS is the arm and the second the hand, the method will make the fingers start moving only when the arm is close to the target. It doesn't in itself change the shape of the DS of the hand but it changes locally where in the DS the fingers should be if the arm is at a certain position from the target.

In this work, we use SEDS as a motion generator for a robotic arm that adapts its behavior to the velocity of a conveyer belt. We do not take into account the dynamics of the robotic arm. Therefore, we consider that the robotic arm can follow any path. We look at different methods in order for a robotic arm to adapt to the varying speed of the conveyor belt. The first solution is to scale the norm of the DS in order to adapt to different speeds. Then we influence not only the norm of the DS but the shape of the DS itself by making some arrangement on the demonstrations. Last, we modify SEDS in order to include an external parameter that is dependent on the other DS we are trying to couple with.

1.1 Problem statement

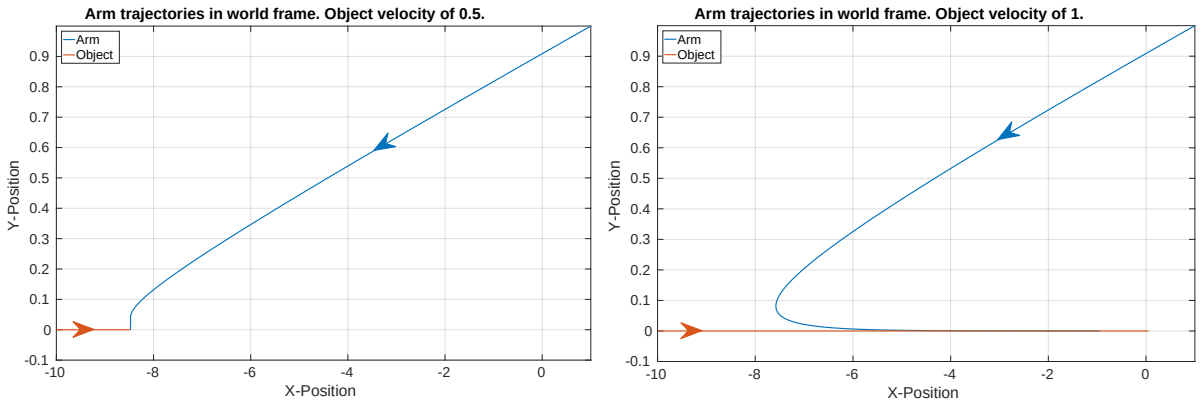
In our work, we neglect the dynamics and kinematics of the robot and focus on the end-effector trajectory of the arm. Therefore we assume that our robot can follow any given path. We start by attributing a DS to the robotic arm and give the conveyor belt a constant speed v^c_{const}. In this case we do not couple them in order to show the necessity of coupling for our problem. The DS of the arm is as follows:

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\end{bmatrix} = \begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}x(t)\\ y(t)\end{bmatrix} \tag{1.1}$$

In Figure 1.1a, by using a low velocity, the robotic arm is able to intercept the object on the conveyor belt. The arm starts at the position $(1, 1)$ and uses the DS enunciated in equation 1.1. The object on the conveyor belt is at $(-10, 0)$ and goes towards the right with a velocity $v^c_{const} = 0.5$.

In Figure 1.1b, we used the same DS used in 1.1. However the speed of the conveyor belt is now of $v^c_{const} = 1$, a greater speed than previously. As one can see the robotic arm using the DS in 1.1 is unable to adapt to a different speed. This means that the arm will go to infinity without catching the object. The system is unstable despite the stable DS given to the arm.

We have shown the necessity of coupling the dynamical system in order to improve the odds of intercepting the object. By using the velocity of the conveyor belt we would like to influence the DS used for the arm in order for it to adapt.



(a) Velocity of object $s = 0.5$ (b) Velocity of object $s = 1$

Figure 1.1: Uncoupled system trajectories in world frame using different speed v^c_{const} for the conveyor belt.

For example with a speed of $v^c_{const} = 1$ we clearly see that the arm is unable to intercept the object but one can change the DS of the arm in order to cope with the new velocity. By giving a new DS to the arm given by:

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\end{bmatrix} = \begin{bmatrix}-0.1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}x(t)\\ y(t)\end{bmatrix} \tag{1.2}$$

Table 1.1: Table of average speed of arm with $v^c_{const=1}$ and different DS.

	Average speed of arm trajectory with $v^c_{const} = 1$
DS in 1.1	0.0630
DS in 1.2	0.0096

In Figure 1.2, one can see the simulation with a fast moving object $v^c_{const} = 1$ and a new DS in equation 1.2 given to the arm. One can notice that the shape changes and the overall velocity is actually lower than previously (Table 1.1). This can be explained intuitively, indeed by lowering the speed along x the robot has more time to go down in y because the arm is not moving toward the object as fast as previously.

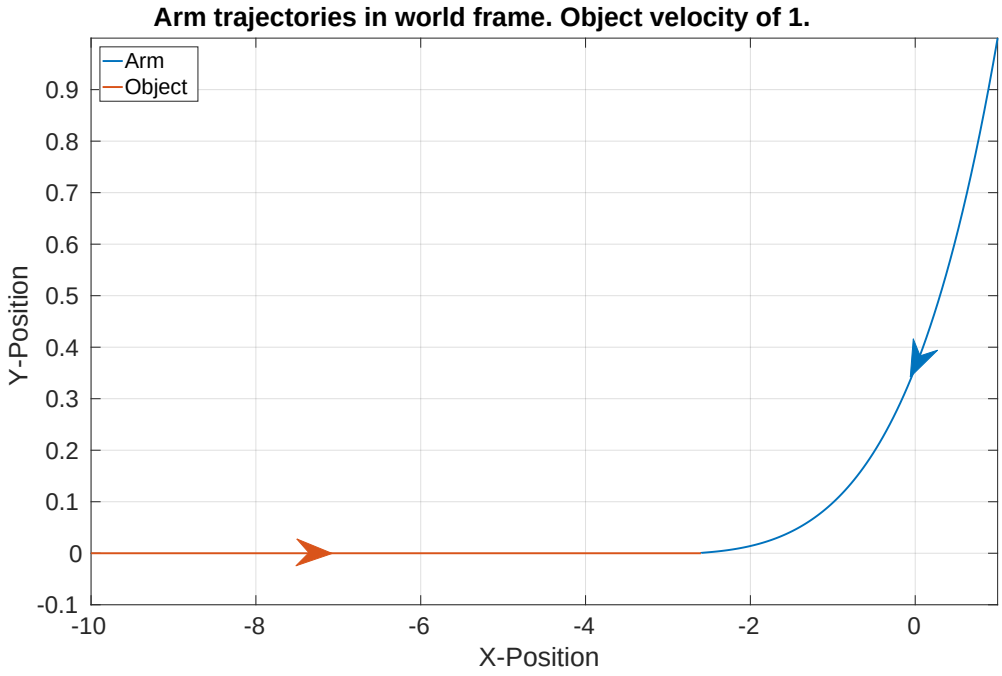


Figure 1.2: Uncoupled system with the arm using the DS in 1.2 and conveyor with speed s=1.

In the following the advantages of coupling, the objectives of this paper and the hypothesis are presented.

Advantages: One can see the benefice of having a system were the DS changes in function of an exterior signal (e.g., the velocity of the target-point). Not only are we now able to catch a fast moving object but the robot does it with an overall slower velocity. This method improves the capabilities of the robot because it can now adapt to a faster object while not increasing its own velocity. Furthermore, according to [4], *"Push-grasping is a robust way of grasping objects under uncertainty. It is a straight motion of the hand parallel to the pushing surface along a certain direction, followed by closing the fingers."*. One can imagine changing the way the robot grasp the object on the conveyor belt depending on the uncertainty therefore the speed of the conveyor belt.

Objectives: The goal is to adapt the DS of the arm for any kind of velocity v^c of the conveyor belt. Different approaches are presented throughout this report. The scaling of the DS,

the arrangement on the demonstrations and SEDS learning the coupling are the three methods presented.

Hypothesis: One can intercept an object on a conveyor belt by coupling a conveyor belt with a robotic arm.

Figure 1.3: A robot executing the required task or maybe it's me, who knows...

1.2 Related works

1.2.1 SEDS (Stable Estimator of Dynamical Systems)

The DS is defined using a set of demonstrations and the SEDS algorithm. Throughout this paper SEDS is used to generate a DS we can do analysis on. The following section reminds the reader of the theory related to SEDS, for further information please refer to [2].

1.2.1.a Mathematical framework of SEDS[2]

The algorithm presented in [2], estimates a DS based on demonstrations while ensuring that the solution is globally stable. A first order autonomous Ordinary Differential Equation (ODE) is given by:

$$\dot{\xi} = f(\xi) \tag{1.3}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is a non linear differentiable function with a single equilibrium point at $\dot{\xi}^* = f(\xi) = 0$ and θ is the set of parameters of f. Given a set of N demonstrations $\{\xi^{t,n}, \dot{\xi^{t,n}}\}_{t=0,n=1}^{T^n,N}$ the SEDS (Stable Estimator Dynamical System) estimates the function f using a mixture of Gaussian functions. The estimate of f denoted $\hat{\text{f}}$ is estimated by $k = 1..K$ gaussians, with π^k the priors, μ^k the means and Σ^k the covariance matrices[2].

$$\mu^k = \begin{pmatrix} \mu_\xi^k \\ \mu_{\dot{\xi}}^k \end{pmatrix} \quad \& \quad \Sigma^k = \begin{pmatrix} \Sigma_\xi^k & \Sigma_{\xi\dot{\xi}}^k \\ \Sigma_{\dot{\xi}\xi}^k & \Sigma_{\dot{\xi}}^k \end{pmatrix} \tag{1.4}$$

Given the demonstrations the probability density function is given by[2]:

$$\mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n}; \theta) = \sum_{k=1}^{K} \mathcal{P}(k) P(\xi^{t,n}, \dot{\xi}^{t,n}|k) \quad \begin{cases} \forall n \in 1..N \\ t \in 0..T^n > 0 \end{cases} \tag{1.5}$$

where $\mathcal{P}(k) = \pi^k$ is the prior and $P(\xi^{t,n}, \dot{\xi}^{t,n}|k)$ is the condition probability density function given by[2]:

$$P(\xi^{t,n}, \dot{\xi}^{t,n}|k) = \mathcal{N}(\xi^{t,n}, \dot{\xi}^{t,n}; \mu^k, \Sigma^k) = \frac{1}{\sqrt{(2\pi)^(2d)|\Sigma|}} - \exp^{\frac{1}{2}([\xi^{t,n}, \dot{\xi}^{t,n}] - \mu^k)^T (\Sigma^k)^{-1} ([\xi^{t,n}, \dot{\xi}^{t,n}] - \mu^k)} \tag{1.6}$$

Taking the posterior mean estimate of $\mathcal{P}(\dot{\xi}|\xi)$[2]:

$$\dot{\xi} = \sum_{k=1}^{K} \frac{\mathcal{P}(k)\mathcal{P}(\xi|k)}{\sum_{i=1}^{K} \mathcal{P}(i)\mathcal{P}(\xi|i)} (\mu_{\dot{\xi}}^k + \Sigma_{\dot{\xi}\xi}^k (\Sigma_\xi^k)^{-1} (\xi - \mu_\xi^k)) \tag{1.7}$$

The notation of (1.7) can be simplified. [2] defined variables as:

$$\begin{cases} A^k = \Sigma_{\dot{\xi}\xi}^k (\Sigma_\xi^k)^{-1} \\ b^k = \mu_{\dot{\xi}}^k - A^k \mu_\xi^k \\ h^k(\xi) = \sum_{k=1}^{K} \frac{\mathcal{P}(k)\mathcal{P}(\xi|k)}{\sum_{i=1}^{K} \mathcal{P}(i)\mathcal{P}(\xi|i)} \end{cases} \tag{1.8}$$

Substituting (1.8) into (1.7) yields[2]:

$$\dot{\xi} = \hat{f}(\xi) = \sum_{k=1}^{K} h^k(\xi)(A^k \xi + b^k) \tag{1.9}$$

1.2.1.b Stability

In order to be stable the model has to satisfy the following theorem [2]:
Theorem 1 *Assume that the state trajectory evolves according to* (1.9)*. Then the function described by* (1.9) *is globally asymptotically stable at the target* ξ^* *in* $\mathbb{R}^d$ *if:*

$$\begin{cases} b^k = -A^k \xi^* \\ A^k + (A^k)^T \prec 0 \end{cases} \qquad \forall k = 1..K \tag{1.10}$$

where $(A^k)^T$ *is the transpose of* A^k*, and* $\prec 0$ *refers to the negative definiteness of a matrix.*

1.2.2 Implementation of SEDS on Matlab

The SEDS source code in Matlab is available at [5]. In our version, one can choose the number of demonstrations and draw them using a mouse directly on a Matlab figure. The code has been adapted from [6]. Examples of the program running are shown in Figure 1.4.

The program needs different demonstrations with the same number of points. Therefore a Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) is computed on the data extracted from the drawings to get the program running.

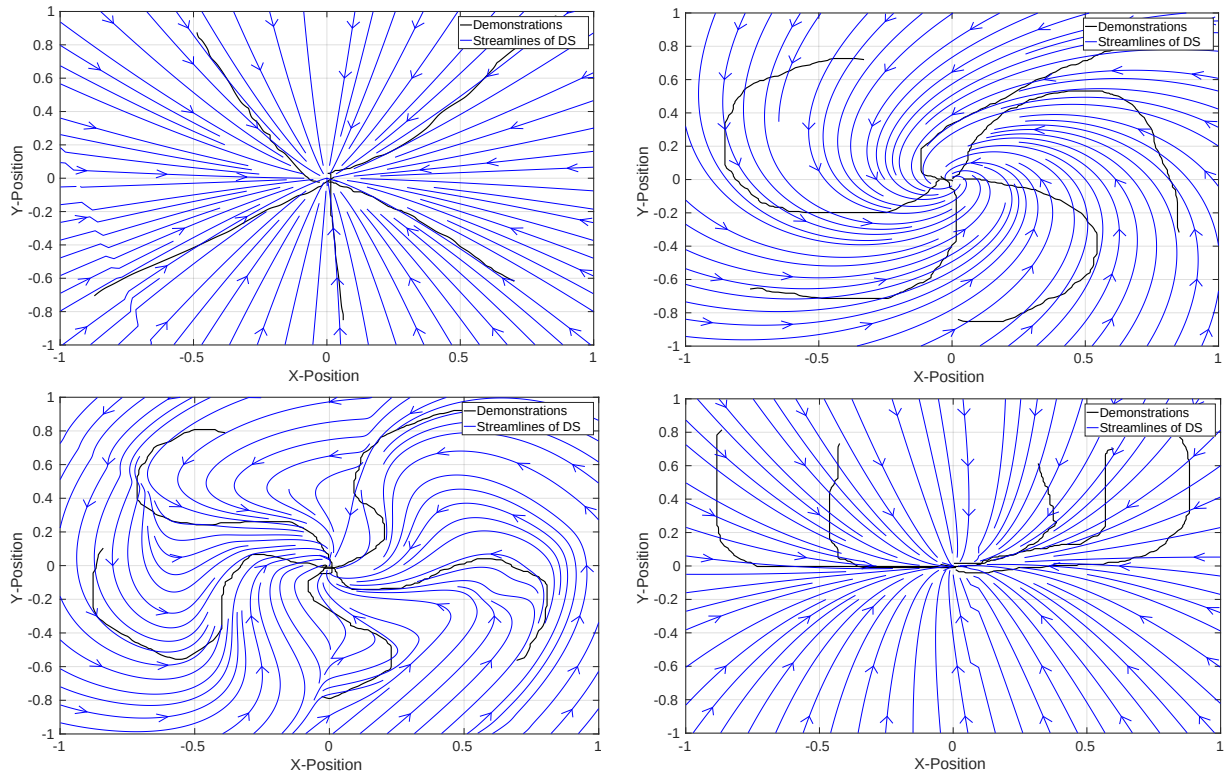


Figure 1.4: Examples of the application of SEDS on Matlab.

1.2.2.a Simple application: intercepting moving objects

The theory and the implementation on Matlab enables us to run a simulation of the system. In Figure 1.5, one can see the case where the conveyor belt is not moving. The robotic arm is following

the lines given by some DS based on demonstrations that we input using the mouse of a computer.

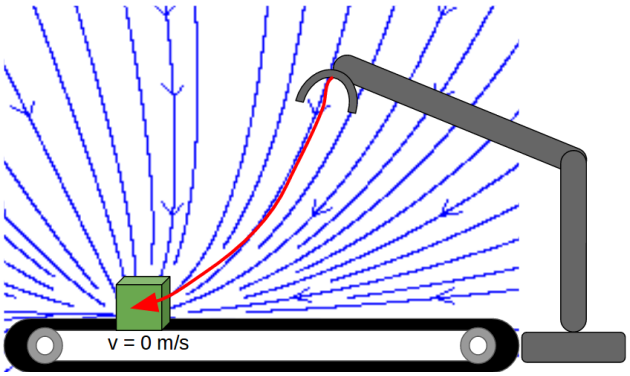


Figure 1.5: Regime with a static object.

In Figure 1.6, we represent the case where the conveyor belt is moving at a certain velocity. One can notice that after the time dt the conveyor belt follows another line along the DS because the object is moving towards it at the same time.

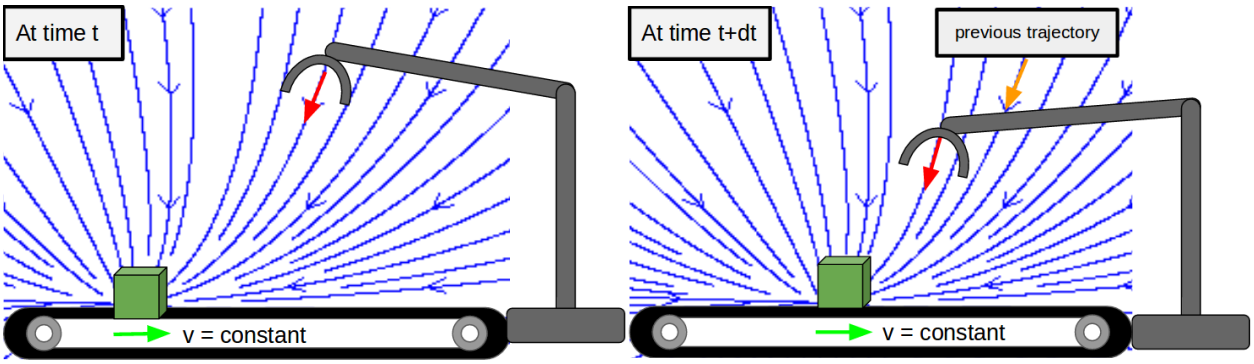


Figure 1.6: Regime with a moving object.

1.2.3 Coupled dynamical system - State of the art

The paper [3], describes how to couple two dynamical systems. The approach uses a robotic arm with a hand that can grasp by closing its fingers. The dynamical systems of the arm and hand are independent. The problem the paper [3] solves is how to make the hand open up in function of the state of the arm. There is a relation Master-Slave between the arm (master) and the hand (slave). Indeed, the objective is to close the hand only when the object is close to the arm.
An algorithm is presented using a DS for the arm and taking the state of the DS of the arm and putting it into a so called coupling function $\Psi(\xi_m)$ with ξ_m the position vector of the master. This coupling function will then infer a value of the target state and give it to the slave. A figure from the paper is shown in 1.7

The coupling presented in our paper is different. In [3], two different DS are synchronized however the shape of the DS of the master or slave is not changed . The movement that the hand will do is only delayed but it is not changed in shape.

In our paper we are trying to modify the DS in norm and in shape in order to accommodate to different speeds. Our coupling is solely based on an external signal v^c of the conveyor belt representing the speed.

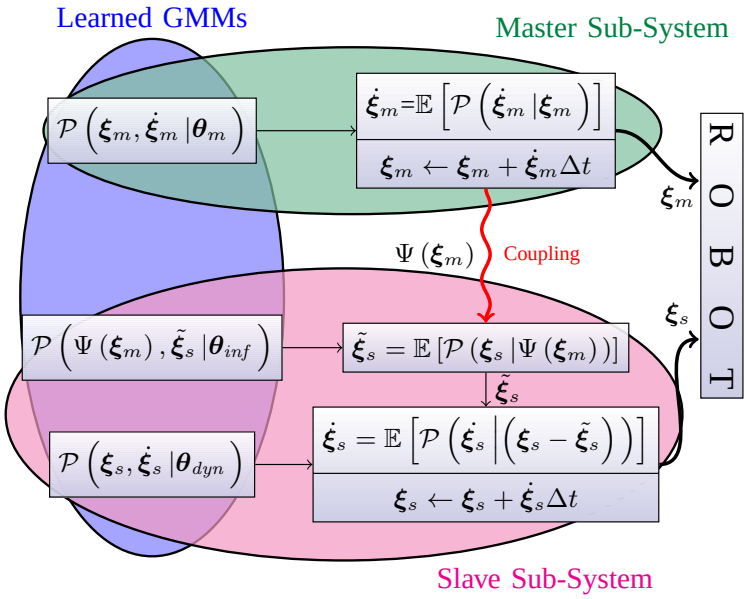


Fig. 2. Task execution using CDS model. Blue region shows the three Gaussian Mixture Models which form the full CDS model. Green region shows the master sub-system where the cartesian position of the robot end-effector evolves in time as a DS and is continuously fed to the robot. Magenta region shows the slave sub-system where the finger joint angles evolve in time as a DS, but also influenced by the state of the master system and fed to the robot. Coupling is ensured by passing selective state information in the form of $\Psi(\boldsymbol{\xi}_m)$ as shown in red.

Figure 1.7: Figure from the paper[3].

Chapter 2

Approach

2.1 Introduction

In this chapter, three different approaches are presented.
In the first part, we try to only change the norm of the dynamical system in order to adapt to the varying speed of the conveyor belt. In the second part, we begin to think of a way to change the shape of the dynamical system. The main idea is to use demonstrations at discrete speeds and do a weighted sum of them to simulate a demonstrations between discrete values of v^c the speed of the conveyor belt. In the third and last part we modify SEDS algorithm in order to include an external parameter v^c that adapts the DS behavior for some speed v^c.

2.2 Scaling method

2.2.1 Motivations

The idea of this solution is to vary the speed along the trajectories. By estimating the position of the arm based on a speed parameter one can tune it in order for the conveyor belt and the arm to be coupled.

2.2.2 Implementation

Application with a static object

In this case the conveyor belt doesn't move. The arm is using the trajectories given by the SEDS to reach the static object on the conveyor belt. This is the first implementation we do to test SEDS algorithm.

The SEDS algorithm gives the relation between the position vector of the robotic arm $\xi = \begin{bmatrix} x^a \\ y^a \end{bmatrix}$ vector and the velocity vector $\dot{\xi} = \begin{bmatrix} \dot{x}^a \\ \dot{y}^a \end{bmatrix}$. $\hat{f} : \mathbb{R}^2 \to \mathbb{R}^2$ is a function with a single equilibrium point at the origin, $\dot{\xi}^* = f(0) = 0$.

$$\dot{\xi} = \hat{f}(\xi) \tag{2.1}$$

Depending on the position of the object on the conveyor belt $[x^c; 0]^T$ one has to shift the function f given by SEDS in order for the arm to target the static object on the conveyor belt. If the

object is within the workspace of the robotic arm the robot is able to perform the given task.

The relation between the current position and the next position according to SEDS is given by the next equation:

$$\begin{bmatrix} x^a_{t+dt} \\ y^a_{t+dt} \end{bmatrix} = \begin{bmatrix} x^a_t \\ y^a_t \end{bmatrix} + \begin{bmatrix} f_x(x^a_t - x_c) \\ f_y(y^a_t - y_c) \end{bmatrix} * dt \tag{2.2}$$

We can however notice that because the object is still not moving the convergence is guaranteed as SEDS is globally stable. An illustration of the situation is shown in Figure 1.5.

Coupling between the robotic arm and the conveyor belt

The DS of the arm is scaled by a factor K_a depending on the speed of the conveyor belt. The norm of the DS is given by $||f(\xi)||$ which can be multiplied by a the parameter K_a depending on the speed of the conveyor belt. The two systems are coupled using the following equations:

$$x^c_{t+dt} = x^c_t + v_c \cdot dt \tag{2.3}$$

$$\begin{bmatrix} x^a_{t+dt} \\ y^a_{t+dt} \end{bmatrix} = \begin{bmatrix} x^a_t \\ y^a_t \end{bmatrix} + K_a \begin{bmatrix} f_x(x^a_t - x_c) \\ f_y(y^a_t - 0) \end{bmatrix} * dt \tag{2.4}$$

We need to define criteria in order to find a meaningful K_a. Indeed, by just taking K_a maximum one can ensure convergence for all velocities as long as there is convergence with K_a and v_c maximum. This is because if the arm is fast enough to reach an object coming with a maximum speed v_c then it is fast enough to reach an object coming slower.

$$\begin{aligned} \overrightarrow{x}^a_{final} &= \overrightarrow{x}^a_{begin} + dt \cdot \sum_{t=0}^{target \cdot reached} K_a \cdot f\left(\begin{bmatrix} x^a_t - x^c_t \\ y \end{bmatrix}\right) \\ x^c_{final} &= x^c_{begin} + dt \cdot \sum_{t=0}^{target \cdot reached} v^c_t \\ \text{target reached when } &\left|\left|\begin{bmatrix} x^a_{final} \\ y^a_{final} \end{bmatrix} - \begin{bmatrix} x^c_{final} \\ 0 \end{bmatrix}\right|\right| < \epsilon \end{aligned} \tag{2.5}$$

The above equation 2.5 gives the final position of each system when the difference in position is inferior to a certain tolerance ϵ. v^c_t is the speed of the conveyer belt at time t.

The arm position in the DS space depends on the position of the conveyor belt at each dt. Those observations show that the only way to solve the system is by brute force meaning computing the position of the conveyor belt at each dt and use it to find the trajectory of the arm. Therefore given a set of planned velocities of the conveyor belt, we can predict the intersection location and time for a certain K_a.

To make our case realistic one has to define more conditions that are present in real life applications in robotics. Every robot has a workspace therefore the point of intersection must be within the workspace of the robotic arm. Furthermore, we assume that the robotic arm can follow any line given by the DS.

Search for the right parameter K_a

We define several different conditions to find the right K_a as follows (with interception within the workspace):

1. Minimize the time to reach the target object. (Time efficiency)
2. Minimize the path taken by the arm to reach the object. (Energy efficiency)
3. Choose a precise location within the workspace as an optimal point of interception. (Precision)

The main ideas of the algorithm are as follows:

- Sweep the parameters K_a from zero to max value
- Run the computation of trajectory to solve the system
- Break the loop if target is not reach after $x^a_{begin} < x^c_{begin}$
- If the target is reached within the workspace and with a better cost function value than previously then save the trajectories or the arm and the conveyor belt.
- Continue until all possible K_a are tested.

The finer the sweep of K_a the more it is likely to find an optimal solution however the computation cost of running the simulation for each K_a increases.

2.2.3 Results

In this section, we run our algorithm for different conditions (time efficiency, precision and energy efficiency). We choose common parameters to compare the chosen K_a. We use the same DS in order to be able to compare our results. If the DS generates a trajectory with $y < 0$ then we set y to zero.
The parameters chosen are defined as follows:

$$\begin{aligned} &K_a \in [0, 5] \text{ tested by steps of } 0.01 \\ &v^c_{slow} = 0.01 \quad \& \quad v^c_{fast} = 0.05 \quad \text{Speed of the conveyer belt} \\ &\text{Workspace: } [x_{min}, x_{max}] = [-1, 1] \\ &dt = 0.1 \end{aligned} \tag{2.6}$$

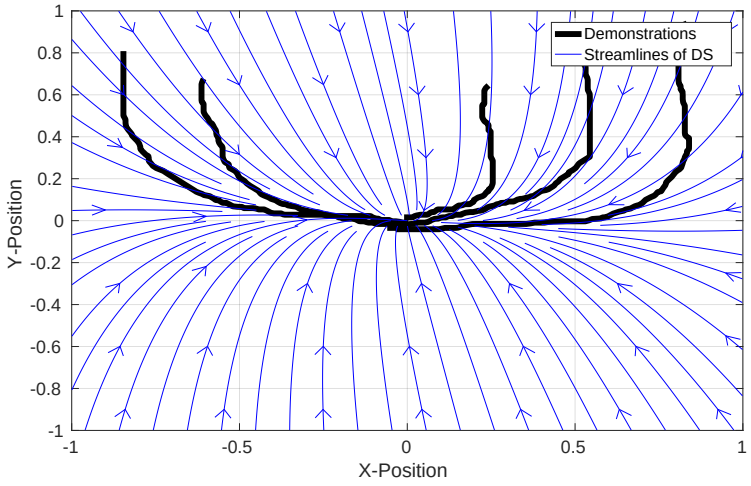
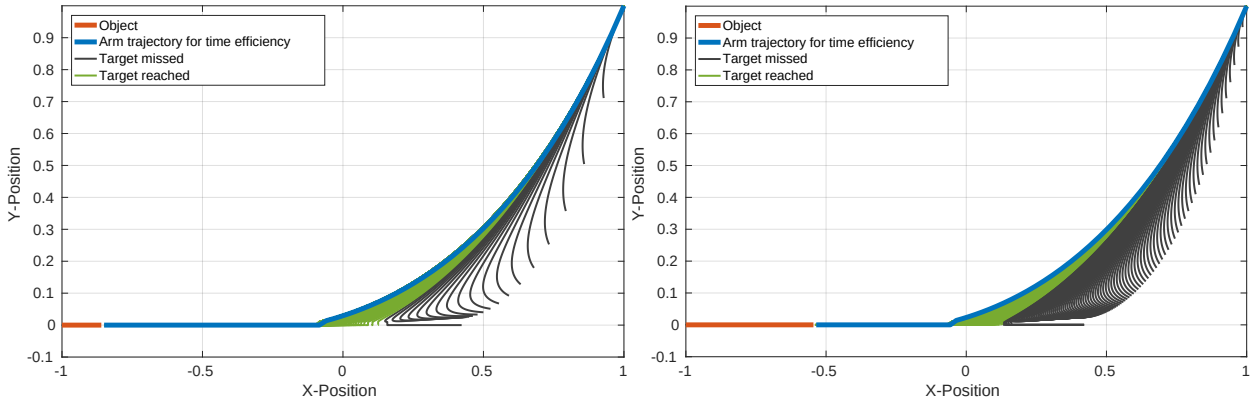


Figure 2.1: Dynamical system used for all results of this part.

Minimize the time to reach the target object.

The results are shown in Figure 2.2. As one would expect the best path is the one with a high K_a. However it is not precisely the maximum which would be 5 but only 4.98 and 4.92. After checking it turns out that above those values it's not worse but it's not better either.
Furthermore there is no upper limit on K_a in order for the object to not converge anymore. Indeed, as K_a increases the velocity of the object becomes negligible to the point where we would have a classical SEDS system with a fixed object.

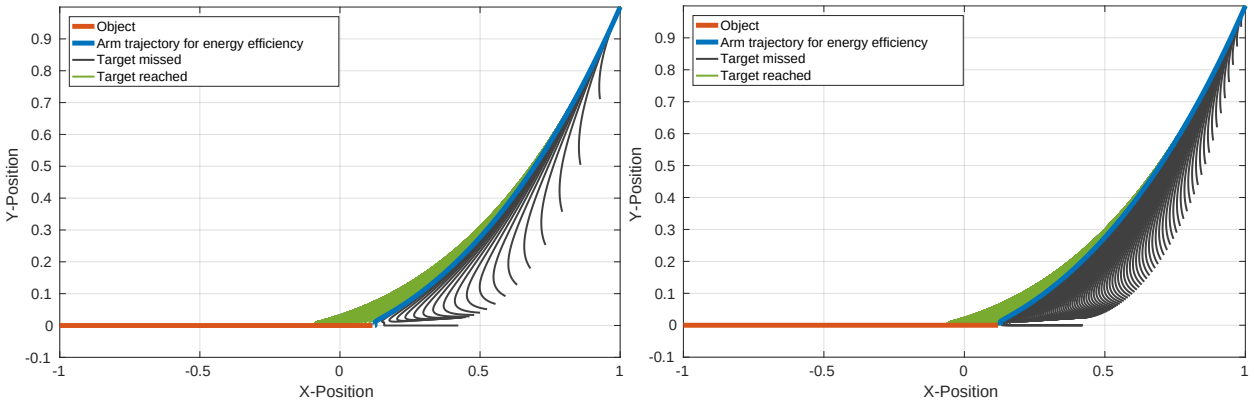


(a) $v^c = 0.01$ and best $K_a = 4.98$ (b) $v^c = 0.05$ and best $K_a = 4.92$

Figure 2.2: The arm and the object trajectory in the world frame. Minimize the time to grab the object.

Minimize the path taken by the arm to reach the object.

The results are shown in Figure 2.3. In order to increase energy efficiency the length of the path taken by the arm to reach the object is computed. The arm tends to wait for the object to come instead of reaching out for it. The values of K_a are as small as possible for a given speed v^c

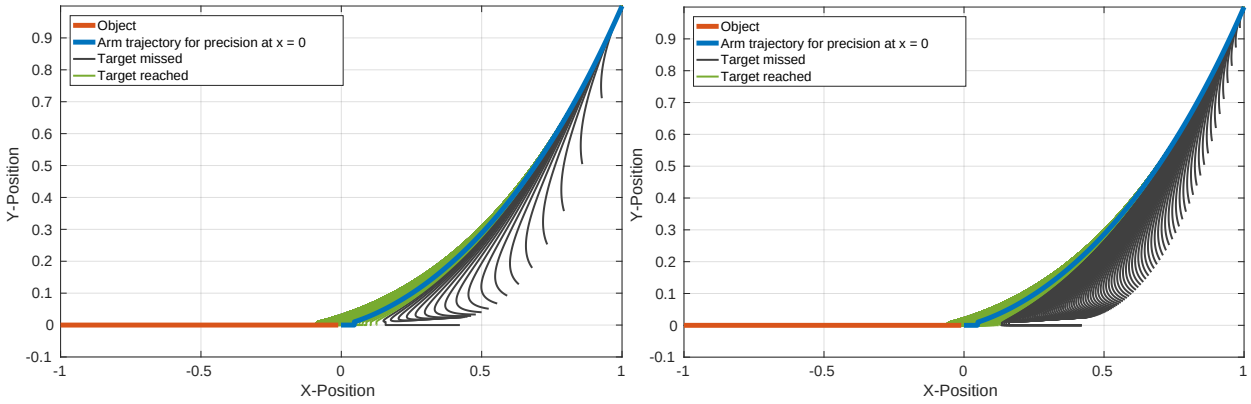


(a) $v^c = 0.01$ and best $K_a = 0.19$ (b) $v^c = 0.05$ and best $K_a = 0.94$

Figure 2.3: The arm and the object trajectory in the world frame. Minimize the path length of the arm trajectory.

Choose a precise location within the workspace as an optimal point of interception.

The results are shown in Figure 2.4. The point of interception between the arm and the object are computed for all K_a. As one can see for the same target $x = 0$ the value of K_a is different for different values of v^c.



(a) $v^c = 0.01$ and best $K_a = 0.25$ (b) $v^c = 0.05$ and best $K_a = 1.24$

Figure 2.4: The arm and the object trajectory in the world frame. Intercept the object close to a chosen location.

Range of interception with defined parameters

In the precedent part a value of $x = 0$ is chosen as a target. In this section an observation is made on the possible target values the arm can have in function of the velocity of the object on the conveyer belt v_c for this specific DS.
In Figure 2.5, one can see represented in red the region where interception of the object is possible given the set of parameters chosen in equation 2.6. Several observation can be noted:

- Red region:
  - The red region gets smaller as the velocity of the object increases. It is therefore more complicated to catch a fast object than a slower one.
  - The red dots are the points of interceptions computed with each K_a. The spacing between them is only dependent on the step chosen for the computation of K_a.
- Blue region:
  - The blue region is the region near the starting position of the object. It gets bigger as the speed of the conveyer belt increases. This is to be expected because as the object goes faster, the object travels more distance until the arm finally reaches it. In order to prevent this, increasing the maximum value of K_a is a solution. Another solution would be to pick up a DS that goes to the origin in a straight trajectory, reducing therefore the traveled distance. However scaling doesn't change the shape of the DS.
- Purple region:
  - The purple region is the region close to the starting position of the robotic arm. It is almost constant in length independently of the speed of the conveyer belt. To reduce this

region one can think of changing the shape of the DS in order to have steeper trajectories in the y direction. Another way is to wait until the object comes close to the arm and start the motion then.

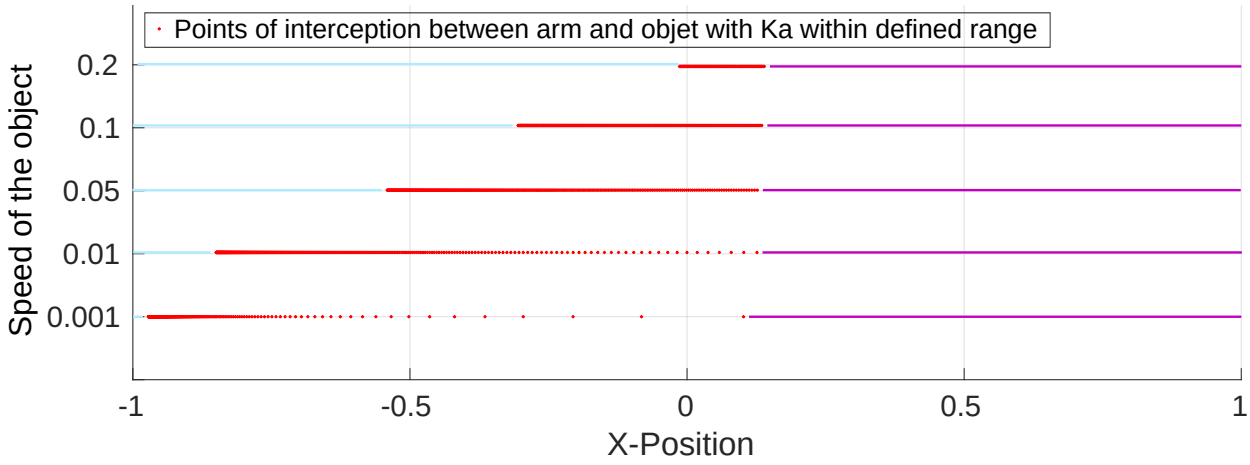


Figure 2.5: Points of interception with all Ka within the defined range in function of the speed of the object on the conveyer belt.

2.2.4 Intermediate conclusion

Our first solution enables us to have some flexibility to grab a moving object on a conveyer belt. As seen in Figure 2.5, the area of exchange depends on the speed of the object however one can always make the system more robust by increasing the K_a maximum.
The necessity of changing the shape of the DS has been shown by the limitations of our first method. Indeed, with steeper trajectories in the y direction, the purple region can be reduced significantly. The blue region can also be reduced by choosing a DS with more direct trajectories to the origin. Changing the shape of the DS not only helps reduce the blue and purple region it also does so without requiring necessarily a faster robotic arm.
In the further sections, the goal is to change the shape and the norm of the DS in function of the speed of the object on the conveyer belt.

2.3 Arrangements on the demonstrations

2.3.1 Motivations

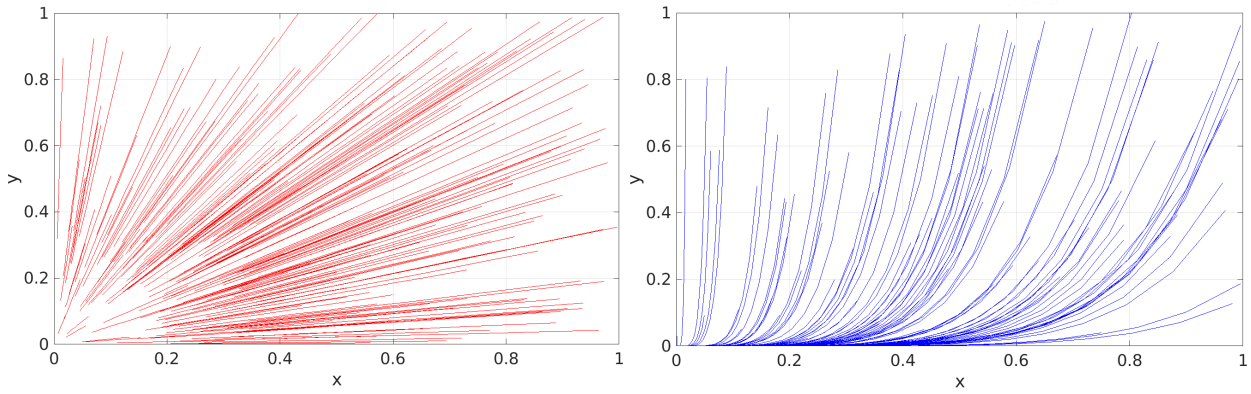
The idea of this solution is to perform several sets of demonstrations with different speeds. By doing a weighted sum of demonstrations at different speeds one can obtain a DS that was not demonstrated. We are now starting to change the shape and the norm of the DS in order to adapt to the speed of the conveyor belt.

2.3.2 Implementation

In order to implement our approach on Matlab, we generate parametrized demonstrations. The approach works for any a set of curves driven by set of parameters $\{\beta_n\}_{n=1}^{n=n_\beta}$, such that $1 \leq n_\beta$ and $n_\beta \in \mathbb{N}$. In this part the set of curves chosen only depends on one parameter $1 \leq \beta_1$ such that:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} x_0 \exp^{-t} \\ y_0 \exp^{-\beta_1 \times t} \end{bmatrix} \tag{2.7}$$

Two sets of demonstrations begins at random points (x_0, y_0) for $\beta_1 = 1$ and $\beta_1 = 6$ are presented in Figure 2.6



(a) $\beta_1 = 1$ (b) $\beta_1 = 6$

Figure 2.6: Demonstrations from different starting positions with $\beta_1 = 1$ and $\beta_1 = 6$

Given the infinite set of curves available one needs to choose the right ones in order to be able to catch an object with a speed v^c. An optimization is implemented and explained later in this paper, see 2.5

To implement the method one needs to collect data of trajectories using different speeds v^c. The idea is to do measurements at discrete speeds $v_1^c, .., v_N^c$. Therefore, given a speed v^c such that $v_p^c < v_c < v_{p+1}^c$ with $1 < p < N$, $p \in \mathbb{N}$, the demonstrations are modified as follows:

$$demos_{v^c} = \frac{v^c - v_p^c}{v_{p+1}^c - v_p^c} * demos_{v_{p+1}^c} + \frac{v^c - v_{p+1}^c}{v_p^c - v_{p+1}^c} * demos_{v_p^c} \tag{2.8}$$

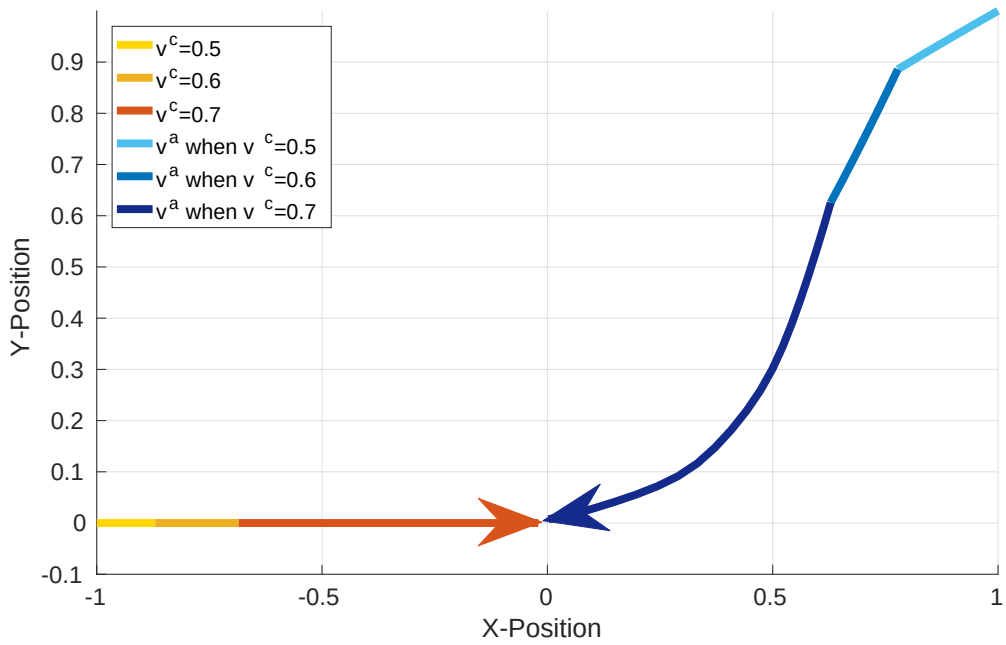


Figure 2.7: Simulation by a change of speed at each change of color. One can notice the change of slope when the speed changes.

2.3.3 Results

In Figure 2.7, the conveyer belt takes successively three different speeds v^c. The robotic arm computes a new DS each time the velocity of the conveyer belt changes. One can notice the sharp changes of trajectory when there is a color change on the blue curve of Figure 2.7. This shows that this approach is capable of changing the shape of the DS.

2.3.4 Drawbacks

We managed to implement an approach that changes the shape of the DS. By using demonstrations on some speeds v_p^c, one can artificially create demonstrations over the continuous parameter v^c. However, averaging the demonstrations requires that one obtains demonstrations at different speeds starting from the same position. Furthermore, the SEDS model must be generated for each speed v^c making it computationally expensive to use.

2.3.5 Intermediate conclusion

In the next part, we try to adapt the SEDS algorithm in order for v^c to become an external parameter of SEDS. The goal is to have gaussians estimating the DS with v^c being a sweep variable that can select the right DS for a certain value of v^c, the speed of the conveyer belt. Furthermore, the optimization function to find the parameters of the set of curves in function of v^c is presented.

2.4 Add a parameter to SEDS

2.4.1 Motivations

The idea of this solution is to add a third dimension to the DS. The three dimensions are now x, y and v^c the speed of the conveyer belt. Given a set of demonstrations with different speeds v^c, one can generate a DS that gives a 2D solution that converges for any speed v^c. The conveyer belt speed v^c is replaced by s parameter which is more general as it represents any external signal.

2.4.2 Implementation

Mathematical approach

The robotic arm is described by a two dimensional vector in $\mathbb{R}^2$. The external parameter $s \in \mathbb{R}$ is a dimension that we add to SEDS. In our case $s = v^c$,

Let A^k be a 2x2 matrix and b^k a 2x1 vector.. According to SEDS presented in section 1.2.1 the function that approximates the DS given a set of demonstrations is given by:

$$\dot{\xi} = \hat{f}(\xi) = \sum_{k=1}^{K} h^k(\xi)(A^k\xi + b^k) \tag{2.9}$$

With our approach the position and velocity vectors are given by:

$$\xi = \begin{pmatrix} x \\ y \\ s \end{pmatrix} \quad \& \quad \dot{\xi} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ 0 \end{pmatrix} \tag{2.10}$$

In order to express the dimension of s as an external parameter the matrices A^k are modified as follows:

$$\begin{pmatrix} \dot{\xi} \\ \dot{s} \end{pmatrix} = \begin{pmatrix} \hat{f}(\xi) \\ \dot{s} \end{pmatrix} = \sum_{k=1}^{K} h^k(\xi; s)\left(\begin{bmatrix} A^k & 0 \\ 0 & -1 \end{bmatrix}\begin{pmatrix} \xi \\ s \end{pmatrix} + \begin{pmatrix} b^k \\ 0 \end{pmatrix}\right)$$

Dummy variable

s influence the activation of k^{th} local dynamics

s does not influences the local dynamics.

Figure 2.8: Modified SEDS

Observations:

- $\dot{s}$ is a dummy variable. The value it takes does not have any significance.
- $h^k(\xi; s)$ influences the activation of the k^{th} local dynamics. In other word this is where the gaussians store the information given by the external input s as well as x and y.
- $\begin{bmatrix} A^k & 0 \\ 0 & -1 \end{bmatrix}$ This matrix has its last line and column set to zero except the last element of the diagonal that is set to -1. The -1 is to keep the matrix stable and invertible. The zeros specially in the column are set in order for the parameter s to not have any influence on the

dimensions of x and y. Indeed A^k is a 2x2 matrix therefore if one develops the first line of the equation one gets :

$$\dot{x} = \sum_{k=1}^{K} h^k(\xi; s) \left(a_1^k \times x + a_2^k \times y + 0 \times s + b_x^k \right)$$

According to the stability theorem seen in (1.10), there are two constraints to satisfy. Let:

$$\bar{b^k} = \begin{pmatrix} b^k \\ 0 \end{pmatrix} \quad \& \quad \bar{A^k} = \begin{bmatrix} A^k & 0 \\ 0 & -1 \end{bmatrix} \tag{2.11}$$

The constraints are as follows:

$$\begin{cases} b^k = \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \forall k = 1..K \\ A^k + (A^k)^T \prec 0 & \text{with } A^k \text{ a 2x2 matrix and } b^k \text{a 2x1 vector.} \end{cases} \tag{2.12}$$

The matrix $\bar{A^k}$ is a 3x3 matrix. In order to ensure stability in the two dimensional plane of $[x, y]^T$ we check only the first two rows and columns of $\bar{A^k}$ for stability, therefore we only check the matrix A^k. By doing so we are sure to have stability in every plane with any defined parameter s.

Optimization of the modified SEDS

In order to find the right parameters a MSE (Mean Square Error) optimization is performed. The parametrization is similar to the original SEDS only the matrices A^k have one dimension lower. the parametrized vector is therefore given by:

$$\Theta = \{ \underbrace{\tilde{\pi}^1 .. \tilde{\pi}^K}_{n \times 1}; \underbrace{\mu_\xi^1 .. \mu_\xi^K}_{n \times 1}; \underbrace{L_\xi^1 .. L_\xi^K}_{n \times \mathrm{n}}; \underbrace{A^1 .. A^K}_{n-1 \times \mathrm{n}-1} \}$$

Figure 2.9: Parametrization vector of the optimization

The parametrization vector shown in Figure 2.9 contains the priors $\tilde{\pi}^k$, the means μ_ξ^k , the cholesky decomposition of the variance matrix L_ξ^k and the component of matrix A^k which only contains the first two dimensions.
MSE method minimizes the error which is the difference between the demonstrations and the model given by SEDS. The cost function is defined as the sum of those errors along a trajectory given by the demonstrations.

$$J = \frac{1}{2 \times \sum_{n=1}^{N} T^n} \sum_{t=0}^{T^n} \left\| \hat{\dot{\xi}}^{t,n} - \xi^{t,n} \right\| \tag{2.13}$$

With $\hat{\dot{\xi}}^{t,n}$ the estimation of SEDS and $\xi^{t,n}$the demonstrations

The constraints of the optimization are the conditions of stability enunciated in equation 2.12.

2.4.3 Results

In this part a real implementation of the modified SEDS is demonstrated. In our case the external signal s corresponds to the velocity of the conveyor belt v^c. In order to find sets of demonstrations that do actually grab the object one has to run an optimization to select the best parametrized curves. Given a set of demonstrations parametrized by the parameter β_n one needs to run an optimization to find the perfect set of parameters β_n for a certain velocity v^c. For further details see the next part 2.5. In this part we set $s = v^c = \beta_1$.

2.4.3.a Set of parametrized curve given as demonstrations

A set of demonstrations has been chosen in order for the trajectory to take a shape that is more appropriate as the speed v^c increases. To have a robust grasping of the object the faster the conveyor the faster the arm should move towards the conveyor belt. The equation of the trajectories depending on β_1 are given by:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} x_0 \exp^{-t} \\ y_0 \exp^{-\beta_1 \times t} \end{bmatrix} \tag{2.14}$$

Number of Gaussians

The SEDS algorithm does an optimization in order to fit the data while ensuring stability. The goal of this part is to provide a method to choose the right number of Gaussians to use in order to get a good fit of the data while have a computational time relatively low.
By taking the final value of the cost function one can get an idea on how close the model fits the data. The algorithm is dependent on random initializations. Therefore we run 50 times SEDS for each number of Gaussians and plot the final value of the optimization function in Figure 2.10. We use 100 demonstrations for each run, the train/test ratio is of 50%. Indeed the cost function is computed on data different than the one used to train SEDS. The precision improves from one to four Gaussians after that the difference is not worth the computational cost. Therefore for our application we will use a regression with $K = 4$ Gaussians.

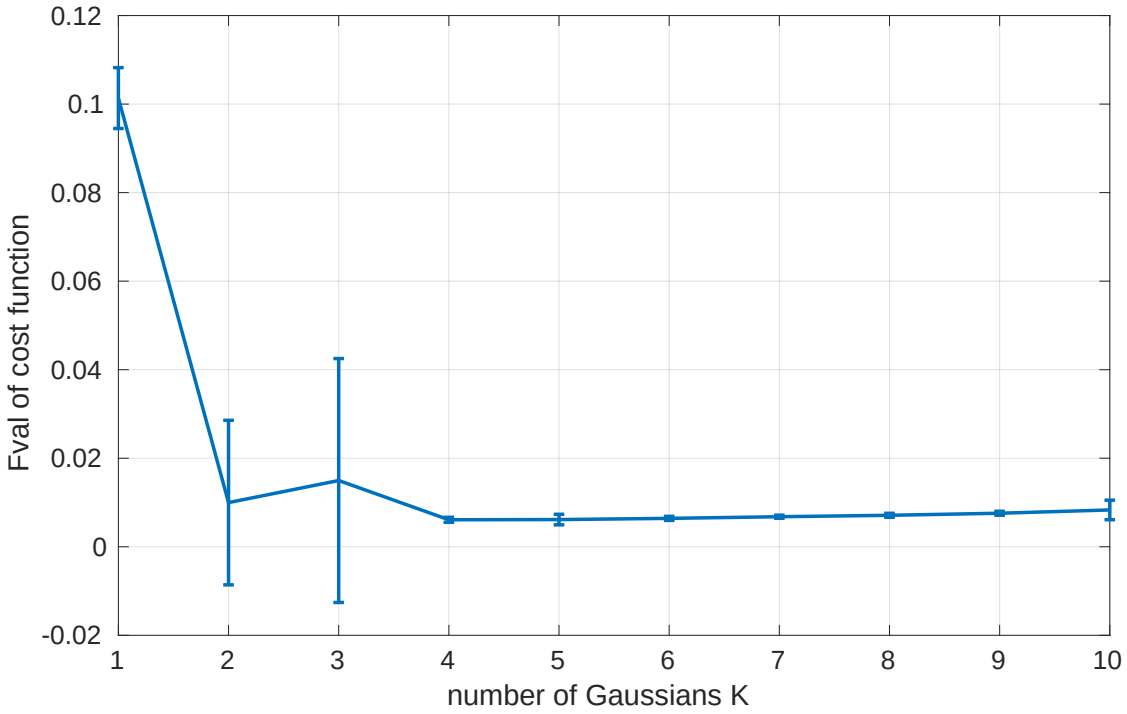
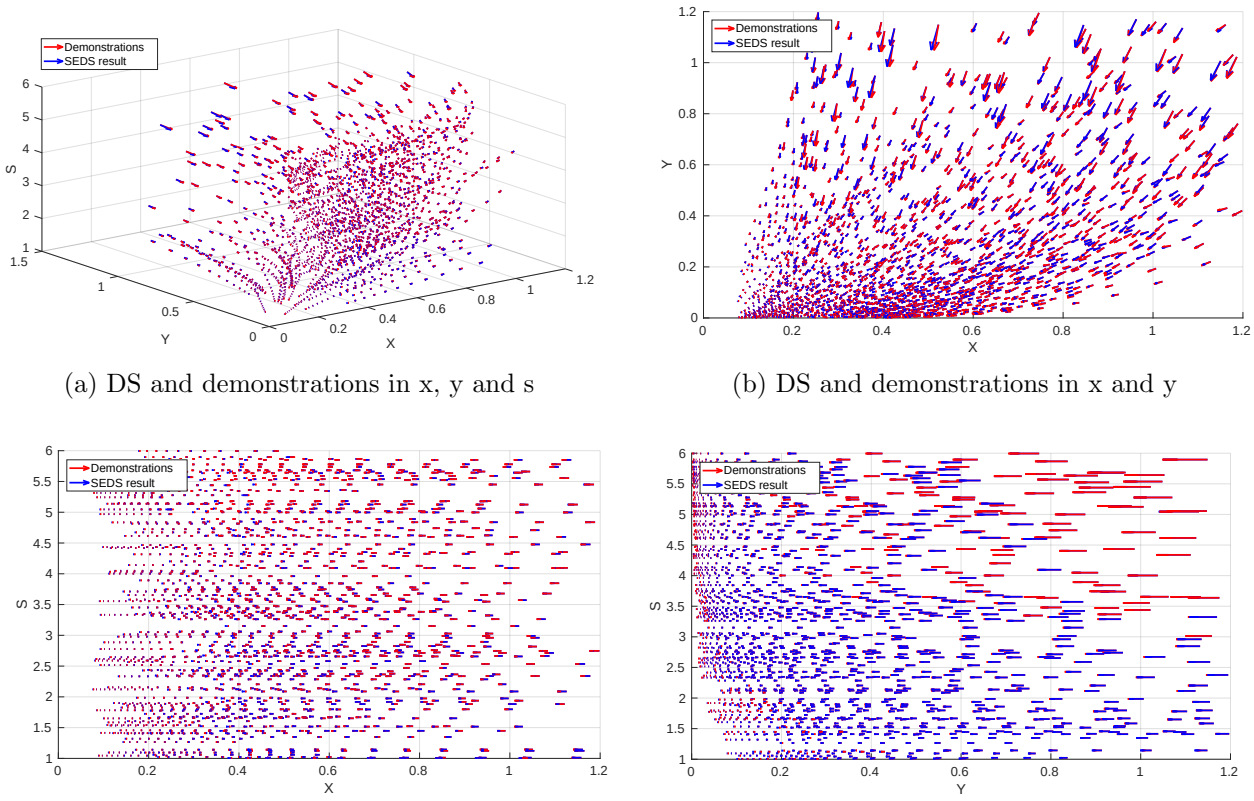


Figure 2.10: Final value of FVAL in function of the number of Gaussians used. The variance of the data is represented by the vertical lines.

DS obtained by running the modified SEDS

The resulting DS is a 3D system with x, y and s as the dimensions. In each plane of parameter s there is only one attractor for x and y. In Figure 2.11a, the DS is represented with the three dimensions, it is stable as the vector field point to the origin for each value of s. In Figure 2.11b, the demonstrations and SEDS approximation are shown. In Figure 2.11c and 2.11d, it is important to notice that all the vector seems flat meaning that if a vector is in a certain plane s it cannot point to another s plan, it can only point in the x-y direction.



(a) DS and demonstrations in x, y and s

(b) DS and demonstrations in x and y

(c) DS and demonstrations in x and s

(d) DS and demonstrations in y and s

Figure 2.11: Vectorial field of demonstrations and modified SEDS

Influence of parameter s

The influence of parameter s is shown in Figure 2.12. One can notice that as the speed $v^c = s$ increases the trajectory becomes more steep in the y direction in order to catch up with a fast object on the conveyor belt.

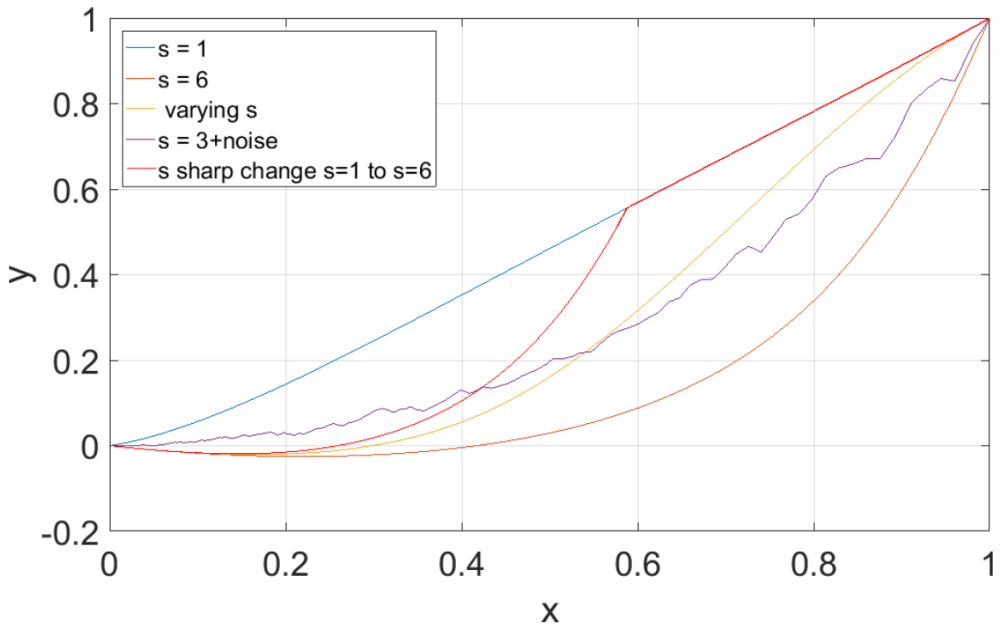
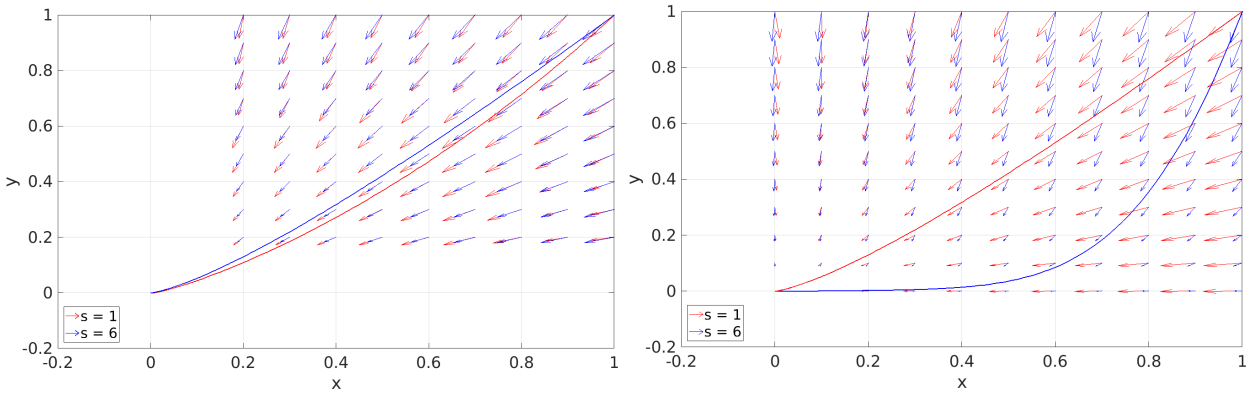


Figure 2.12: DS result with different values of s

Comparison of our new method with the original SEDS

In Figure 2.13, one can see the difference between the original SEDS algorithm and our modified version. The goal is to have a DS that depends on the external signal s. Because of our curve parametrization depending on the single parameter $\beta_1 = s = v^c$, one expects the DS to have a steeper slope when β_1 increases.



(a) Before the adaptation

(b) After the adaptation

Figure 2.13: DS results with s=2 and s=6

2.4.4 Intermediate conclusion

It is important to note here why the original SEDS does not work as well as our modified version. Indeed, the original SEDS can provide the same results as ours however it needs to find a solution where in the activation matrices A^k the zeros as placed exactly as what we did. This means that the original SEDS is not constrained enough for the optimization we are trying to accomplish.

Second important note is that in the previous result section we set $s = v^c = \beta_1$. However, one needs to distinguish carefully between the three variables. The variables s is a general external signal independent on the other dimension of SEDS. The variable v^c is an example of what could be the variable s in a real application, here v^c is the speed of the conveyor belt. Last, the variable β_1 is a parameter of the demonstrations curves. It can possibly be part of a bigger set of $\{\beta_n\}_{n=1}^{n=n_\beta}$, such that $1 \leq n_\beta$ and $n_\beta \in \mathbb{N}$. In order to provide quality demonstrations one has to run an optimization with the parameters of the set $\{\beta_n\}_{n=1}^{n=n_\beta}$, this will generate trajectories that will be selected or not using a defined cost function. The trajectories that can pick up an object at a certain speed v^c are saved as $[x; y; v^c]^T$ and $[\dot{x}; \dot{y}; 0]^T$. This is further discussed in 2.5.

2.5 Find suitable demonstrations for moving objects

2.5.1 Motivations

In order to generate demonstrations suitable for a certain speed v^c of the conveyor belt, one has to perform a search. Indeed, there are an infinite number of trajectories that can successfully intercept an object on a moving conveyor belt. In this part an approach is presented to choose the right trajectories for a certain speed v^c. Once one has trajectories in function of different speeds v^c, one can use them to learn the DS using the approaches in 2.3 and 2.4.

2.5.2 Implementation

Given a a set of curves driven by a set of parameters $\{\beta_n\}_{n=1}^{n=n_\beta}$, such that $1 \leq n_\beta$, $n_\beta \in \mathbb{N}$ and $\beta_n \in \mathbb{R}$. This set of parameters generates different curves with the function $g(\beta_1, \beta_2, ..., \beta_{n_\beta}, t) \in \mathbb{R}^d$ with $d \in \mathbb{N}$ the dimension of the robotic arm (in our case $d = 2$). In order to select the right combination of $\{\beta_n\}_{n=1}^{n=n_\beta}$ for a certain speed v_c. One can run the simulation with a given set of $\{\beta_n\}_{n=1}^{n=n_\beta}$ by following the given equation:

$$\begin{aligned} &\overrightarrow{x}^a(t) = g(\beta_1, \beta_2, ..., \beta_{n_\beta}, t) \\ &x^c(t) = x_0^c + v^c * dt \\ &\text{target reached when } \left|\left| \begin{bmatrix} x^a_{final} \\ y^a_{final} \end{bmatrix} - \begin{bmatrix} x^c_{final} \\ 0 \end{bmatrix} \right|\right| < \epsilon \end{aligned} \tag{2.15}$$

One has to define a cost function that will restrict the infinite possibilities. For example, a cost function that minimizes the time to pick up the object will search for parameters $\{\beta_n\}_{n=1}^{n=n_\beta}$ such that the time at which the target is reached is minimized.
The constraints required are to bound the parameters $\{\beta_n\}_{n=1}^{n=n_\beta}$. Simulations that do not converge must be interrupted. We defined that if the robot x-position gets bigger than the arm starting x-position then the object went through the arm and the simulation is interrupted. A fixed value is then given to the cost function. The cost function has flat regions due to those failed simulations. In order to perform an effective search fmincon[8] was not used but instead patternsearch[9] which is a global optimization function.
Note that the demonstrations selected are given to the modified SEDS in respect to the object position.

2.5.3 Results

Figure 2.14, is a diagram that presents the global view of the implementation. Given a set of parametrized curves one can run simulations to find the most fitting curve given a certain velocity of the conveyor belt. Those curves are then used to train effectively the modified SEDS.

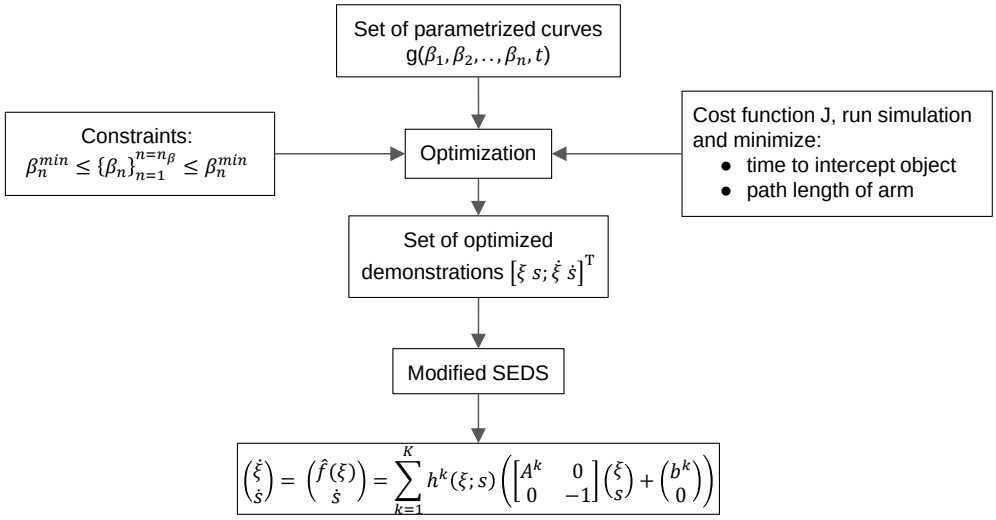
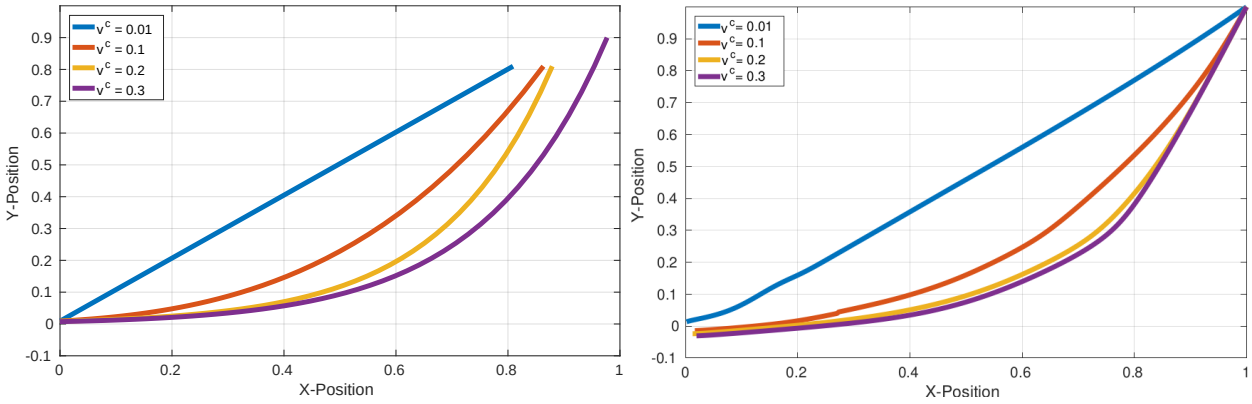


Figure 2.14: Diagram of the whole system.



(a) Example of trajectories selected by curve optimization.

(b) Trajectories generated by SEDS after learning from optimized demonstrations.

Figure 2.15: Optimizing the time to intercept the object. Object starts at x = -1; Robotic arm starts at $[1; 1]^T$

An optimization minimizing the time to intercept the object by the arm is implemented. The set of curves used depends on two parameters $\{\beta_1, \beta_2\}$. The curves are represented as a DS as it is presented in 1.1.

$$\dot{x} = \begin{bmatrix} -\beta_1 & 0 \\ 0 & -\beta_2 \end{bmatrix} x \quad \text{with } 0 \leq \beta_1 \leq 1 \text{ and } 0 \leq \beta_2 \leq 1 \tag{2.16}$$

As seen in 1.1, when the velocity of the object is low the trajectory provided is direct therefore $\beta_1 = \beta_2 = 1$. When the velocity increases, β_1 decreases in order to not go towards the object as it is already coming towards the robotic arm.

2.5.4 Limitations of the method

By doing an optimization on parametrized curves one chooses a random starting position, a random speed v^c and try to find a trajectory that intercepts the object. However starting position conflict with each other. Indeed, if one starts from $[1; 1]^T$ and from $[0.1; 0.1]^T$ the trajectory from $[1; 1]^T$ will go through the region of $[0.1; 0.1]^T$ in order to reach the origin. SEDS tries to find a compromise between the two trajectories present in the region close to the origin for a certain speed v^c. This often result in a solution that neither works for both of the starting positions. One can conclude that it is only possible to implement a certain region of the space.
Furthermore, SEDS tries to estimate the data as best as it can while ensuring stability of the DS. Therefore the resulting DS is not overfitting the data. While making an optimization on the curves to find suitable demonstrations one has to find demonstrations that are suitable even if they are modified by SEDS algorithm.

Chapter 3

Conclusion

In conclusion, by scaling the DS as we did in 2.2 we observed a possible way of coupling the conveyor belt with the robotic arm. While this method provides a range of solutions it is constrained by the fact that the shape of the DS cannot be changed. This results in an array of solutions that is more limited.
In 2.3, we manipulate demonstrations at discrete velocities v^c in order to find suitable demonstrations for a given continuous value of the speed of the conveyor belt. Running SEDS on those manipulated demonstrations provides DS with different shapes as the demonstrations used are different for each speed v^c of the conveyor belt. The main problem is that the method is computationally expensive as one has to find a new DS for each new speed v^c.
Last in 2.4, we add an external signal s to the modified SEDS algorithm. The modified SEDS treats this signal (in our case this signal is v^c) as an independent variable. The DS is still stable in the dimensions of x and y but the shape and norm of it changes in function of s. Finally to train effectively the DS, we present an approach using parametrized curves that are selected using an optimization.
Our hypothesis is validated. Indeed, we manage to improve the probability of intercepting the object by coupling the conveyor belt and the robotic arm. However, the reliability of the method is not perfect. Conflict between demonstrations starting at different positions and SEDS stability conditions make the curve generated by our final DS not exactly the same as the demonstrations. The method is effective when a certain region of the space of the DS is trained.

Chapter 4

Bibliography

[1] https://ifr.org/img/uploads/Executive_Summary_WR_Industrial_Robots_20161.pdf *Executive Summary World Robotics 2016 Industrial Robots*

[2] S. M. Khansari-Zadeh and A. Billard, *Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models*, IEEE Transactions on Robotics, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[3] Shukla Ashwini and Aude Billard. *Coupled dynamical system based arm–hand grasping model for learning fast adaptation strategies*, Robotics and Autonomous Systems 60.3 (2012): 424-440.

[4] H. F. Durrant-Whyte, N. Roy, and P. Abbeel, *A framework for push-grasping in Clutter*, Robotics: Science and Systems VII. MIT Press, 2012.

[5] http://lasa.epfl.ch/sourcecode/

[6] http://www.alecjacobson.com/weblog/?p=2473

[7] S. M. Khansari-Zadeh and A. Billard, *The derivatives of the SEDS optimization cost function and constraints with respect to the learning parameters*, Technical report. EPFL-REPORT-165414

[8] https://ch.mathworks.com/help/optim/ug/fmincon.html?requestedDomain=www.mathworks.com

[9] https://ch.mathworks.com/help/gads/patternsearch.html

[10] Li, Zexiang, Ping Hsu, and Shankar Sastry. *Grasping and coordinated manipulation by a multifingered robot hand*, The International Journal of Robotics Research 8.4 (1989): 33-50.

[11] Yamamoto Yoshio and Xiaoping Yun. *Coordinating locomotion and manipulation of a mobile manipulator* Decision and Control, 1992., Proceedings of the 31st IEEE Conference on. IEEE, 1992.