ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

STI - Microtechnique - Master II

Semester Project

Human-intention recognition in ambiguous hand-overs

Author :
Camilla CARTA

Supervisors :
Prof. Aude BILLARD
Mahdi KHORAMSHAHI

December 7, 2020

Contents

1 Introduction

In the last decades the advancements in robotics revolutionised the world as we knew it. Breakthroughs were made in many fields: robots are now being adopted in industrial production lines, in dangerous situations for humans, for example military operations and hazardous environments, as well as in search and rescue. Still robots are not very present in everyday life. This can be explained by the number of challenges that arises in interaction with humans: multitasking, safety for the user and intuitive interaction are just a few examples.

In most of these applications, the robots are mainly thought to complete a series of specific tasks by switching between pre-programmed states. However, one of the challenges that is faced is the ability to handle unforeseen scenarios where the choice of the current task to complete may be ambiguous. Indeed, as humans users can be unpredictable, it is important to have a system sufficiently flexible to allow a safe interaction with human beings. Another important point is that robots intended for cooperative tasks with humans should allow for intuitive interactions, as to facilitate collaboration between them.

In this setting, this project follows the current state of the art and the current work done in the lab (presented in Sect.1.1) and tests a new approach to the problem of human-robot cooperation and interaction by using visually tracked manipulations. This approach is explained in details in Sect.1.2.

1.1 Previous work

Previous semester projects implemented a motion adaptation using parametric DS based approach first as theoretical simulation [1], then on the real robot [2]. The goal of these projects was for a robot to understand the human intention by allowing the robot to switch behaviour, that is, to have a motion generator based on DS which would have extra degrees of freedom. These degrees of freedom, or parameters, would be identified by the robot to best adapt to human intention.

Furthermore, the project was based on the work done in [3], where the focus was to address the compliance and the adaptation mechanism of robots at a task-level, by understanding the intention of the user. In this approach, a dynamical system based framework functions as motion-generator and as predictive model for the human intention. Moreover, the human intention manifests as exertion of forces on the robot and deviating from the nominal trajectories. The details of this approach are presented in Sect.1.3 and 1.4.

1.2 Proposition

Based on the previous works presented above, this project aimed at taking a novel approach to the problem of human-robot interaction and robot understanding of human intentions in ambiguous scenarios. More precisely, a new approach would be to communicate with the robot directly through visual interaction instead of the usual haptic interaction.

In order to do this, it was proposed to introduce an RGB camera that could visualize a predefined workspace, defined such that a robot's end-effector could arrive to any point of the workspace. In this workspace, a number of tasks would be presented to the robot.

In the scope of this project, the number of tasks was limited to two objects. The tasks in this project would be presented to the robot under the form of red cubes (an example of which can be seen in Fig.1), held by a human subject that would handle the tasks in such a way that the robot could track them using the RGB camera. The human subject would then propose one of the tasks to the robot by moving it closer to it. The idea is that the robot should converge to the proposed task, and it could then operate on it. The steps for this case scenario can be seen in Fig.2a-2c for a right-sided hand-over and in Fig.3a-3c for a left-sided hand-over.

Additionally, a second scenario would be to take away the previously proposed task before the robot could converge on it, while approaching the second one to the robot. An algorithm should be designed such that this correction should cause the robot to converge to the newly proposed task. An example of this scenario can be seen in Fig.2d-2f for a right-sided hand-over and in Fig.3d-3f for a left-sided hand-over.

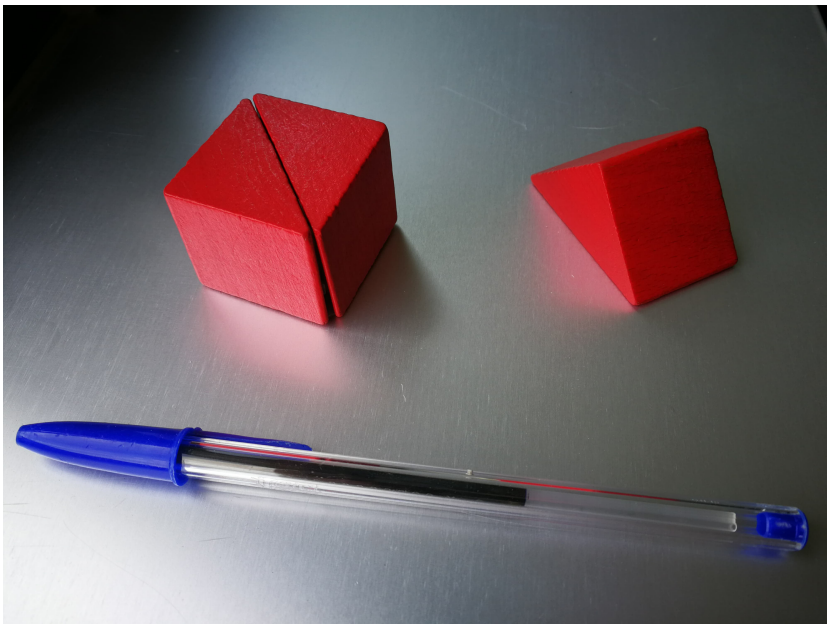
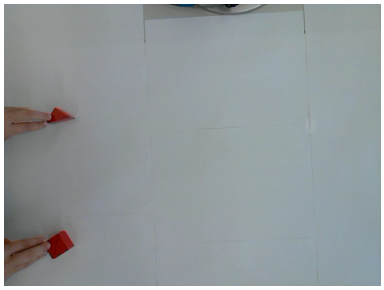
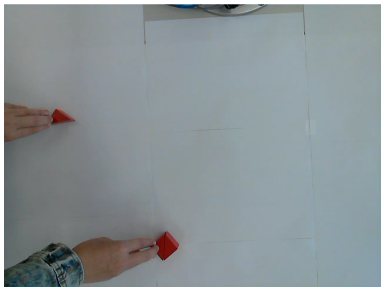


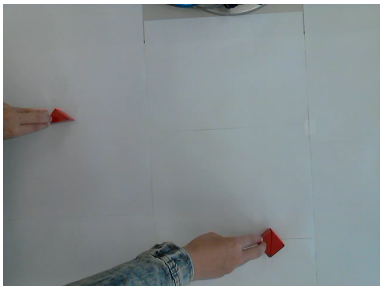
Figure 1: Example of the red objects used as tasks in the videos, with a pen as size reference



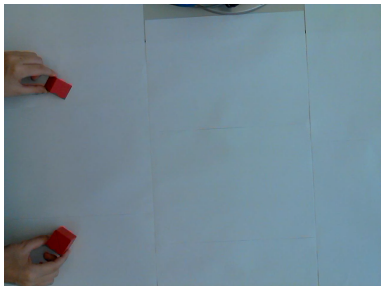
(a) Simple - step 1



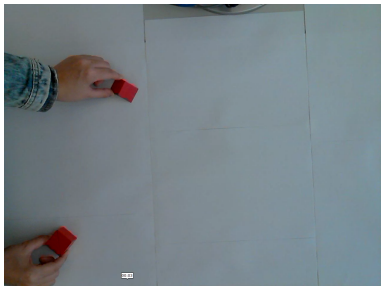
(b) Simple - step 2



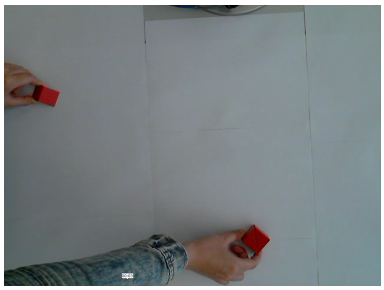
(c) Simple - step 3



(d) Correction - step 1

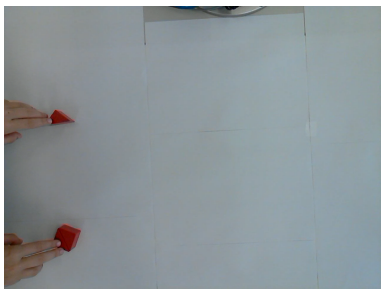


(e) Correction - step 2

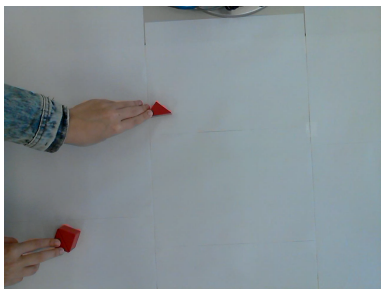


(f) Correction - step 3

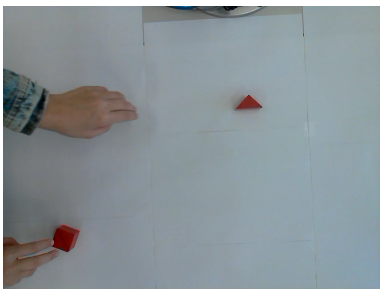
Figure 2: Examples of right-sided hand-overs for the simple hand-over scenario and a correction scenario



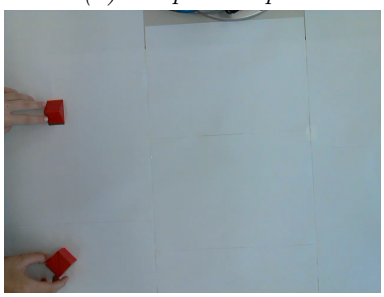
(a) Simple - step 1



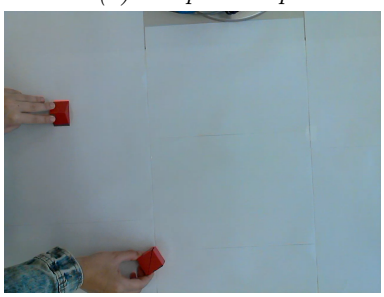
(b) Simple - step 2



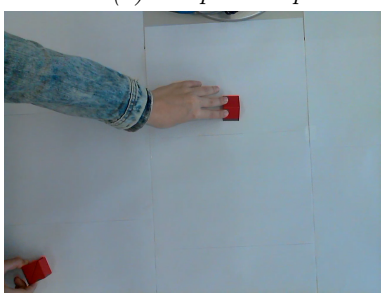
(c) Simple - step 3



(d) Correction - step 1



(e) Correction - step 2



(f) Correction - step 3

Figure 3: Examples of left-sided hand-overs for the simple hand-over scenario and a correction scenario

Both scenarios are particularly challenging as there is no exact solution. Indeed, this kind of setup can lead to ambiguous situations, where there is no obvious task which the robot should

operate on. Thus it is necessary to find a decision-making method allowing the robot to coherently decide which task to converge to. In Fig. 4 an example of an ambiguous situation is presented.

In order to generate the motion of the robot and to fulfil the tasks, we employ Dynamical Systems (DS) which is explained in detail in Sect. 1.3 and 1.4. Different adaptation rules were discussed and tested on a series of benchmarking videos that reproduced the two scenarios explained above. The series was constituted of 40 videos in total, equally split between the two cases. In the 20 videos for the simple case, 10 were of a right-sided hand-over, each with a different type of movement (examples of which are shown in Fig. 5). These movements were repeated symmetrically for the left-sided hand-over. The same was then done for the hand-over case with correction.

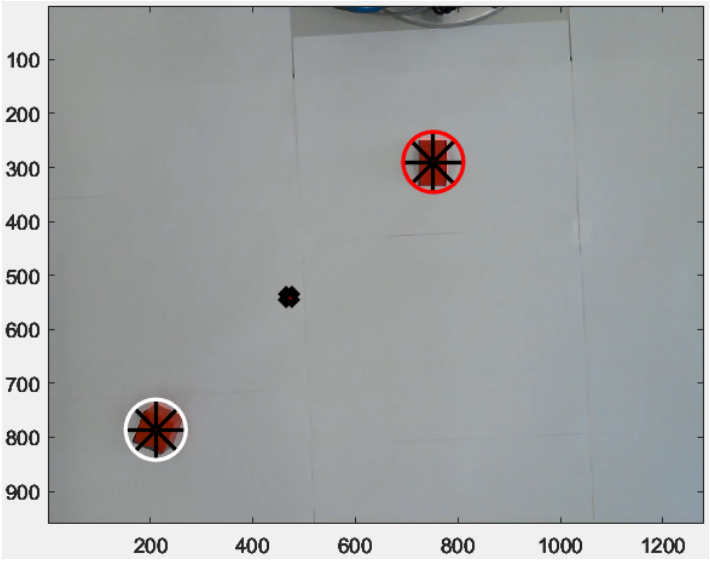
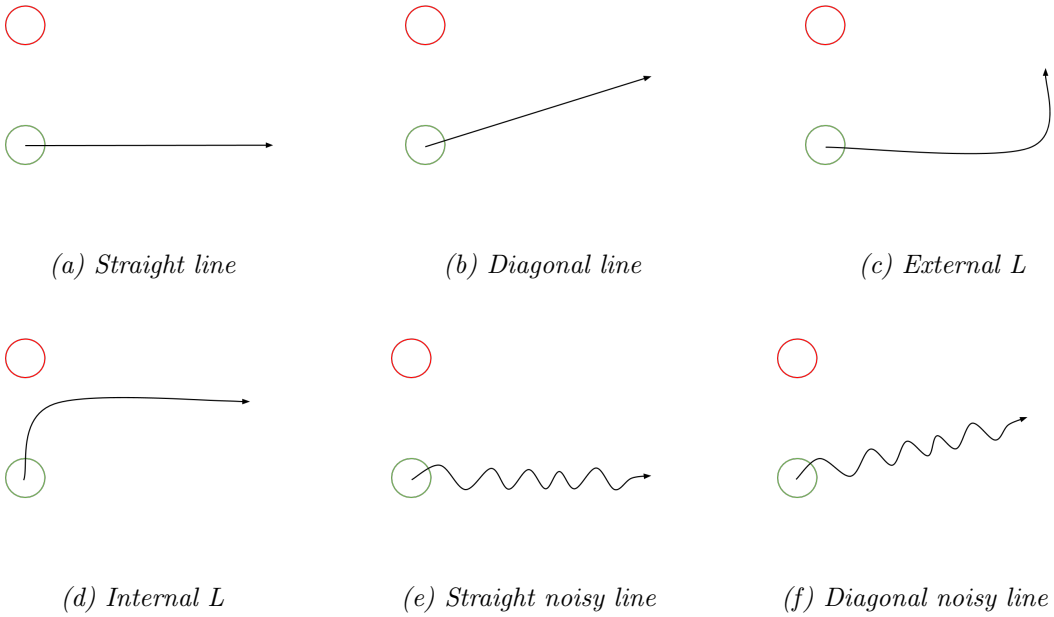


Figure 4: In this figure, the visualization of the Matlab implementation can be seen. The task positions are circled (task 1 in red and task 2 in white) while the robot end-effector position is represented by the black dot. In this particular case, the robot is stuck in between two static tasks, which are at the same distance from it. In this case, it is difficult to establish which would be the "right" task to move towards



(a) Straight line *(b) Diagonal line* *(c) External L*

(d) Internal L *(e) Straight noisy line* *(f) Diagonal noisy line*

Figure 5: Examples of trajectories used as benchmarking videos

1.3 Motion planning using dynamical systems

As a reminder, this project was done in continuity with two previous semester projects ([1] and [2]) and with the work done on task-adaptation ([3]). For the approach used in these papers, DS are used for motion generation and for task identification. Fig. 6 shows a simplified version of the mechanism of motion planning used in the previous works, which lays the basis for this project. It can be seen that a DS based module is in charge of computing a desired velocity $\dot{x}_d$. The robot is then controlled in velocity by the resultant velocity of the module. The robot's velocity $\dot{x}_r$ then follows the direction of the DS. Indeed, this mechanism can be used to direct the robot towards a specific attractor. In Fig. 7, an example of such motion generation can be seen. A DS, represented by the gray streamlines, results in the final trajectory in black. When an external disturbance is applied, in this case the interaction of a human subject, the robot deviates from the nominal trajectory and resumes following the DS after the perturbations.

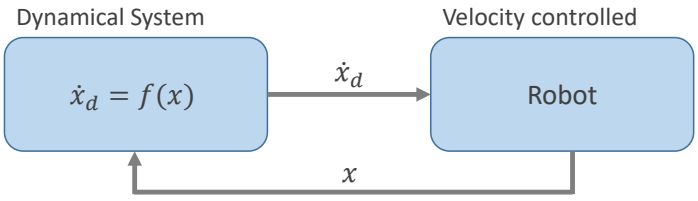


Figure 6: Architecture of the motion planning

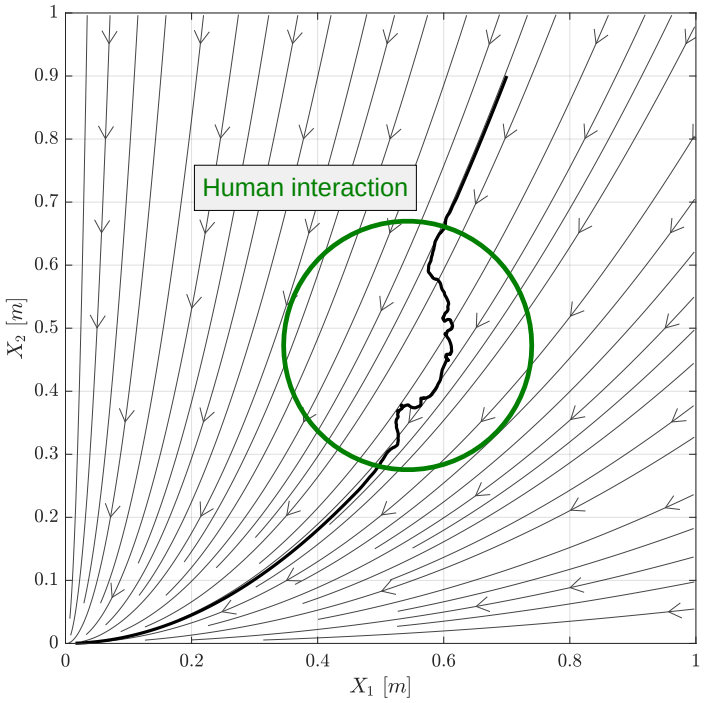


Figure 7: DS based motion generation behaviour when subjected to external perturbations

1.4 Task adaptation using Dynamical systems

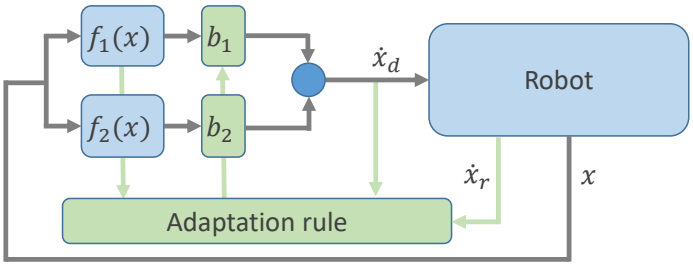


Figure 8: Architecture of the task adaptation mechanism for a scenario with static tasks where f_1 and f_2 are the DS associated with the task 1 and 2 respectively, b_1 and b_2 are the beliefs associated with task 1 and 2 respectively, $\dot{x}_d$ is the desired velocity which commands the robot, and $\dot{x}_r$ is the actual velocity of the robot, x is the robot position.

Fig. 8 presents the structure of the task adaptation approach used in [3] for two static tasks. It can be seen that the motion generator corresponds to the following linear combination:

$$\dot{x}_d = \sum_{i=1}^{N} b_i f_i \tag{1}$$

The tasks are identified through values assigned to each, which are the *beliefs* (b_i). These

values must respect the following conditions:

$$\sum_{i=1}^{N} b_i = 1 \quad \text{and} \quad 0 \leq b_i \leq 1 \tag{2}$$

The beliefs are updated at each iteration by an adaptation rule, which is based on the DS results, the robot's desired velocity and the robot current velocity. In particular the adaptation rule in [3] uses the following adaptation mechanism to calculate the vector of belief-update:

$$\dot{\hat{b}}_i = -\epsilon(|\dot{x}_r - f_i|^2 + 2\sum_{j \neq i} b_j f_j^T f_i) \tag{3}$$

where the parameter ϵ is the adaptation rate.

In a second step, the beliefs are recalculated using the Winner-Takes-All (WTA) process, which was taken from [3] and is reported in Alg. 1.

Algorithm 1 Winner-Takes-All

Input:A vector of beliefs $B = [b_1, ..., b_N]$

Input:A vector of belief-updates $\dot{\hat{B}} = [\dot{\hat{b}}_1, \ldots, \dot{\hat{b}}_N]$

$w \longleftarrow \arg\max_i \dot{\hat{b}}_i$
if $b_w = 1$ **then**
  $\dot{b}_i \longleftarrow 0$ for $\forall i$
  return $\dot{B}$
end if
$\nu \longleftarrow \arg\max_i \dot{\hat{b}}_i \quad \forall i \neq w$
$z \longleftarrow (\dot{\hat{b}}_w + \dot{\hat{b}}_\nu)/2$
$\dot{b}_i \longleftarrow \dot{\hat{b}}_i - z \; \forall i$
$S = 0$
for i **do**
  if $b_i = 0$ or $\dot{b}_i > 0$ **then**
    $S \longleftarrow S + \dot{b}_i$
  end if
end for

As effect of the WTA process, the two highest belief-updates $\dot{\hat{b}}_w$ and $\dot{\hat{b}}_\nu$ are taken and their average subtracted. As shown in Fig. 9, this allows to calculate the beliefs derivatives, which in return indicate how the beliefs b_1 and b_2 are changing. As such, only one task (i.e., the winner) has a positive update. This allows the belief of the most similar task to increase and the other beliefs to diminish.

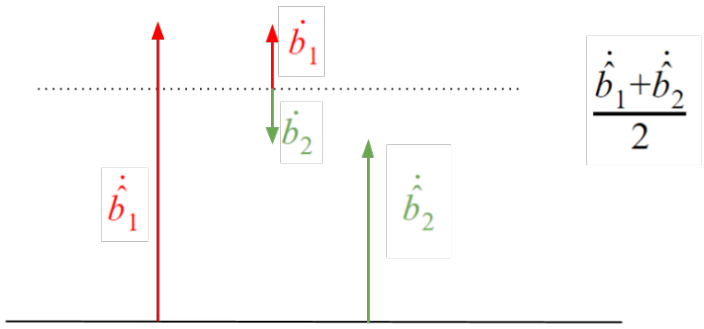


Figure 9: The belief update mechanism

Fig.10 shows the application of the adaptation rule detailed above. In this case, for two fixed tasks (represented by the DS f_1 and f_2) a same robot trajectory is represented in red for the referential of the task 1 and in green for the referential of the task 2. This case represents a robot approaching two fixed tasks under human supervision. While at the beginning the trajectory of the robot is straight towards the tasks and shows no decisive preference, a human intervention steers the robot towards task 2, which is finally the task the robot converges to. It can be observed that the robot trajectory coincides better with the second DS than with the first one, as such the robot converges towards the second task.

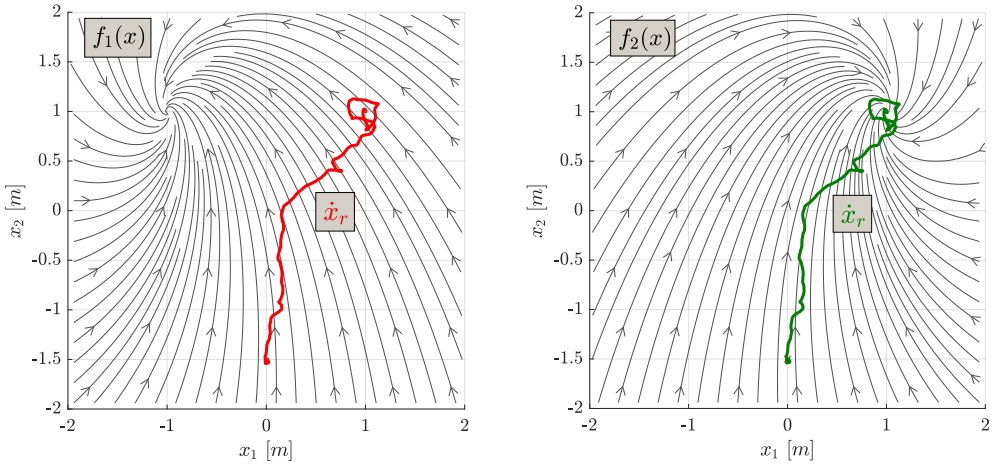


Figure 10: Example of the adaptation rule application for fixed tasks and with human haptic intervention. Human-generated motion is compared against two dynamical system where the second one has higher level of similarity.

2 Methodology

In Sect. 1.4, the implementation used in [3] was presented. In this section, a modified implementation to adapt to the new approach is presented. Indeed, the previous approach only took into account static tasks, more precisely tasks with a defined spatial position. For this project, the tasks are moving objects, as such their dynamic positions and velocities were to be considered into the motion planning for the robot, as well as in the adaptation mechanism.

Moreover, different adaptation rules were studied for this project and are detailed below.

2.1 Motion generation

In Fig. 11 the architecture of the algorithm implemented for this project is presented. By comparing this architecture with the one presented in Fig. 8, it can be seen that the tasks positions $\bar{x}_i$ and velocities $\dot{\bar{x}}_i$ are dynamical variables that have to be integrated in the motion generation and in the task adaptation.

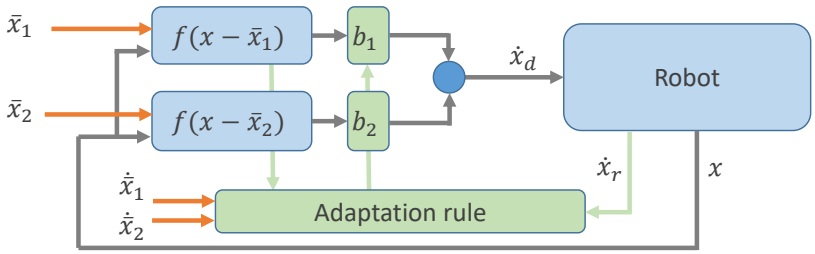


Figure 11: Architecture of the new task adaptation mechanism where f is the DS in charge of motion generation, b_1 and b_2 are the beliefs associated with task 1 and 2 respectively, $\dot{x}_d$ is the desired velocity which commands the robot, $\dot{x}_r$ is the actual velocity of the robot, x is the robot position, $\bar{x}_1$ and $\bar{x}_2$ are the task-target positions, $\dot{\bar{x}}_1$ and $\dot{\bar{x}}_2$ are the task-target velocities.

It is assumed that the position of each object is available as $\bar{x}_i$. Moreover, a dynamical system f generates reaching motions for a given target as follows (e.g., the first object).

$$\dot{x}_d = f(x - \bar{x}_1) \tag{4}$$

However, in the adaptive case, the two reaching motions are combined based on the beliefs b_i as follows.

$$\dot{x}_d = \sum_{i=1}^{N} b_i f(x - \bar{x}_i) \tag{5}$$

In the following, different adaptation rules to update such beliefs according to the motion of the tasks are described.

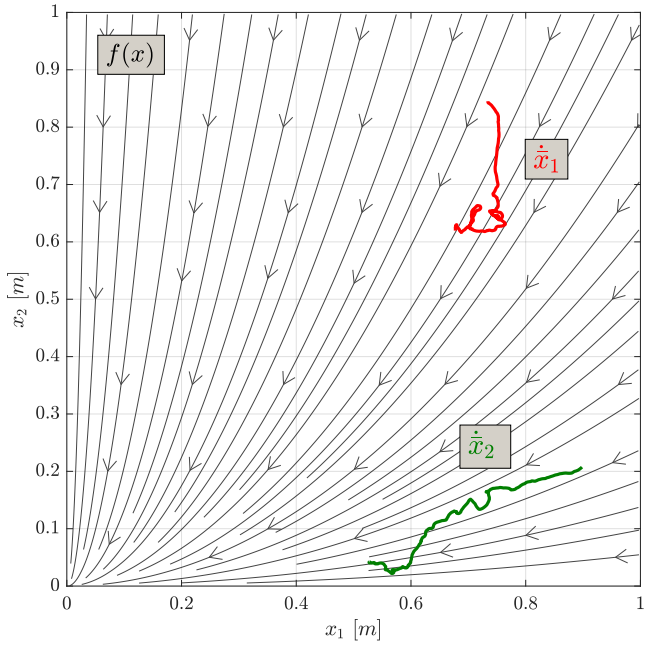


Figure 12: Example of the task adaptation for the new implementation in a scenario with two tasks (task 1 in red and task 2 in green, while the gray arrowed lines indicates the form of the DS f). The robot would then be the attraction point of the DS f in $[0, 0]$. It is important to notice that here the tasks are moving independently from the robot. In this case, the robot is trying to identify which of the tasks should be the target. As such the tasks velocities are compared to the DS f. Task 1 is further from the robot and its trajectory is not coinciding with the DS as much as the trajectory of the 2^{nd} task, which is also the closest one to the robot. As such, 2^{nd} task should be identified as the target.

2.2 Adaptation

Adaptation rule 1 Initially, the following reasoning was developed: as the objective was to have the robot to move towards the task going towards it, a cost function that minimizes the sum of the inner products between the velocities of the robot and of the tasks should be developed. Consider the following cost function:

$$J_1 = \sum_{i=1}^{N} b_i f(x - \bar{x}_i)^T \dot{\bar{x}}_i \tag{6}$$

This cost function will be minimized by adapting to the task which has the velocity aligned with the reaching velocity given by f but in the opposite direction. This can be visualized in Fig. 13.

The adaptation rule for beliefs derived from this cost function can be calculated as follows.

$$\dot{\hat{b}}_i = -\epsilon \frac{\partial J_1}{\partial b_i} = \epsilon f(x - \bar{x}_i)^T \dot{\bar{x}}_i \tag{7}$$

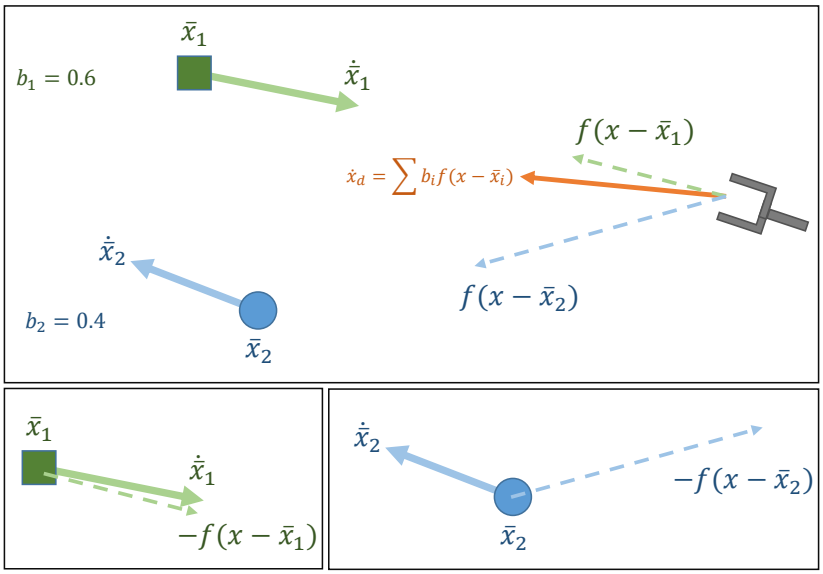


Figure 13: Example of application for the 1^{st} adaptation rule. It can be seen in the first image that two tasks have different velocities $\dot{\bar{x}}_1$ and $\dot{\bar{x}}_2$. They also have different beliefs $b_1 = 0.6$ and $b_2 = 0.4$. The desired velocity coming from the motion generation is then the linear combination shown in Eq. 5. In the two images below the cost function application can be observed: for the task 1, the task velocity $\dot{\bar{x}}_1$ is compared with the opposite of the reaching velocity given by f and the same is done for task 2. It can be seen that the inner product between these two vectors will be higher for task 1 than for task 2. As such the belief of task 1 will increase and the belief for task 2 will decrease

Adaptation rule 2 The first adaptation rule, once implemented, showed some weaknesses: in particular, the main limit was that it would not take into account the distance of the tasks from the robot, but only their velocities. For example, for stationary objects ($\dot{\bar{x}}_i = 0$) the first adaptation would lead to no belief updates. This brought to design a new adaptation rule which would take the distance aspect into account.

Consider the following cost function

$$J_2 = \sum_{i=1}^{N} b_i \, ||\dot{\bar{x}}_i + f(x - \bar{x}_i)||^2 \tag{8}$$

This cost function, as well as the previous one, minimizes the dissimilarity between the object's velocity ($\dot{\bar{x}}_i$) and its expected velocity ($-f(x - \bar{x})$).

The adaptation rule for beliefs derived from this cost function can be calculated as follows

$$\dot{b}_i = -\epsilon \frac{\partial J_2}{\partial b_i} = -\epsilon ||\dot{\bar{x}}_i + f(x - \bar{x}_i)||^2 \tag{9}$$

As such, for static tasks ($\dot{\bar{x}}_i = 0$), the new cost function leads to $\dot{b}_i = -\epsilon ||f(x - \bar{x}_i)||^2$.

Adaptation rule 3 Both previous methods were found to have some strengths and weaknesses. As such, it was decided to test an approach that would combine both methods.

By expanding J_2, different terms can be found.

$$J_2 = \sum_{i=1}^{N} b_i \ (||\dot{\bar{x}}_i||^2 + ||f(x - \bar{x}_i)||^2 + f(x - \bar{x}_i)^T \dot{\bar{x}}_i) \tag{10}$$

The minimization of the first term requires the object to move slowly, as the minimum of the norm of the vector would be 0. The second term requires the generated reaching motions to be slow (in other words, the object to be close if $f(x) \propto ||x||^2$). Finally, the third term is equivalent to J_1. From the second term derives that fast objects, even if approaching the robot, will not be recognised. Since it is not desirable for the adaption to be influenced by the objects velocities (i.e., $||\dot{\bar{x}}_i||^2$), a final cost function was proposed as follows:

$$J_3 = \sum_{i=1}^{N} b_i \ (f(x - \bar{x}_i)^T \dot{\bar{x}}_i + \beta ||f(x - \bar{x}_i)||^2) \tag{11}$$

where β controls the influence of the task-target positions.

The corresponding adaptation rule is as follows.

$$\dot{\bar{b}}_i = -\epsilon \frac{\partial J_3}{\partial b_i} = -\epsilon (f(x - \bar{x}_i)^T \dot{\bar{x}}_i + \beta ||f(x - \bar{x}_i)||^2) \tag{12}$$

2.3 Matlab code

In order to prototype the adaptation mechanisms rapidly, Matlab 2017b was used. More precisely, the video tracking of the tasks and a simulation of the robot's decisions and motion was implemented in order to test and validate ideas that were discussed during development. Moreover, the vision element was implemented in 2D. This can be defended by the fact that the RGB camera used in the experiments would be set up with a top down view and could thus extract the position of the task. As such, this 2D prototype remains similar to the real robot's setup in the sense that a proper inverse kinematic would remain to be used in order to replicate the results on the hardware.

Initially, videos of possible scenarios were recorded by a RGB camera with varying resolutions (240p 4:3 30fps, 480p 4:3 30fps and 720p 16:9 30fps). These videos were constituted of a human subject holding two red cubes in a such a manner that only the cubes and the hands holding them were visible, positioning them on the left side of the image. The cubes were then moved in a way deemed interesting for the purpose of the project. a selection of these motions are reported in Fig. 5.

In the `image processing` module, the videos were read frame by frame and a simple image processing filter, described by Eq. 13, was used.

$$justRed = r - 0.5 * (g + b); \tag{13}$$

where

justRed is the resulting image containing the red intensity,

r, b, g are respectively the red, blue and green channels of the original frame.

This filter would extract the relevant coloured information, red in this case. By using a threshold, the image was then transformed in a black and white binary image, which would contain white blocks in the place of the red cubes. The threshold was dependent on the light conditions, as such it was tuned empirically. The best results were given by a threshold of 70% of the image after filtering ($justRed$).

A blob detection algorithm, given by `regionprops` in Matlab 2017b, was then used to find the center position of the blobs x_i. In order to obtain better results with the blob detection, it was deemed necessary to filter the image again, by applying an "open" operation following Eq. 14. This would reduce the number of small white spots that could be found by the blob detection algorithm.

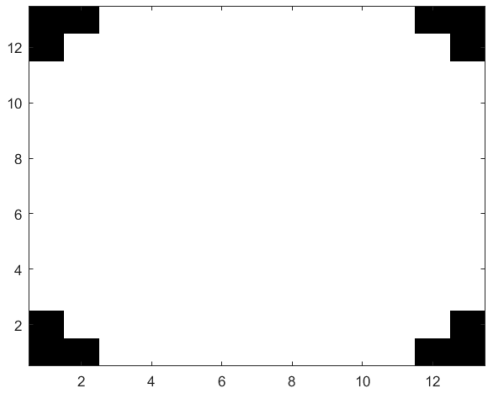


Figure 14: Morphological structuring element

$$OUT = BW \circ SE \tag{14}$$

where

OUT is the output of the filter,

BW is the black and white image before filtering,

SE is the chosen morphological structuring element, which can be seen in Fig. 14.

Thenceforth, a `dynamical processing` module took care of all the dynamical calculations and was successfully run in real time. In particular, the layer handling applied a linear low pass filter to the positions of the tasks $\bar{x}_i$. Filtering the tasks positions was deemed essential as it would guarantee the continuity of the tracking even in the case where the tasks would disappear for a short or long time. Indeed, if the tasks were to suddenly disappear from the video, the filtered positions $\bar{x}_i$ would behave independently, by going outside the limits of the workspace after a certain delay.

In the `dynamical processing` module, the different adaptation rules presented in Sect. 2.2 were implemented by the `adaptation function` module. Finally the `dynamical processing` module used all the informations to calculate the trajectory and velocity of the robot end-effector. In particular the robot velocity was saturated in order to avoid violent spikes in its movement.

The full code structure is resumed in Fig. 15 and an example of the implementation is shown in Fig. 16.

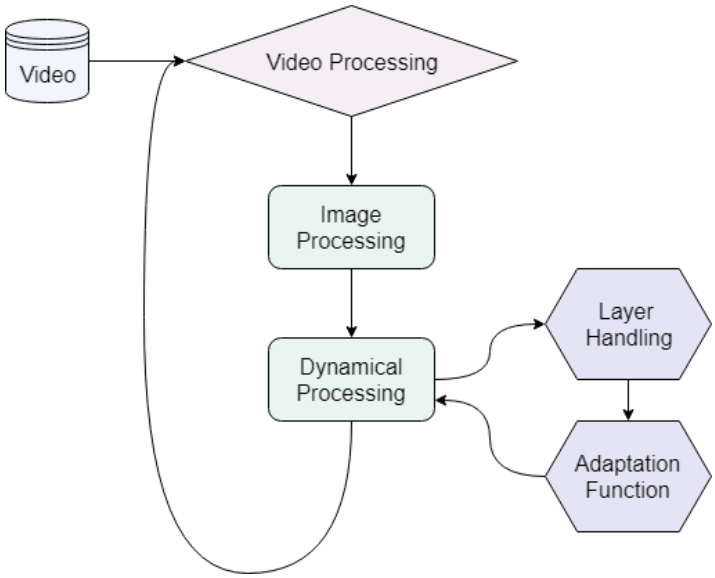
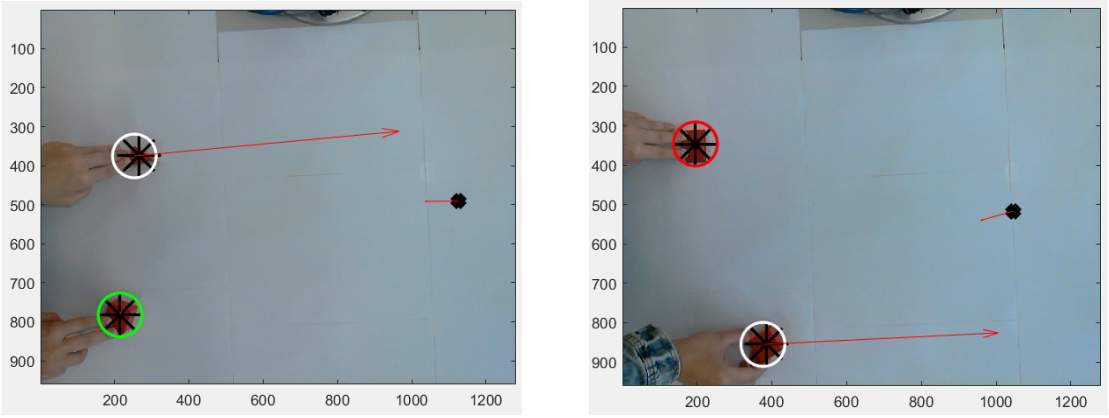


Figure 15: Matlab architecture



(a) Target task is task 1 (in white)

(b) Target task is task 2 (in white)

Figure 16: Example of the Matlab implementation: the two tasks are tracked in position (represented by the black asterisks), their filtered positions are shown (represented by the circles) which also show the associated task colour (red or green) and the targeted task (in white). Their velocities are represented by red arrows. For the robot, its end-effector position is represented by the black dot and its velocity is also shown.

3 Results

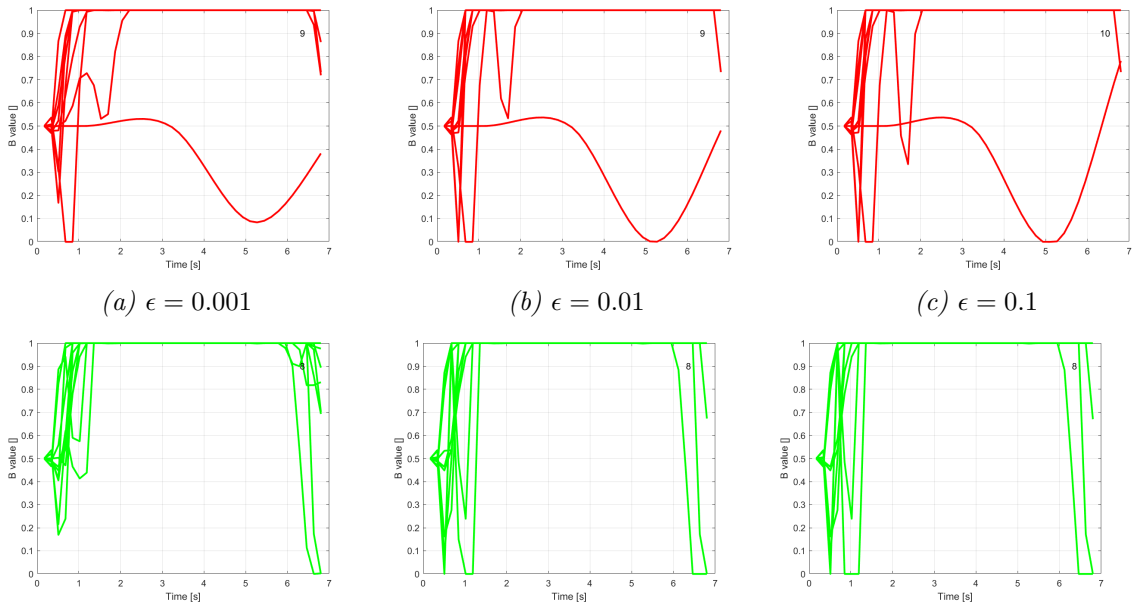
In this section the results obtained using the implementations of the different adaptations described in Sect. 2.2 are shown. The results were obtained by testing the different adaptation rules on a set of benchmarking videos, as described in Sect. 1.2. The results of the tests were collected and the belief corresponding to the task that should be targeted was plotted for each video. The graphs were divided depending on the side of the winning task (left or right), for the complexity of the hand-over (simple or with correction) and for the corresponding adaptation rule (1, 2 or 3), coinciding respectively to Eq. 7, 9 and 12.

Moreover, the free parameters of these adaptation rules were also tested. For algorithm 1 and 2, the only parameter to test was ϵ, which acts as the *adaptation rate*. The values for which it was tested were $[0.001,\ 0.01,\ 0.1,\ 0.5]$.

For the 3^{rd} algorithm, two free parameters were tested: ϵ and β. More precisely, for each ϵ value as mentioned previously, β was tested for the following values: $[0.5,\ 2,\ 10]$.

3.1 First adaptation rule

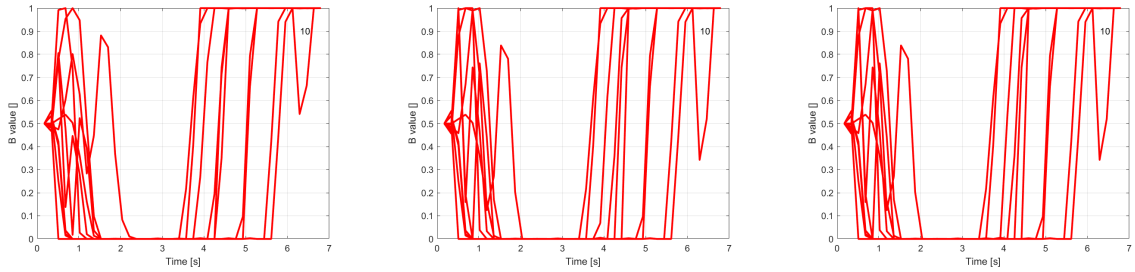
The first adaptation rule corresponded to the formulation described in Eq. 7. The results of this implementation are shown in Fig. 17 for a simple hand-over and in Fig. 18 for a hand-over with correction. In particular, the beliefs for the task to be targeted were plotted.



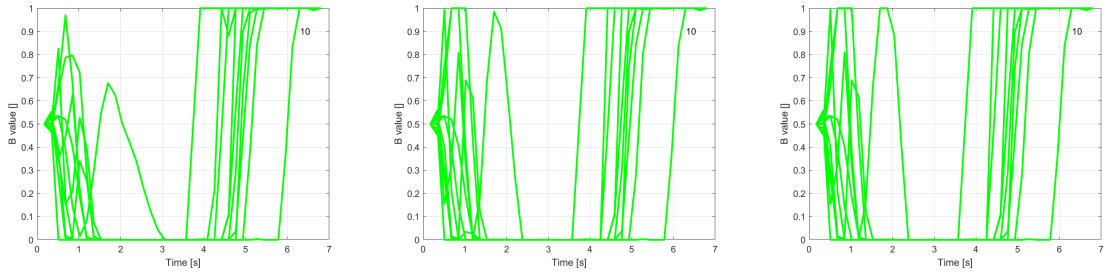
(a) $\epsilon = 0.001$ *(b)* $\epsilon = 0.01$ *(c)* $\epsilon = 0.1$

(d) $\epsilon = 0.001$ *(e)* $\epsilon = 0.01$ *(f)* $\epsilon = 0.1$

Figure 17: Results for left (top row) and right (bottom row) simple hand-overs using the first adaptation rule



(a) $\epsilon = 0.001$ *(b)* $\epsilon = 0.01$ *(c)* $\epsilon = 0.1$



(d) $\epsilon = 0.001$ *(e)* $\epsilon = 0.01$ *(f)* $\epsilon = 0.1$

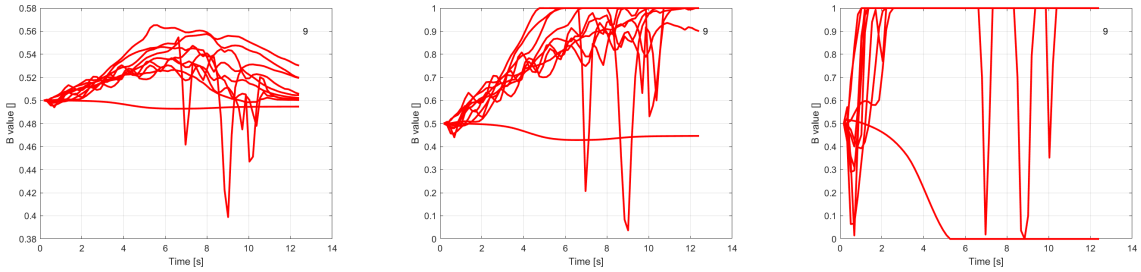
Figure 18: Results for left (top row) and right (bottom row) hand-overs with correction using the first adaptation rule

It can be seen that the results are quite promising. Indeed, the beliefs of the targeted task should tend to 1. In the simple scenario, it is expected for the beliefs to directly converge to 1, while for the scenario with correction it is expected to see a switch in the beliefs, which should first tend to 0 and after the correction converge towards 1. It is also expected that the switch would happen promptly.

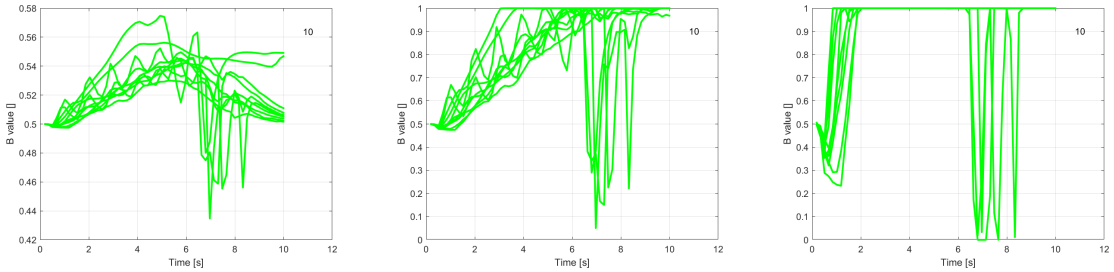
It can be seen here that for the simple scenario the beliefs quickly stabilize to 1 most of the time. As for the second scenario, the beliefs seem to switch correctly and quite rapidly, which is the wanted behaviour. Nevertheless, some problems with this method were found and are discussed in Sect. 4.

3.2 Second adaptation rule

The results corresponding to Eq. 9 are shown in Fig. 19 for a simple hand-over and in Fig. 20 for and hand-over with correction.

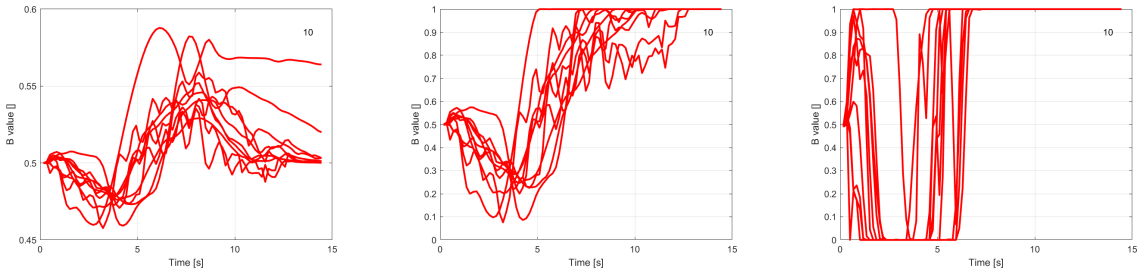


(a) $\epsilon = 0.001$ *(b)* $\epsilon = 0.01$ *(c)* $\epsilon = 0.1$

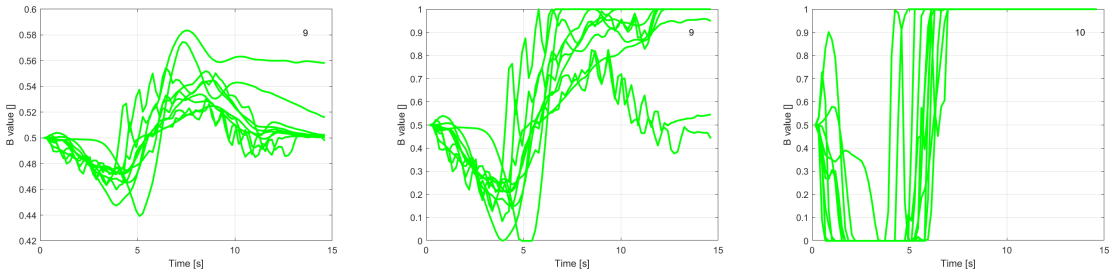


(d) $\epsilon = 0.001$ *(e)* $\epsilon = 0.01$ *(f)* $\epsilon = 0.1$

Figure 19: Results for left (top row) and right (bottom row) simple hand-overs using the second adaptation rule



(a) $\epsilon = 0.001$ *(b)* $\epsilon = 0.01$ *(c)* $\epsilon = 0.1$



(d) $\epsilon = 0.001$ *(e)* $\epsilon = 0.01$ *(f)* $\epsilon = 0.1$

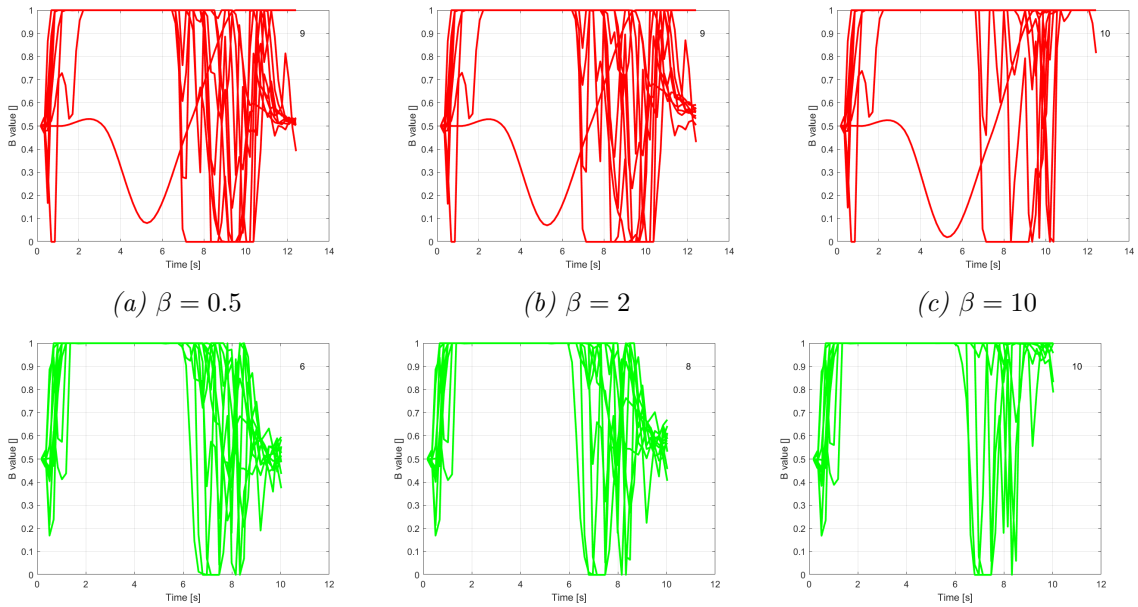
Figure 20: Results for left (top row) and right (bottom row) hand-overs with correction using the second adaptation rule

From the figures above, it can be seen that the behaviour of this adaptation rule is quite different from the previous one. In particular, while it was tested for the same values of ϵ, it

seems more sensitive when higher values are used, while for lower values it does not seem very reactive. This results are discussed more in details in iSect. 4.

3.3 Third adaptation rule

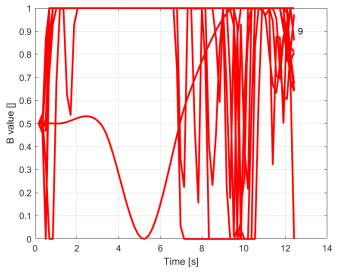
The 3^{rd} adaptation rate was tested for two different parameters: $\epsilon = [0.001,\ 0.01,\ 0.5]$ and $\beta = [0.5,\ 2,\ 10]$. For each value of ϵ, all values of β were tested.



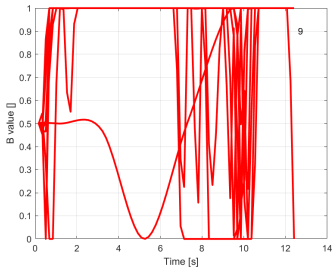
(a) $\beta = 0.5$ *(b)* $\beta = 2$ *(c)* $\beta = 10$

(d) $\beta = 0.5$ *(e)* $\beta = 2$ *(f)* $\beta = 10$

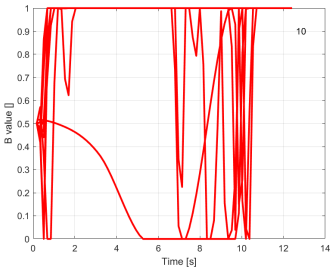
Figure 21: Results for left (top row) and right (bottom row) simple hand-overs using the third adaptation rule with $\epsilon = 0.001$



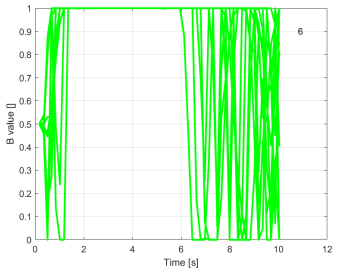
(a) $\beta = 0.5$



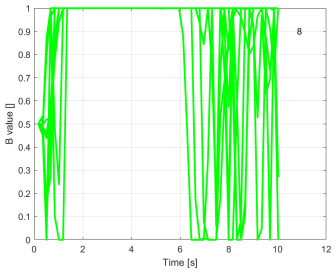
(b) $\beta = 2$



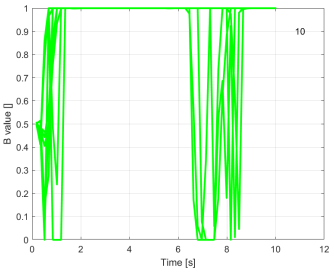
(c) $\beta = 10$



(d) $\beta = 0.5$

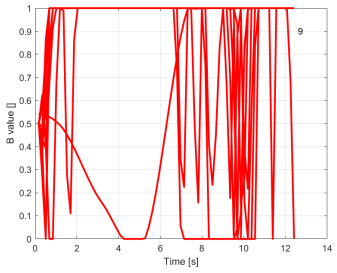


(e) $\beta = 2$

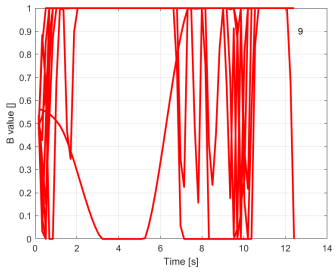


(f) $\beta = 10$

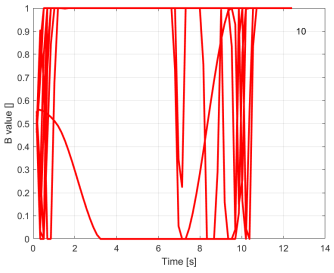
Figure 22: Results for left (top row) and right (bottom row) simple hand-overs using the third adaptation rule with $\epsilon = 0.01$



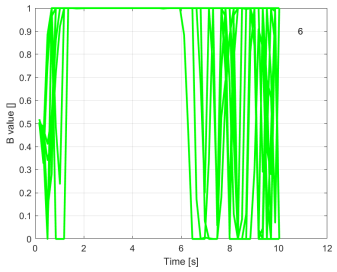
(a) $\beta = 0.5$



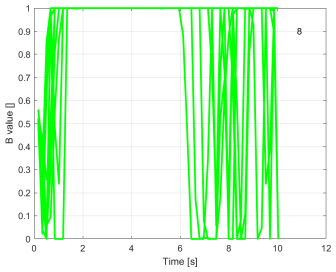
(b) $\beta = 2$



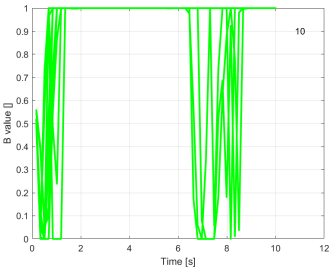
(c) $\beta = 10$



(d) $\beta = 0.5$

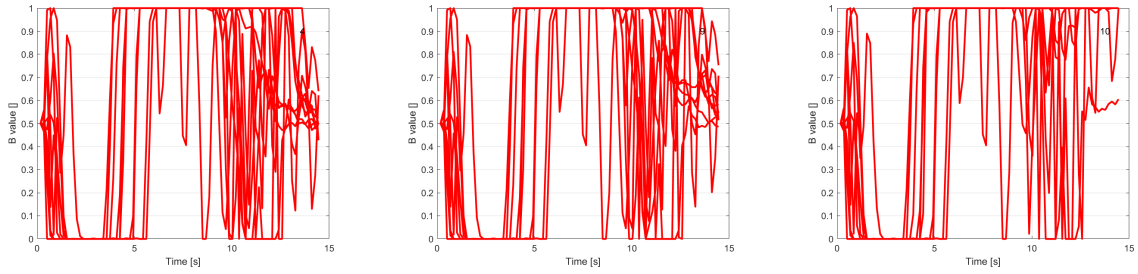


(e) $\beta = 2$

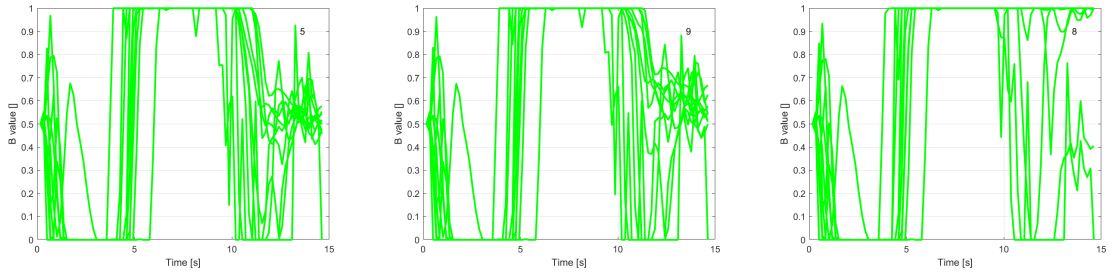


(f) $\beta = 10$

Figure 23: Results for left (top row) and right (bottom row) simple hand-overs using the third adaptation rule with $\epsilon = 0.5$

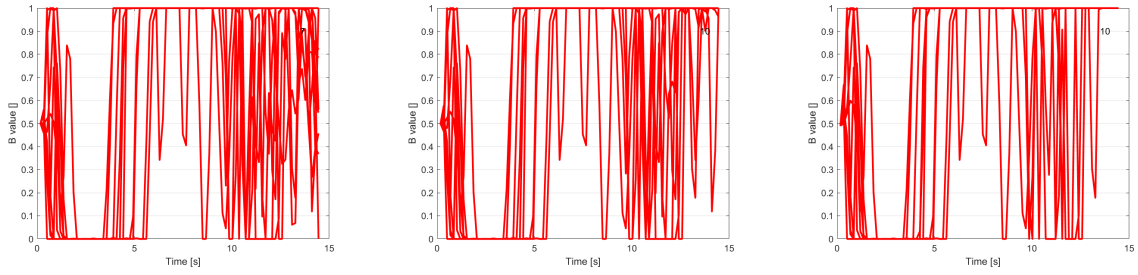


(a) $\beta = 0.5$ (b) $\beta = 2$ (c) $\beta = 10$

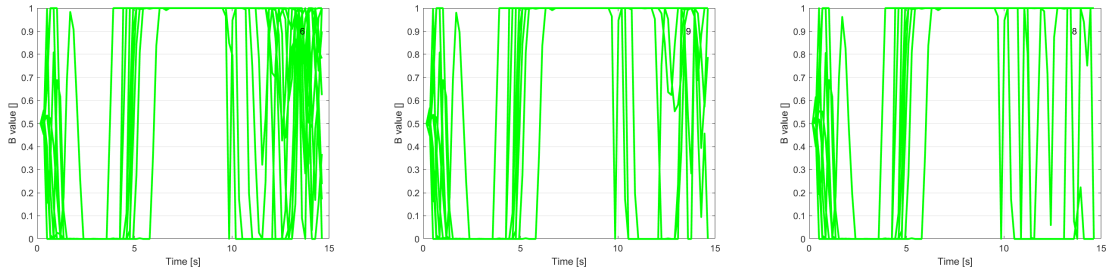


(d) $\beta = 0.5$ (e) $\beta = 2$ (f) $\beta = 10$

Figure 24: Results for left (top row) and right (bottom row) hand-overs with correction using the third adaptation rule with $\epsilon = 0.001$

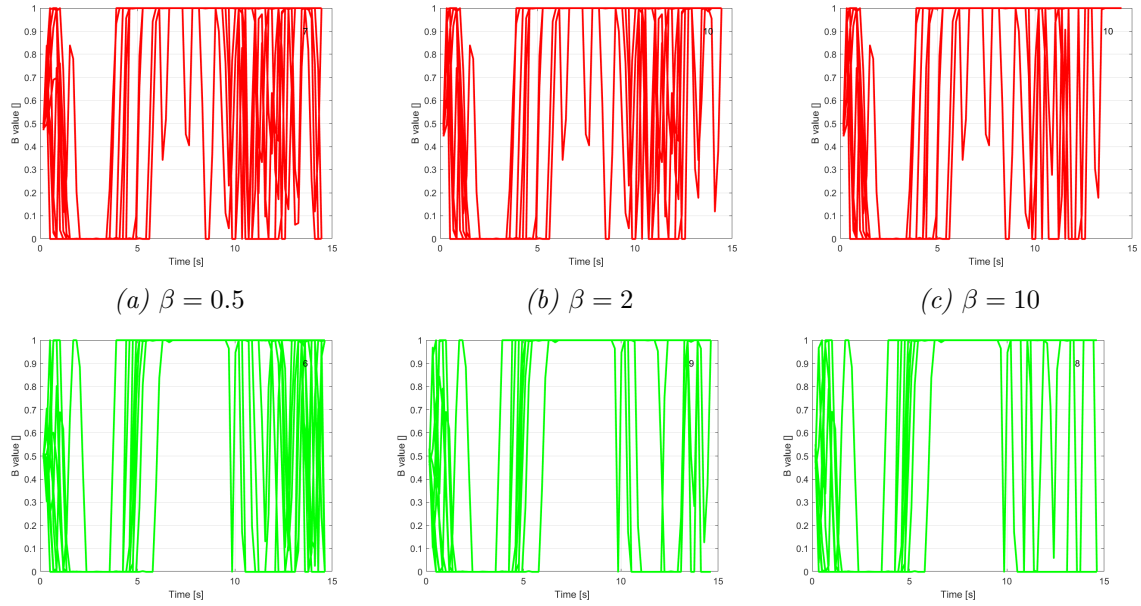


(a) $\beta = 0.5$ (b) $\beta = 2$ (c) $\beta = 10$



(d) $\beta = 0.5$ (e) $\beta = 2$ (f) $\beta = 10$

Figure 25: Results for left (top row) and right (bottom row) hand-overs with correction using the third adaptation rule with $\epsilon = 0.01$



(a) $\beta = 0.5$ *(b)* $\beta = 2$ *(c)* $\beta = 10$

(d) $\beta = 0.5$ *(e)* $\beta = 2$ *(f)* $\beta = 10$

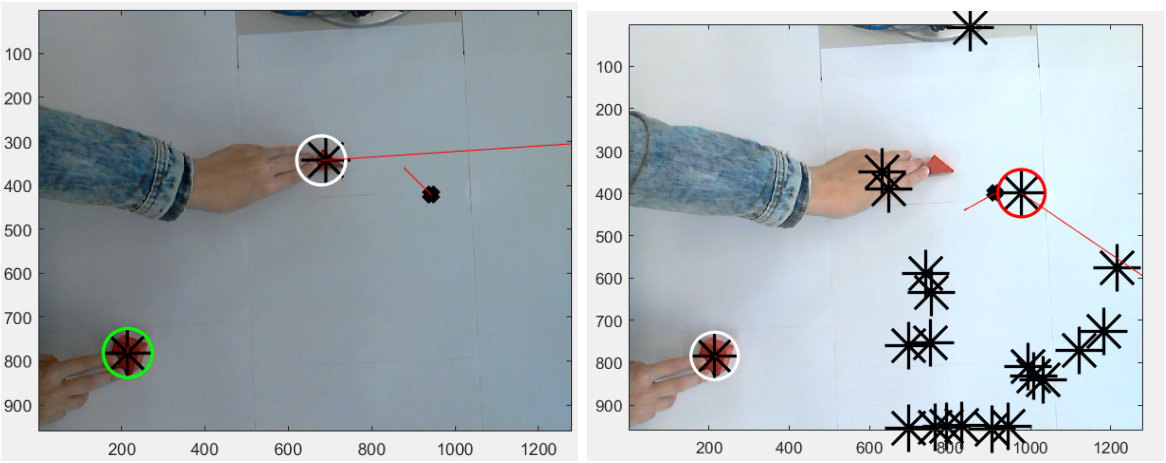
Figure 26: Results for left (top row) and right (bottom row) hand-overs with correction using the third adaptation rule with $\epsilon = 0.5$

Here above the results for the 3^{rd} adaptation rule are shown. Different parameter values were tested, although these results do not seem to vary as much as one could expect. Moreover, it did not seem possible to reduce the noise on the beliefs. Further discussion about the results and what could be needed to obtain a more performant results is presented in Sect. 4.

4 Discussion

The results shown in Sect. 3 showed that a method to enable visual communication between a human subject and a robot through direct task manipulation was successfully developed and implemented. With regards to performance, the results were satisfactory but may be improved.

One difficulty encountered early on was that the benchmarking videos were often changing luminosity violently (as can be seen in Fig. 27), which was probably due to incorrect settings of the webcam they were recorded with. In order to strengthen the implementation against noisy surroundings, these settings were used nevertheless. In order to deal with task detection noise, the tasks positions are filtered and stored. At each frame iteration, the distance between the previous filtered positions and the new raw positions is calculated and each raw position is assigned to the closest filtered position.



(a) Before luminosity change *(b) After luminosity change*

Figure 27: Example of noisy blob detection

In Sect. 3.1, the results of the first adaptation rule are shown. It can be noticed in Fig. 17 and in Fig. 18 that most of the time the beliefs are correctly identifying the task that should be targeted, although it can be observed that for one specific case, the behaviour was always incorrect.

It can also be seen that the beliefs are very reactive and quickly switch and saturate. This can be noticed especially in the correction scenario, where the switch from 0 to 1 is very fast.

Nevertheless, a problem was encountered while testing the 1^{st} adaptation law on Matlab: in both the simple and complex scenario, the robot would choose correctly the task to approach in the beginning. But as it got close to the task with a certain speed, the robot would pass the task position and continue to pursue the other task after switching the belief. This is due to two main factors:

- the adaptation rule in this case depended only on the velocities of the tasks, as such their position was not considered. This explained why the robot, when surpassing the

task, would switch belief, as in the robot's referential the first task velocity would have switched sign.

- the adaptation rules were tested on pre-recorded videos, which can be compared to testing in "open-loop". More precisely, there was no way to adapt in real time to the robot's behaviour, which would be possible in a real setting.

This effect could be avoided by cutting the last second of the simulations or by reducing the velocity of the robot by lowering the saturation threshold. Nevertheless it can be considered a weakness of this adaptation rule.

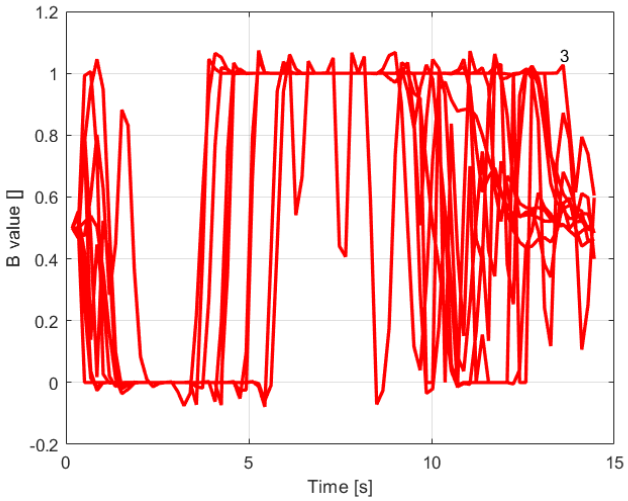
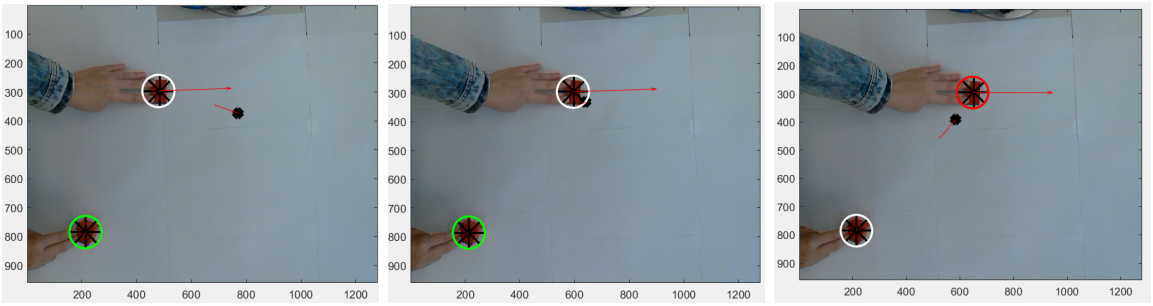


Figure 28: Example of complete plot of beliefs. It can be noticed that in the last part, the beliefs tend to return to 0.5



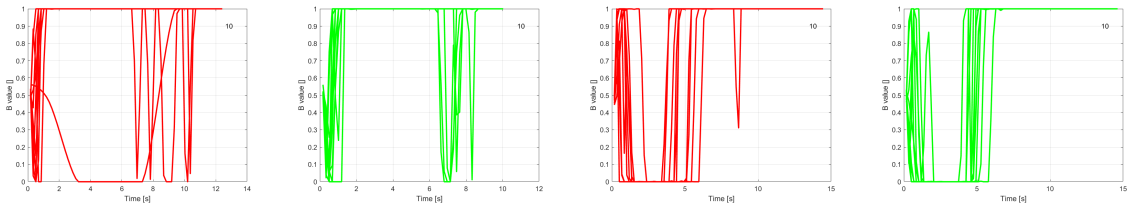
(a) Step 1 *(b) Step 2* *(c) Step 3*

Figure 29: Example of "bad" behaviour for the first adaptation rule. As the task adaptation depends on the tasks velocities, the robot sees the target task velocity switch direction when it surpasses the task position. It then changes belief and starts moving towards the other task

In Sect. 3.2, the results for the second adaptation rule are shown. It can be seen in Fig. 19 and in Fig. 20 that, although the results seem less noisy compared with the previous ones,

the decision-making is in this case slower to attain high values of belief. It can also be seen that the parameter ϵ influences this effect quite dramatically. The solution to a slow decision making was then to increase the ϵ parameter to 0.1, which then gave results similar to the ones from the previous adaptation rule. It can also be noticed that some results of the simple scenario are quite noisy after stabilization. This is mainly due to the detection noise, which as shown in Fig. 27 sometimes causes random variations of the task position. This could be improved by checking that the task position is not moving further than a certain distance at every iteration.

In Sect. 3.3, the results for the third and final adaptation rule are shown. In Fig. 21, 22 and 23 results for the simple scenario can be observed. It can be noticed that there is similarity with the results from the first adaptation rule. It also seems that by increasing either parameter, the noisiness of the beliefs does not improve greatly. Nevertheless, increasing β seems to have an effect mainly at the end of the simulation, by helping the robot to converge on the closest task. On the other hand, increasing ϵ unstabilizes the results, which become noisier. Similar observations can be made for the Fig. 24, 25 and 26 which correspond to the correction scenario. To improve the performance of this rule, another test was made with $\beta = 100$ and $\epsilon = 0.01$, whose results can be seen in Fig. 30. This results seem to have less noise and to have clear and fast decision-making, as such it can be convened that the 3^{rd} rule should be further tested for higher values of β and/or lower values of ϵ.



(a) Simple left *(b) Simple right* *(c) Correction left* *(d) Correction right*

Figure 30: The 3^{rd} adaptation rule tested for $\epsilon = 0.01$ and $\beta = 100$

Success rate Here below the success rates for each algorithm in function of the parameters tested are reported. In order to obtain them, the number of correctly classified target tasks (cf. the number of beliefs higher than 0.5) was counted for each method. In Table 1, 2 and 3 the success rate percentages are reported. It can be observed that the best performance is given by the second adaptation rule, although the score of the third adaptation rule is lowered by parameters that give worse results ($\beta = 0.5$ in particular). This reconfirms the importance of considering distance between the robot and the tasks in the optimization.

It is worth noticing that the success rate is not the only valuable performance. The rapidity of the switch of belief when the target changes and the quantity of noise in the belief are also important criteria. For the prompt response to target changes, the three rules can be quite fast, although the second rule needs higher ϵ values to achieve similar results as the others. As for the noise, the most stable seems to be the second one based on the results obtained.

As discussed above, further tests should be conducted on the three rules, particularly on the final one, as other parameter values can be found which could have promising results.

	$\epsilon = 0.001$	$\epsilon = 0.01$	$\epsilon = 0.1$	
Adaptation rule 1				
Simple	85.0%	85.0%	90.0%	86.7%
Correction	100.0%	95.0%	95.0%	96.7%
	92.5%	90.0%	92.5%	91.7%

Table 1: Success rate for adaptation rules 1

	$\epsilon = 0.001$	$\epsilon = 0.01$	$\epsilon = 0.1$	
Adaptation rule 2				
Simple	95.0%	95.0%	95.0%	95.0%
Correction	95.0%	95.0%	100.0%	96.7%
	95.0%	95.0%	97.5%	95.87%

Table 2: Success rate for adaptation rule 2

	$\epsilon = 0.001$	$\epsilon = 0.01$	$\epsilon = 0.5$	
Adaptation rule 3				
Simple				
$\beta = 0.5$	75.0%	75.0%	75.0%	75.0%
$\beta = 2$	85.0%	85.0%	85.0%	85.0%
$\beta = 10$	100.0%	100.0%	100.0%	100.0%
Correction				
$\beta = 0.5$	45.0%	65.0%	65.0%	58.3%
$\beta = 2$	90.0%	95.0%	95.0%	93.3%
$\beta = 10$	90.0%	90.0%	90.0%	90.0%
	80.83%	85.0%	85.0%	83.6%

Table 3: Success rate for adaptation rule 3

5 Conclusion

In this project, the recognition of human intentions in ambiguous situations was studied. In particular, the project aimed to develop an approach that would allow to communicate with a robot exclusively through visual interaction through task manipulation.

In order to delineate the scope of the project, a hand-over situation was studied, where a human subject would offer one of two tasks to a robot. Two cases were considered: a simple one where the task would be offered directly; a more complex one where the task offered initially would be withdrawn and the second task would be then offered to the robot.

As such, a decision-making method was developed through the use of dynamical system based mechanisms. It was then possible to test different approaches by implementing a simulation on Matlab. This implementation would receive a pre-recorded video, track the tasks and generate a motion by adapting to the tasks movement.

To attain a coherent behaviour, a set of pre-recorded videos was created and different adaptation rules were tested. All the rules were capable of discerning human intention, to different degrees of success. In particular, the second adaptation rule was able to achieve a success rate close to 100%. The pre-recorded videos contained unwanted changes in luminosity as well: this gave the opportunity to test the robustness of the implementation, which was quite satisfactory but can be improved.

Future work As discussed above, the final adaptation rule should be tested on different parameter values from those shown in the Results section, as it showed promising results that could be further optimized. A next stage could be to find a systematic tuning optimization, possibly by using machine learning techniques.

The Matlab implementation was successfully run in real-time during simulation, nonetheless the next step would be to adapt the complete algorithm to the ROS framework, although some steps were already made in that direction during this project.

Moreover, the vision processing could also be improved, by assuring the continuity of the task position tracking and the ability to consider a higher number of tasks.

References

[1] T. Triquet, M. Khoramshahi, and A. Billard, "Task-adaptation for assistive robotics using switching dynamical systems," 2017.

[2] A. Laurens, M. Khoramshahi, and A. Billard, "Adaptive human-robot interaction: From human intention to motion adaptation using parameterized dynamical systems,"

[3] M. Khoramshahi and A. Billard, "A dynamical system approach to task-adaptation in physical human-robot interaction," *Autonomous Robots*, pp. 1–20, 2018.