Lazy Control of an Anthropomorphic Robotic Arm

José Avelar, René van de Molengraft, Herman Bruyninckx

***Abstract*— The classic control approach used for industrial robot manipulators implicitly assumes a known, controlled, and predictable environment. However, these assumptions cannot be guaranteed for robots operating in uncertain and dynamic environments. The development of service robots, for instance, requires control implementations that allows them to robustly execute tasks under these conditions. In this paper, we introduce an approach for designing a lazy control strategy that provides robustness towards specific changes and uncertainties in dynamic environments which are not considered in the classic control approach. The methodology is applied for a 7-DOF anthropomorphic robotic arm to perform the task of opening a drawer. It exploits knowledge of the task context to design a lazy control strategy and controller that does only as much as it necessary to achieve the task.**

I. INTRODUCTION

The success of robot manipulators in the industry and the technological advances from recent years have increased the interest for developing commercial robots. These type of robots have the potential of increasing productivity in many sectors never before considered. Service robots, for instance, have the potential of assisting humans in performing repetitive and mundane household tasks.

Unfortunately, taking robots from an industrial to a household environment still has many unresolved problems. The unpredictable nature of a household environment makes it difficult to program robots for executing tasks. Moreover, accidental or deliberate human interaction is expected to happen. Safety must be guaranteed for people sharing the workspace. As a result, methods that have been applied for years in the industry cannot be used for programming robots in this context. Manufacturing processes focus on meeting strict quality and efficiency requirements in highly controlled and engineered workspaces to guarantee an ideal execution. These spaces are then modelled for tuning controller parameters offline with the goal of creating optimal and precise actions. This methodology, naturally, is not well suited for dealing with changes and uncertainties in the environment.

Robot manipulators are an example of industrial robots controlled for precise and optimal motions. This is generally achieved by solving the classic *Tracking and Disturbance Rejection Problem*. It consists on determining the control inputs necessary to follow a desired trajectory, while simultaneously rejecting disturbances due to un-modelled dynamic effects. To do so, a kinematic and dynamic model of the robot are derived. PD and PID controllers are the preferred and most common type of control algorithms used. A feedforward controller may also be included to improve tracking and disturbance rejection when these disturbances can be modelled. These control techniques have been well studied in the literature, e.g., [1] [2], and are very effective at tracking and rejecting disturbances.

However, applying these methods for controlling service robots in dynamic environments can result in unwanted and even dangerous behaviour for the robot, and especially, any humans in close contact with it. In this context, any unplanned behaviour, including human interaction, is seen by the controller as a disturbance that needs to be eliminated. Pushing the robot away from its programmed path would result in the robot pushing back. This is not the type of response one would expect a service robots to have.

Nevertheless, recent studies have developed methods for safe human-robot interaction by giving the robot a compliant behaviour when reacting to external forces. In [3] and [4], an impedance control approach that safely shapes the interacting forces between the robot and its environment is proposed. Furthermore, [5] uses this method combined with position and force control to safely deal with unexpected collisions with dynamic obstacles while moving in an unstructured environment. However, these techniques are applied considering implicit assumptions, intrinsic to the classic control paradigm, that do not always hold in dynamic and uncertain environments where service robots operate.

It can be argued that the controller's ability to achieve a task in these conditions is constrained by the following limitations: ***i)*** the available information of the workspace, and ***ii)*** sensor data quality. Addressing these limitations requires to re-evaluate predefined conditions embedded in the classical control paradigm. Creating robust implementations in scenarios where these assumptions do not hold is linked to the trade-off with reducing performance in terms of its tracking speed and precision [6].

In general terms, the context of household tasks has different performance requirements than manufacturing tasks. It can easily be noted that the former requires lower levels of precision: knowledge of the exact joining point is required when assembling two parts together, but one does not *need* to specify an exact point for grasping a bottle. There is a range of valid points that yield the same result. Similarly, meticulously following a trajectory in time is a fundamental requirement in welding applications but is not as relevant when opening a drawer. Hence, it seems clear that control requirements for household applications can be reduced, while maintaining an acceptable system performance, specified by the given task.

However, a common household task, such as opening a drawer, becomes a non-trivial problem in a dynamic and uncertain environment. This task has been previously studied in the literature considering such conditions. In [7], the author proposes a method for identifying the directions of unknown constraints imposed by a drawer or door to properly orient the mechanism and reduce the required pulling force.

Furthermore, one of the most successful implementations is found in [8] [9], where it estimates a trajectory for the predicted type of joint (prismatic for a drawer or revolute for a door) and uses a form of impedance control to open them. The author accurately proposes to consider the surface of the drawer/door to first identify contact, facilitating grasping its handle.

This paper presents a general lazy control approach for controlling robotic systems in a dynamic and uncertain environment. We use this approach for controlling a 7-DOF anthropomorphic robotic arm (referred in the rest of this document as *the arm*) equipped with a passive "hook-like" end-effector for opening a drawer. Furthermore, we explore the consequences of making the controller *lazy* by:

- Expanding the notion of reference set-point to a *region of motion* by introducing an *error margin*. The system's performance is considered as acceptable if its behaviour is within this region.
- Exploiting knowledge of how the the arm's behaviour is constrained by its dynamics, kinematics, and the task's common workspace to selectively control only the joints that are required to achieve a specific action, while keeping the others in a passive and compliant mode.

The outline of this paper has been organized as follows: Section II generally introduces the lazy control approach. Section III follows by applying this approach for the task of opening a drawer. Next, Section IV implements a lazy adaptive controller for the arm. Section V shows the experiments performed on the arm and the controller's behaviour while opening a drawer. Finally, Section VI concludes with the results of this investigation and discusses future work.

II. LAZY CONTROL APPROACH

Addressing changes and uncertainties in the environment requires to look at alternative approaches for controlling systems that operate under such conditions. The traditional control approach is based on finding an optimal solution to the control problem. It requires a high level of precision and is, therefore, limited to implicit assumptions that generally do not apply in these environments. The lazy control approach relaxes classic control aspects based on the task being performed and the requirements for successfully achieving it. These conditions are defined as the task context and are classified into three different groups:

- *Robot capabilities*: The robot capabilities are defined by the tools and sensors that let the robot interact with the environment in specific ways. The type of actuators, end-effector, force and image sensors, etc. are all examples of these elements. Knowledge of the robot's dynamic and kinematic behaviour is also considered here since it determines how this interaction occurs.
- *Common workspace*: The objects and surfaces in the environment that are always present when performing the task are referred to as the common workspace. These are the elements with which the robot typically interacts while executing a task.
- *Task specifications*: The task specifications contain the plan of how a task can be achieved. Precision requirements for executing it are defined here.

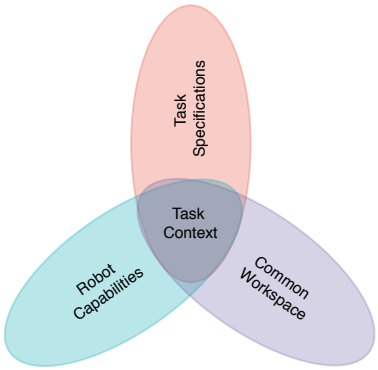

Figure 1: Structure of the task context.

The task context contains all the required information for executing a task considering the robot capabilities and how it is able to interact and be coupled together with the common workspace to obtain the desired behaviour. The insight obtained from this analysis allows us to design a lazy controller by considering relaxing the following control aspects:

Set-point and trajectory tracking Introducing an error margin when tracking a set-point or trajectory expands the notion of a reference point (or path) into a region of motion where the error is considered to be zero. Within this area, the system's performance is considered to be sufficient for achieving the action.

The goal is for the controller to act as open-loop when the error is within the error margin. When this condition is met, the control action is not influenced by the feedback error signal because it is already sufficient to provide the desired system output. However, when the error exceeds this margin the feedback signal is "reconnected" to drive the system back into the desired region of motion.

Controlled system state variables Exploiting knowledge of how the system's behaviour is constrained by its dynamics, kinematics, and the task's common workspace allows to reduce the amount of controlled system state variables.

A controller that is lazy requires less precise information to compute control actions because it does not attempt to find an optimal solution, but rather a solution that is sufficient for the desired performance requirements. Furthermore, it not concerned about changes in system state variables that are not relevant to the action being performed. Nonetheless, that does not imply that the system will not act with precision. An action can result precise by cleverly using the constraints set on the system to guide its motion. Moreover, its behaviour must maintain an acceptable performance, according to the task context specifications, and behave in a safe and predictable manner.

Robots working in a dynamic environment need a way to plan and coordinate actions based on (predictable and unpredictable) events that occur at run-time. A series of situa-

tions that cannot be foreseen offline can occur at any moment. The control system must be able to identify and react to the system's behaviour. An expanded control system diagram is shown in Figure 2, including two additional entities: a Motion Monitor and a Planner. In this research, we limit our analysis to actions performed in a nominal order. A brief description of the basic role of these entities, as used in this paper, is provided. However, in a general case, these entities have a broader role composing and coordinating skills for unplanned non-nominal scenarios. An in-depth analysis of the most common methodologies for coordinating, configuring, and composing robotic behaviours is presented in [10].

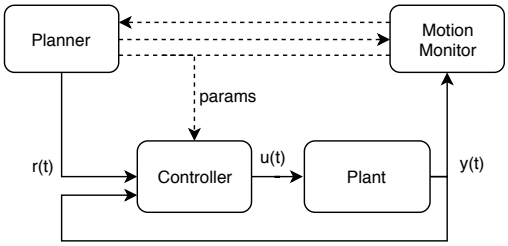

Figure 2: Diagram for the lazy control scheme. The signals $r(t)$, $u(t)$, and $y(t)$ are the reference, control action, and system output signals of the classic control scheme, respectively. The Motion Monitor and Planer influence the behaviour of the Controller by reconfiguring its parameters given different situations. Communication between the Planner requires Motion Monitor occurs in both directions.

The Motion Monitor is added to the control process for monitoring the system's behaviour. It determines if the system is operating as expected or not, and it identifies events that trigger actions based on the available sensor information. The Planner contains information of the execution plan and how it is achieved. Both entities are able to reconfigure the controller's parameters to adjust its behaviour adequately for the current action.

III. OPENING A DRAWER

In this section we implement a control strategy based on the lazy control approach . The system to be controlled is a 7-DOF anthropomorphic robotic arm and the task it performs is opening a drawer. We define the context of the task by gathering information of the robot capabilities, common workspace, and the task specifications. This knowledge is used to design a lazy control strategy, followed by a controller implementation, in Section IV, that will accomplish such behaviour.

i) Robot capabilities The arm mechanism can be simplified, without loss of functionality for this specific task, into a 3-DOF model as shown in Figure 3. This mechanism has a set of revolute joints with parallel axes at the *Shoulder*, *Elbow*, and *Wrist*. The motors controlling the joints are backdrivable and can be set in a high-impedance state that is passive and compliant to external forces, such as gravity and human interaction. When the joint is being actively controlled it is said to be in a *controlled* mode, when it is set in a high-impedance state it is referred to as in *passive* mode.

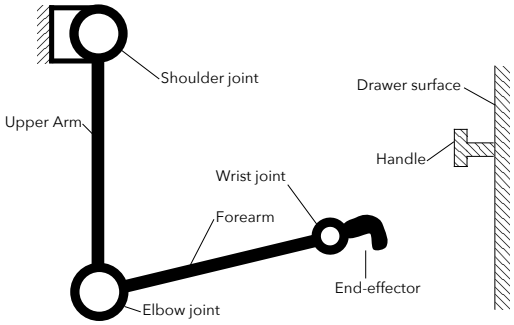

Figure 3: 3-DOF representation of the anthropomorphic robotic arm and the common workspace for the task of opening a drawer. The arm's links and joints are named after their human counterparts. The end-effector is a passive 3D printed hand in a "hook-like" shape."

Furthermore, motors are equipped with position sensors and encoders that compute the motor's velocity in every actuator. No vision capabilities are available. Hence, for the sake of simplicity, it is assumed that an approximation of the handle's height is already available. Finally, the arm is equipped with a passive end-effector: a 3D printed hand in an L-shaped configuration.

ii) Common workspace The workspace elements that are always present when opening a drawer are shown in Figure 3. Naturally, these are the surface and handle of the drawer. The type of handle is just as important as the type of end-effector when determining how the task will be achieved. Here, we consider a drawer with a typical cabinet "pull handle".

ii) Task specifications The task specifications describe how the task is executed and how precise the system needs to be in order to achieve it. One convenient form of describing an execution plan is by using a skill-based approach. This is a modular form of programming robot actions that divides a task into a set of simpler actions, or skills. These skills can then be interchanged and rearranged to achieve other tasks. This results convenient, among other things, since we can focus on a control strategy for each skill individually.

Figure 4: Simple nominal skill decomposition for the task of opening a drawer.

Figure 4 shows a nominal skill decomposition for opening a drawer. The plan is executed as follows: first, the Reach skill requires the arm to drive the end-effector towards the handle. The Grasp skill represents the action of properly attaching the end-effector to the handle. Finally, the Pull skill requires the arm to open the drawer by pulling the handle.

The grasping action can occur at any point along the handle. It is not important to define a precise point of interaction. Precisely controlling the motion of the end-effector is also not required for reaching and pulling the drawer, as long as its behaviour is predictable and safe for humans. As it has been stated before, many other household tasks have these same performance requirements.

The task context brings together the knowledge of the robot's capabilities, the common workspace, and how a task can be executed to design a detailed control strategy for successfully achieving it. When describing a task, it is useful to acknowledge how we, as humans, approach these actions. Sensing the environment, for instance, is an important resource for effectively achieving them. We do not know the precise location of an object's position when we reach for it. Rather, we move our hand in the object's direction, using our sight to react to the motion as it occurs. Then, we use touch to acknowledge we have reached the object before grasping it. The following paragraphs presents a detailed control strategy for each skill, keeping in mind these ideas. A few assumptions need to be made before considering this strategy. We assume that the drawer's global position is known and that the robot is placed at a valid distance in front of it. Note, however, that the handle's position in space and its geometry is not certain since it is not easily measured, but an approximation of its height is given.

- *Reach skill*: The action of reaching the drawer represents a forward motion in space of the end-effector constrained only by the dynamics and kinematics of the system. However, at a certain point, displacement is limited by the drawer's surface. Assuming the drawer is within reach, the end-effector will eventually make contact if it keeps moving forward. Making contact guarantees that the end-effector is indeed at the drawer's exact location without the need of precisely defining it offline.

  Notice that under the previous conditions, the only information from the workspace that is required by the controller to successfully execute the skill is to define where contact should be made. For convenience, we are interested in controlling the arm to place the end-effector above the handle. However, precise positioning is not a requirement: there is no motivation for defining a specific point in the surface where this should happen. Instead, we argue that there is a region above the handle where making contact is equally valid, as seen on Figure 5. This region is limited vertically by the handle's height and the available surface above it; and, if considering a 3-dimensional analysis, by the handle's width.

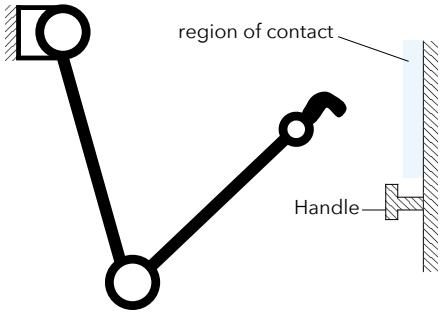

Figure 5: Illustration of the Reach skill. The area shaded in blue represents the region where the end-effector is expected to make contact with the surface. Any point in the surface within this region is equally valid for successfully executing the skill.

- *Grasp skill*: Attaching the end-effector to the handle can be conveniently achieved by using it as a hook. Assuming the previous skill has been executed successfully, the end-effector can be hooked into the handle by simply sliding it down along the surface. This is achieved by setting the *Elbow* joint to passive mode and allowing gravity to manipulate it. The *Forearm* and end-effector, coupled between the *Upper Arm* and the drawer's surface, will slide down with a backwards motion of the arm (away from the drawer) generated by the *Shoulder* joint.

- *Pull skill*: The transition between the Grasp and Pull skill occurs seamlessly, since the same backwards motion continues for this actions. The "pulling" behaviour happens when the end-effector makes contact with the handle. This coupling translates the drawer's constraints to the arm and restrict its movement to only one possible direction. Then, the *Forearm* can "rest" on the handle and act as a passive coupling link between the *Upper Arm* and drawer.

Skills are triggered by events that occur when the arm interacts with the environment. The controller's behaviour depends on what skill is being executed. For this task, the controller's behaviours are reduced to two: a forward motion, when executing the Reach skill, and a backwards motion, for both the Grasp and Pull skills. Making contact with the drawer's surface triggers the change. This event can be identified by using the position sensors in the *Wrist* joint and noticing its displacement as consequence of the interaction. Therefore, the control scheme of Figure 2 results useful for monitoring and managing these changes.

IV. CONTROLLER IMPLEMENTATION

We now explore how the arm can be *lazily* controlled to achieve the task while providing the sufficient performance considering its context.

The first point to acknowledge is that the arm's motion does not need to be described by a time-dependent trajectory. Instead, it can be defined with a series of guarded motions that are triggered by specific events. Here, the *Shoulder* joint is used as the "driver" of the motion, pushing the arm forward or backwards. The *Elbow* joint, on the other hand, continuously adjusts the end-effector's height to the desired region of motion as a result of the movement. Hence, the end-effector's trajectory is not a pre-planned input to the system, but rather a result of how the joints respond to a given input signal while being constrained by its dynamics, kinematics, and the environment.

The next point to consider is that there is no need to continuously control all joints during the complete execution of the task. Precisely orienting the end-effector during execution, for instance, does not require active control. A compliant behaviour of the *Wrist* joint allows the end-effector to accommodate to the surface as it is pushed against it. This is also convenient as it adds damping to the motion on impact. In addition, when pulling the drawer, the *Forearm* rests its weight in the handle and acts as a passive link adjusting itself to the interaction. Its motion and the end-effector's position are of no concern and do not need to be controller, since the constraints (set by the handle and drawer) will guide them as necessary.

The last point to consider is that the controller must be able to adapt to workspace changes and uncertainties. One example is friction. Its effects on the system while pulling the drawer are unknown and may vary depending on the drawer, or even as the drawer is being pulled. Therefore, the forces that occur from this interaction are not known prior to the action.

Based on the previous considerations, we propose an adaptive controller that computes the driving torque by monitoring and reacting to the system's behaviour. It does not require knowledge from the system's parameters or a pre-planned trajectory to move the arm. The controller adapts its control action to compensate for un-modelled system dynamics, including coupling effects among the links, and to the unmodelled interaction with the workspace, i.e., forces resulting from pulling the drawer.

The adaptive law is based on the *Adaptive-Bias/Adaptive-Gain* algorithm proposed in [11]. It keeps track of the overall sign of the error, defined here as the *low-passed filtered error sign* or $\bar{e}_k$, to compute the control action. The algorithm has been modified to consider a set-point tracking error margin, represented by σ. As stated in Section II, this results in expanding the notion of set-point into a region of motion. $\bar{e}_k$ stabilizes around the set-point, when $|\bar{e}_k| < \varepsilon$, where ε is a stabilizing threshold. Once stable, its value remains as zero until the system exits the region of motion. Algorithm 1 shows the controller's logic, including these modifications.

Here, `hside(.)` refers to the Heaviside function defined as 0 for negative numbers, 1 for positive. The algorithm is composed of two main steps. In the first step, the error is evaluated. If $\bar{e}_k$ is stable and the system is within the error margin, $\bar{e}_k = 0$. Else, $\bar{e}_k$ is updated by applying a low pass filter to the sign of the instantaneous error value. In case of the latter, this value is compared to the stabilizing threshold to determine if $\bar{e}_k$ has been stabilized. If that is the case, its value drops to 0 and remains as such until the error is larger than σ.

The second step updates the Adaptive Bias term b_k and Adaptive Gain term g_k. Their values are increased or decreased at a specific rate, given by δ_b and δ_g respectively, and depending on an adaptation threshold defined by $\bar{e}_b$ and $\bar{e}_g$. Finally, the control action scaling factor u_k is computed as follows:

$$u_k = \texttt{sat}_{[-1,1]}(g_k + b_k) \tag{1}$$

It is important to remark that when $\bar{e}_k = 0$, b_k remains constant and g_k becomes 0. The resulting value for u_k scales the maximum allowed torque τ_{max}. This value is chosen considering the type of skill being performed and safety requirements given by the task context. The resulting driving torque is calculated by:

$$\tau = \tau_{max} u_k \tag{2}$$

Figure 6 shows the region of motion, shaded in blue, given a desired set-point configuration q^d and an error margin σ. In this simple example, the controller attempts to rotate the *Elbow* joint towards q^d. Once $\bar{e}_k$ has been considered stable, its value drops to 0 and remains as such while the system is within the region of motion. Inside this area, the controller acts as open-loop since feedback measurements are not considered for computing the control action. The feedback signal is "reconnected" when the joint exits the region of motion. As this occurs, the controller will update $\bar{e}_k$ and the control variables, attempting to stabilize $\bar{e}_k$ again.

Algorithm 1 ABAG Algorithm

Input: $y_k \in \mathbb{R}$ % measured output
$y^d \in \mathbb{R}$ % desired output
Output: $u_k \in [-1, 1]$ % control action scaling factor
Parameters: $\sigma \in \mathbb{R}$ % error margin
$\varepsilon \in (0, 1)$ % stabilizing margin
$\alpha \in (0, 1)$ % error filtering factor
$\bar{e}_b \in (0, 1)$ % bias adaptation threshold
$\delta_b \in (0, 1)$ % bias adaptation step
$\bar{e}_g \in (0, 1)$ % gain adaptation threshold
$\delta_g \in (0, 1)$% gain adaptation step
Variables: $\bar{e}_k \in [-1, 1]$ % low-passed filtered error sign
$b_k \in [-1, 1]$ % adaptive gain
$g_k \in [-1, 1]$ % adaptive bias

$k = 0, u_0 = \bar{e}_0 = b_0 = g_0 = 0$
`isStable` $\leftarrow$ `false`
while $k++$ **do**
  % update error
  if not `isStable` **or** $|y_k - y^d| > \sigma$ **then**
    $\bar{e}_k = \texttt{sgn}(y_k - y^d)(1 - \alpha) + \alpha \bar{e}_{k-1}$
    if $\bar{e}_k < \varepsilon$ **then**
      `isStable` $\leftarrow$ `true`
      $\bar{e}_k = 0$
    else
      `isStable` $\leftarrow$ `false`
    end if
  end if
  % update controller variables
  $b_k = \texttt{sat}_{[-1,1]}(b_{k-1} + \delta_b \texttt{hside}(|e_k| - \bar{e}_b)\texttt{sgn}(\bar{e}_k - \bar{e}_b))$
  $g_k = \texttt{sat}_{[-1,1]}((g_{k-1} + \delta_g)\texttt{hside}(|e_k| - \bar{e}_g)\texttt{sgn}(|e_k| - \bar{e}_g)$
  $u_k = \texttt{sat}_{[-1,1]}(b_k + g_k)$
end while

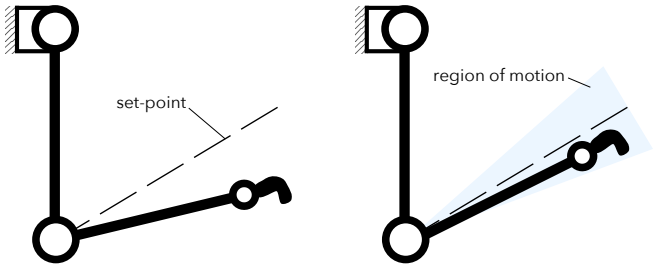

Figure 6: (Left) The controller moves the *Forearm* towards the desired configuration q^d. (Right) Once the joint has stabilized, the error margin σ is taken into account. It is marked by the area shaded in blue. This region represents an area where the controller error will remain at 0, once the stabilizing margin has been reached.

Properly tuning the controller's parameters to obtain the desired behaviour requires knowledge of the system's general dynamics. In this analysis, we refer to the system's equation of motion to understand how its behaviour is influenced by the

adaptive bias and gain of the controller. The general equation of motion for dynamic systems is:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) = \tau \tag{3}$$

Where $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$ are the joint configuration, velocity, and acceleration vectors; $M(q)$ is the inertia matrix; $C(q,\dot{q})$ is the vector of Coriolis and centripetal torques; $F(\dot{q})$ represents the friction torques in the joints; $G(q)$ is the vector of gravitational torques; and τ is the vector of driving torques.

Keeping the arm at the desired configuration q^d implies that if $q - q^d = \bar{e}_k = 0$, then we expect $\ddot{q} = \dot{q} = 0$ because we do not need to accelerate the system any further. Also, if $\bar{e}_k = 0$, then $g_k = 0$ and the b_k remains constant. A function $h(\ddot{q}, \dot{q}, q)$ that groups the dynamic forces of the system is introduced to rewrite Equation 3 as:

$$h(q,\dot{q},\ddot{q}) + G(q) = \tau \tag{4}$$

We can directly relate this pair of terms to the ones in Equation 1. The Adaptive Bias term compensates for the gravitational forces in $G(q)$ when the system is no longer in motion; the Adaptive Gain term compensates for the dynamic forces $h(\ddot{q}, \dot{q}, q)$ of the system when a motion is required to move the joint towards q^d.

If instead we consider a desired velocity $\dot{q}^d$, the relation with the bias and gain terms changes. In this case, if $\bar{e}_k = 0$, then we expect $\ddot{q} = 0$ and $\dot{q} = \dot{q}^d$. Rewriting Equation 3, this time by introducing a function $f(q,\dot{q})$ that groups all terms not dependent on $\ddot{q}$, we obtain:

$$M(q)\ddot{q} + f(q,\dot{q}) = \tau \tag{5}$$

The Adaptive Gain contribution is then related to $M(q)\ddot{q}$, while the Adaptive Bias must compensate for the remaining terms in $f(q,\dot{q})$. This function is non-linear because, for this system, (at least) $G(q)$ is non-linear. Therefore, it is not possible to fully adapt to these effects using only the bias term. Consequently, the Adaptive Gain must be continuously updated for correcting the resulting motion.

V. EXPERIMENTS

A series of tests have been performed to evaluate the control strategy and behaviour of the lazy controller. The experiments have been performed in a 7-DOF anthropomorphic robotic arm and consist on controlling the arm to open a drawer. The setup is described as follows. The arm has been positioned in front of a cabinet. The exact distance towards it is not relevant for the task, but must of course be within the arm's reach. It has been assumed that the region of motion, the area where the end-effector must make contact with the drawer, is obtained based on a given approximation of the handle's heigh. The end-effector is controlled by using the *Shoulder* joint as the "driver" joint, pushing the arm forward (or backwards), while the *Elbow* joint adjusts the end-effector's height to the desired set-point as a result of the motion

Controlling the *Elbow* joint for positioning the end-effector at a specific height can be achieved by monitoring its displacement. This joint is only required to rotate the *Forearm*. Therefore, the low presence of inertia and friction effects facilitate its control. Figure 7 (top left) shows the height of the end-effector as it successfully tracks a set-point while it is disturbed by external forces. The controller allows disturbances so long as the error remains within σ. When the error margin is exceeded, the controller is activated to steer the end-effector back to the set-point h^d. Three different disturbances are applied to the *Forearm*.

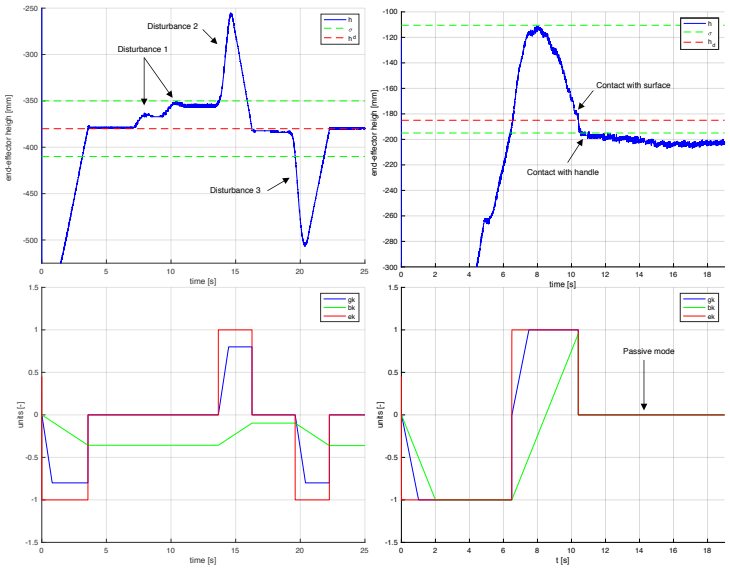

Figure 7: In the first experiment, the *Elbow* joint is being controlled to set the end-effector at a desired height h^d (top left). The controller (bottom left) is only active when the end-effector is outside the bounds of the error margin. The next experiment is opening a drawer (right). The end-effector's height is affected by both the motion of the *Elbow* and *Shoulder* (top right). When the end-effector makes contact with the drawer, the controller switches to passive mode (bottom right).

The first disturbance is a force that displaces the end-effector upward and is then quickly removed. As seen in Figure 7, the end-effector's height remains within the error margin and is held constant even after the disturbance has been removed. Therefore, there is no need to update the control action. As it turns out, the open-loop solution found by the controller still holds in this new configuration.

The next disturbance is applied to the *Upper Arm* generating a forward rotation that displaces the end-effector in an upwards direction again. This force is applied and held until the end-effector is back in the desired region of motion. Notice that the controller is not necessarily required to set the end-effector at the set-point. Here, it settles just below this value.

The final disturbance is a force that brings the *Upper Arm* back to its initial configuration. The resulting motion brings the end-effector down. The controller is again activated to lift its height into the region of motion. Notice that the jittering effects of the position after it has settled is neglected since $\bar{e}_k$ has stabilized and the error generated by this effect does not exceed σ.

The role of b_k in this experiment is to compensate

for gravity, as described in Section IV. For this motion, gravity effects are constrained to a torque in only one direction and its magnitude increases as the angle increases ($0° \leq q \leq 90°$). The Adaptive Bias term compensates the effects of gravity by generating a torque in the opposite direction. Furthermore, it has bee identified empirically that the maximum torque produced by gravity can be compensated with a value of $b_k = -0.4$. For this reason, the Adaptive Bias is constrained by $b_k \in [-0.4, 0]$. This way, we guarantee that the contribution of this term is always opposing the gravitational torque (or is 0) and we limit the amount of torque available to compensate its effects.

Next, we analyse the behaviour of the controller while opening a drawer. In this case, the end-effector must remain at a certain height while the *Shoulder* joint pushes it forward. A non-linear gravity compensation term has been added to the control action of both joints improve the system's performance. The driving torque from Equation 2 becomes:

$$\tau = \tau_{max} u_k + \hat{g}(q) \tag{6}$$

Where, $\hat{g}(q_n)$ is a non-linear function that estimates gravity effects based on an approximation of the arm's mass and current joint configurations. The resulting behaviour is shown in Figure 7 (right). Notice that at $t \approx 10$, the *Elbow* joint is set to passive mode. This action is triggered by making contact with the drawer. The pulling behaviour starts simultaneously. Also note that as the drawer is being pulled, the end-effector's trajectory is not completely horizontal as one would expect. Predicting this trajectory is not trivial. However, its vertical displacement is of no concern to the controller. The end-effector's position in this part of the task is guided by the drawer and its constraints, transferred to the arm through the coupling between the end-effector and handle. The controller's parameters for this experiments are shown in Table 1.

Table 1: Controller parameters for each joint. The *Shoulder* joint parameters vary depending on its behaviour.

Parameters	*Elbow* joint	*Shoulder* joint	
		Forward motion	Backwards motion
τ_{max}	10.325 Nm	12.320 Nm	20 Nm
σ	15 mm	0.268 rad/s	0.100 rad/s
ε	0.01	0.3	0.3
α	0.55	0.2	0.2
b_k	$\in [-1, 1]$	$\in [-1, 0]$	$\in [0, 1]$
$\bar{e}_b$	0.5	0.3	0.3
δ_b	0.0005	0.0005	0.0005
g_k	$\in [-1, 1]$	$\in [-1, 1]$	$\in [-1, 1]$
$\bar{e}_g$	0.99	0.8	0.8
δ_g	0.001	0.0003	0.001

Inertial and friction effects have a higher impact on the dynamic behaviour of the *Shoulder* joint because its motion is associated to the complete weight of the arm. Therefore, controlling it by monitoring its position quickly saturates the driving torque to its maximum value allowed. As a result, the system accelerates beyond the expected values. To avoid this, we choose to monitor the joint's velocity instead.

Figure 8 and 9 show the *Shoulder* joint behaviour while performing the task of opening a drawer. Such behaviour is separated into two guarded motions: one for reaching, between 3 s and 10 s, and one for pulling, followed immediately after. The controller parameters are adjusted accordingly for each, as shown in Table 1.

Figure 8 shows the *Shoulder* joint velocity as it influences the rest of the arm. The forward motion is activated when either the *Elbow* joint has reached its rotation limit or when the end-effector is close to reaching the desired height set-point. The backwards motion is then triggered when contact is made with the drawer. The motion stops when the *Shoulder* joint reaches a specific angle.

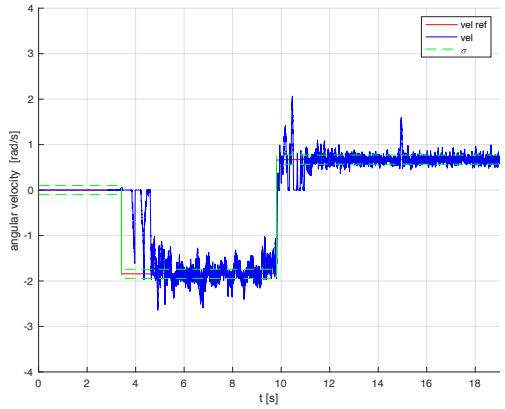

Figure 8: Velocity of the *Shoulder* joint while opening a drawer.

For this joint, a gravity compensation term has also been added to the driving torque, as in Equation 6. The controller is able to track the velocity set-point ω^d with an acceptable performance. However, it has problems attempting to stabilize $\bar{e}_k$ and remain within the desired error margin σ.

Figure 9 (top) shows the controller variables corresponding to the velocity plot in Figure 8. The peaks in g_k at $t \approx 4$ s and $t \approx 10$ s correspond to the high torque value necessary to overcome the striction torque. Once in motion, the contribution of b_k predominates over g_k. The adaptive bias attempts to find a constant value that will keep the velocity at the desired value ω^d. Concurrently, g_k contributes to the control action with quick adjustments.

Furthermore, Figure 9 (bottom) shows the low-passed filtered error sign $\bar{e}_k$. The red marks represent the time when $\bar{e}_k$ stabilizes, i.e., when the controller enters an open-loop behaviour. It can be seen that this state is continuously interrupted by the controller's inability to keep $\bar{e}_k$ stable. Regardless, it is still able to successfully control the *Shoulder* joint for executing the task in a safe and predictable motion.

A series of experiments have been performed to show the range of values in which the arm successfully achieves to open a drawer. A cabinet with drawers at different heights was used. The arm was placed at various distances and orientations towards the drawer. The range of values of the parameters that resulted in a successful execution is summarized in Table 2.

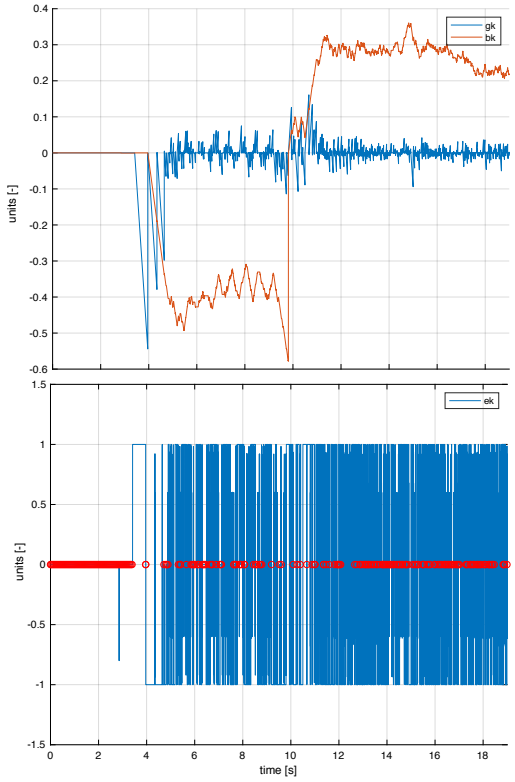

Figure 9: Adaptive bias and gain (top) while controlling the *Shoulder* joint for opening a drawer. The red circles represent the moment where the low-passed filtered error sign is within the stabilizing threshold (bottom).

Table 2: Range of values for test parameters that resulted in a successful execution.

Parameters	Range
Height of drawer	$h^d = \{\text{-}315, \text{-}275, \text{-}180\}$ mm
Distance to drawer	450 mm $\leq$ d $\leq$ 600 mm
Orientation to drawer	-25°$\leq \theta \leq$ 25°

The limitations in the distance to the drawer are due to the minimal space required to lift the end-effector above the handle and the maximum length of the arm. The orientation angle is constrained by the *Wrist* joint's mobility. This has consequences in the ability of attaching to the handle and pulling the drawer. Furthermore, it is important to remark that while not required to be precise, the end-effector's orientation at the moment of contact certainly has influence on properly hooking itself to the handle.These factors differ depending on the type of end-effector selected to perform the task.

VI. CONCLUSIONS

In this paper we have introduced a general lazy control approach that focuses on providing robustness towards changes and uncertainties in dynamic environments that are not considered in the classic control approach. Furthermore, we explored the behaviour of a lazy adaptive controller for a 7-DOF anthropomorphic robotic arm to perform the task of opening a drawer.

The controller does not require knowledge of the system parameters or a precise description of the workspace. It adapts the driving torque by monitoring and reacting to the system's behaviour. It is *lazy* in the sense that it finds a sufficient solution to the controller's performance requirements given by the task context. This is achieved by considering an error margin for set-point tracking precision and using the common workspace to guide the arm's motion.

The controller is robust towards uncertainties and changes in workspace parameters: uncertain measurements of the handle's position and changes in the distance and orientation to the drawer; and towards unknown workspace and system parameters: friction in the drawer rails and unknown dynamic behaviour of the arm, such as the coupling effect between links. Furthermore, motion is defined as a series of guarded motions. Therefore, it is able to adjust and pause its motion when disturbed by applying an external force since there is no dependency on a specific end-effector trajectory.

This approach results beneficial in situations where conditions in the environment are dynamic or make it difficult for obtaining accurate sensor measurements. Future work for improving the open-loop performance of the lazy controller when dealing with non-linear behaviour can still be done. This can be achieved through: 1) adding feed-forward control to reject predictable non-linear dynamic behaviour, i.e., gravity, friction, and coupling effects between links; and 2) applying a learning-based approach for improving adaptive parameter estimation.

References

[1] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling And Control*. John Wiley and Sons, December 2005.

[2] R. Kelly, V. Santibáñez, and A. Loría, *Control of Robot Manipulators in Joint Space*. Springer London, June 2005.

[3] G. Raiola, C. A. Cardenas, T. S. Tadele, T. de Vries, and S. Stramigioli, "Development of a safety and energy aware impedance controller for collaborative robots," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1237–1244, April 2018.

[4] T. S. Tadele, T. de Vries, and S. Stramigioli, "Combining energy and power based safety metrics in controller design for domestic robots," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, September 2014.

[5] B. Nemec and L. Zlajpah, "Force control of redundant robots in unstructured environment," *IEEE Transactions on Industrial Electronics*, vol. 49, pp. 233–240, February 2002.

[6] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control*. Wiley, 2005.

[7] E. Lutscher and G. Cheng, "Constrained manipulation in unstructured environment utilizing hierarchical task specification for indirect force controlled robots," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, June 2014.

[8] A. Jain and C. C. Kemp, "Pulling open novel doors and drawers with equilibrium point control," *2009 9th IEEE-RAS International Conference on Humanoid Robots*, January 2010.

[9] A. Jain and C. C. Kemp, "Pulling open doors and drawers: Coordinating an omni-directional base and a compliant arm with equilibrium point control," *2010 IEEE International Conference on Robotics and Automation*, July 2010.

[10] E. Scioni, *Online Coordination and Composition of Robotic Skills: Formal Models for Context-aware Task Scheduling.* IUSS Ferrara 1391, University of Ferrara, Italy and Department of Mechanical Engineering, KU Leuven, Belgium, April 2016.

[11] A. Mallet and A. Franchi, "Adaptive closed-loop speed control of bldc motors with applications to multi-rotor aerial vehicles," *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.