A Self-tuning Impedance Controller for Autonomous Robotic Manipulation

Pietro Balatti\textsuperscript{1,3}, Dimitrios Kanoulas\textsuperscript{2}, Giuseppe F. Rigano\textsuperscript{2}, Luca Muratore\textsuperscript{2,4}, Nikos G. Tsagarakis\textsuperscript{2}, and Arash Ajoudani\textsuperscript{1}

\textbf{Abstract}—Complex interactions with unstructured environments require the application of appropriate restoring forces in response to the imposed displacements. Impedance control techniques provide effective solutions to achieve this, however, their quasi-static performance is highly dependent on the choice of parameters, i.e. stiffness and damping. In most cases, such parameters are previously selected by robot programmers to achieve a desired response, which limits the adaptability of robots to varying task conditions. To improve the generality of interaction planning through task-dependent regulation of the parameters, this paper introduces a novel self-regulating impedance controller. The regulation of the parameters is achieved based on the robot’s local sensory data, and on an interaction expectancy value. This value combines the interaction values from the robot state machine and visual feedback, to authorize the autonomous tuning of the impedance parameters in selective Cartesian axes. The effectiveness of the proposed method is validated experimentally in a debris removal task.

I. INTRODUCTION

During the past decades, both natural catastrophes caused by the Earth’s natural processes (e.g., earthquakes, tsunami and their consequences) and man-made disasters (e.g. Chernobyl nuclear accident) have highlighted the need for effective and efficient robotic systems that can be quickly deployable after the disaster, to assist in tasks too hazardous for humans to perform.

To prevent and respond to such disasters, some common manipulation tasks have been studied during the past years. The DARPA Robotics Challenge (DRC) \cite{1} sought to address this problem by investigating tasks such as removing debris blocking an entryway, using a cutting tool, and turning a valve near a leaking pipe. The presence of high uncertainty levels in such environments and tasks urges the development of autonomous robotic behaviors that are triggered by the rich local sensory systems. Such systems must be capable of distinguishing expected interactions from disturbances, to be able to react accordingly and appropriately.

A set of autonomous skills can be developed based on the observations from human demonstrations \cite{2}–\cite{5}, however, their performance is highly dependent on the richness of the training data sets. In addition, current wearable sensory systems still do not provide effective solutions for the measurement of contact forces while performing complex ma-
removal task using a KUKA LBR IIWA 14 robot to demonstrate the potential of the proposed methodology. Starting from a compliant end-effector, the robot will adjust the Cartesian stiffness and damping in directions which are necessary to perform the task, e.g. in the direction of gravity due to the deviations from the desired trajectory caused by the payload. Other unexpected disturbances will not lead to the tuning of the impedance parameters, and will be responded by a soft contact relying on the robot Cartesian and nullspace compliant behavior.

II. METHOD

The proposed framework can be subdivided into four main units: (1) a self-tuning Cartesian impedance controller that enables the tuning of the stiffness and damping parameters in certain Cartesian axes, (2) an object localization method that provides the exact pose from where the object needs to be grasped, (3) an interaction expectancy module that provides the possibility of an expected physical interaction between the robot and the object/environment, and (4) a sensor fusion of force and visual localization from step (2) to effectively execute the manipulation task. The aforementioned parts are regulated by means of a Finite State Machine (FSM) in charge of assembling all the components of the system.

A. Self-tuning Cartesian impedance controller

The ability to autonomously tune impedance parameters of the robot interaction controller can help to address varying uncertainty levels of a manipulation task, and contribute to improved performances. Preliminary work in this direction investigated the adaptation of impedance parameters based on human observations [12] and energy tank-based optimization [13], to improve the physical interaction performance based on the varying task conditions.

However, one of the biggest challenges in this context is to make such techniques work properly in unstructured environments, by distinguishing expected interactions from external disturbances. This consideration will enable adaptation of the impedance parameters only to the interactions that are expected, while achieving a compliant behavior in response to the external disturbances to avoid the generation of unnecessary high interaction forces (e.g. collisions with the environment).

To this end, in this paper, a novel self-tuning impedance controller is developed that functions based on an interaction expectancy value. This value is calculated from the FSM and visual feedback interaction values (see sections below), to enable autonomous tuning of impedance parameters when an external interaction is expected. Furthermore, the tuning of the parameters are achieved in selective axes of the Cartesian space based on end-point sensory data, to avoid unnecessary stiffening/complying of the remaining axes. The block scheme of the proposed self-tuning impedance controller is provided in Fig. 1.

The robot is torque controlled and the vector of robot joint torques $\tau \in \mathbb{R}^n$ is generated as follows:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + \tau_{ext}$$  \hspace{1cm} (1)

where $n$ is the number of joints, $q \in \mathbb{R}^n$ is the joint angles vector, $J \in \mathbb{R}^{6 \times n}$ is the robot arm Jacobian matrix, $M \in \mathbb{R}^{n \times n}$ is the mass matrix, $C \in \mathbb{R}^{n \times n}$ is the Coriolis and centrifugal matrix, $g \in \mathbb{R}^n$ is the gravity vector and $\tau_{ext}$ is the external torque vector. $F_c$ represents the forces vector in the Cartesian space and $\tau_{st}$ the second task torques projected onto the null-space of $J$.

Forces $F_c \in \mathbb{R}^6$ are calculated as follows:

$$F_c = K_c(X_d - X_a) + D_c(\dot{X}_d - \dot{X}_a)$$  \hspace{1cm} (3)

where $K_c \in \mathbb{R}^{6 \times 6}$ and $D_c \in \mathbb{R}^{6 \times 6}$ represent respectively the Cartesian stiffness and damping matrix, $X_d$ and $X_a \in \mathbb{R}^n$ the Cartesian desired and actual position, $\dot{X}_d$ and $\dot{X}_d \in \mathbb{R}^n$ their corresponding velocity profiles. The Cartesian desired position and velocity are generated with a fifth-order polynomial trajectory.

The first rule of the proposed robot self-tuning impedance controller is to achieve compliant Cartesian and nullspace behaviors. To this end, the default $K_c$ and $K_n$ matrices are chosen to achieve a good tracking performance in the presence of joint friction, while providing a compliant response in case of accidental collisions. $K_n$ is a constant nullspace stiffness matrix used in the secondary task of the self-tuning Cartesian impedance controller. A stiffness-consistent nullspace projection [16] was used to calculate $\tau_{st}$.

When an expected interaction with the environment is detected by the interaction expectancy module (see Sec. II-C), the Cartesian stiffness matrix $K_c$ and consequently

\[\tau_{ext} = J(q)^T F_c + \tau_{st}\]  \hspace{1cm} (2)

$\tau_{ext}$ is the vector of external torques. $F_c$ is the vector of Cartesian forces. $\tau_{st}$ is the vector of second task torques.

Fig. 1: Block scheme of the proposed self-tuning impedance controller.
the damping matrix $D_c$ are subject to changes as follows, depending on $\Delta F_{ext,t}$, i.e. the variation of the external forces detected by the robot at time $t$ w.r.t. the ones measured in the previous control loop:

$$\Delta F_{ext,t} = F_{ext,t} - F_{ext,t-1}. \quad (4)$$

First case: if $\Delta F_{ext,t} < 0$, the stiffness needs to be increased on a certain Cartesian axis to ensure the movement is completed in that particular direction in a precise manner. For instance, when a heavy object is grasped, the loading effect will introduce deviations from the desired trajectory. In this scenario the stiffness is regulated following equations:

$$\Delta X_t = |X_d - X_a| \quad \text{(5)}$$

$$K_{c,t} = K_{c,t-1} + \alpha \Delta X_t \Delta T \quad \text{(6)}$$

subject to $-f_{max} < F_{ext,t} < f_{max}$

subject to $K_{c,min} < K_{c,t} < K_{c,max}$

where $\alpha$ is a coefficient to be set, $\Delta T$ the control loop sample time, $f_{max}$ the maximum allowed interaction force level, $K_{c,min}$ the minimum Cartesian stiffness value and $K_{c,max}$ the maximum Cartesian stiffness value.

Second case: if $\Delta F_{ext,t} > 0$, the stiffness needs to be decreased on a certain Cartesian axis, switching back to compliant mode, e.g., when an object is laid down on the ground before being released, the robot needs to adapt to the ground forces to avoid the generation of high interaction forces between the object and the environment. The stiffness parameters are decreased according to the following law:

$$K_{c,t} = K_{c,t-1} - \beta \Delta F_{ext,t} \Delta T \quad \text{(7)}$$

subject to $\Delta X_t < \delta x_{max}$

subject to $K_{c,min} < K_{c,t} < K_{c,max}$

where $\beta$ is a coefficient to be set, $\Delta F_{ext,t}$ is defined as in (4) and $\delta x_{max}$ represents the maximum allowed displacement error from the desired trajectory, i.e. $\Delta X_t$.

The damping matrix $D$ is tuned online based on the resulting $K_{c,t}$:

$$D = \Lambda_s D_{diag} K_{adj} + K_{adj} D_{diag} \Lambda_s, \quad \text{(8)}$$

where $D_{diag}$ is the diagonal matrix containing the damping factors ($\zeta = 0.7$), $K_{adj}=K_{adj} \Lambda_s$ and $\Lambda_s \Lambda_s = \Lambda$, where $\Lambda$ is the desired end-effector mass matrix [17].

B. Object Localization and Grasping Pose Extraction

The second module involves the visual-based method of localizing the object to be manipulated in the environment and extracting its grasping pose frame, noted as $\hat{x}_g, \hat{y}_g, \hat{z}_g)$. In this paper, we will focus on objects that are stick-like and lie in front of a planar segment, such as a door or wall. As an input to this method, we will consider point cloud data coming from an RGB-D sensor at $30Hz$. Given the input point cloud, we first segment the dominant plane using the RANdom SAmple Consensus (RANSAC) method [18].

In this way, we extract the normal of the plane $\hat{n}_w$ (in cyan color in Fig. 2) and we segment all the points above the plane to different Euclidean clusters [19], [20]. Each point cloud cluster represents a potentially different object in the scene. For each one of the clusters, we fit a 3D line using again RANSAC. Those that respect some length constraints, i.e., longer but narrower than the hand’s palm size, are considered to be stick-like objects, potentially good for grasping. In this way, we extract the fitted 3D line’s axis as the grasping frame’s $\hat{x}_g$-axis (in red color in Fig. 2). We construct the remaining axes of the grasping frame in the following way. The $\hat{y}_g$-axis is the cross product between the plane normal vector $\hat{n}_w$ and the $\hat{x}_g$-axis, i.e., $\hat{y}_g = \hat{n}_w \times \hat{x}_g$. Finally, the $\hat{z}_g$-axis is the cross product of the other two, i.e., $\hat{z}_g = \hat{x}_g \times \hat{y}_g$. In this way, we generate the grasping frame to be matched to the hand frame during grasping, in a way that the closing fingers can encapsulate the object and the hand is perpendicular to the flat surface (wall/door), so that we avoid as much as possible early collisions with the object during the approaching phase. The position of the grasping frame has a fixed height with respect to the robot base frame and is at the edge of the object. We pick the right edges for right-
handed grasps and left edges, otherwise. To do that, for each
object point cloud cluster, we extract the point cloud nearest
neighborhood at the fixed height from the robot’s base and we
select as grasping position the right/left-most point.

Another important information that can be extracted from
vision is a cylinder around the object main axis, that repres-
ts the interaction field between the hand and the object.
This information serves as input to the interaction expectan-
cy module (Sec. II-C). The cylinder is extracted along the fitted
3D line axis for each object, with radius \( r \) that equals the
maximum size of the hand fingers (appears as red cylinder in
Fig. 2) and its length is coming from the extreme 3D points
of the object point cloud clusters that were extracted after
the plane segmentation phase.

### C. Interaction Expectancy Module

The third unit includes a method responsible of deciding
if a physical interaction between the robot and the ob-
ject/environment is to happen or not. There are some areas
in which this interaction is expected and so the self-tuning
Cartesian impedance module has to be activated and some
other areas in which no interaction is predicted and the robot
should keep the same or default level of compliance. Two
Boolean values are used to decide if an external interaction
is expected and if the interaction expectancy value has to
be triggered. The first one, the FSM interaction value, is
given by the FSM and is activated only in those states in
which an interaction is expected, e.g., while picking an object
and placing it down on the ground. The second one, the
interaction field value, is given by the vision module and it
is True if the end-effector is located inside the interaction
field, or False, otherwise. These two values are correlated
by the following Boolean logic rule:

\[ I_e = I_m \land I_f \]  \hspace{1cm} (9)

where \( I_e \) is the interaction expectancy value, \( I_m \) is the FSM
interaction value, \( I_f \) is the interaction fields value and \( \land \)
is the logical AND operator.

### D. Sensor Fusion of Force and Visual Localization

In disaster scenarios, since the environment is uncertain,
the manipulation workspace may not be always bright and
clear, for instance because of illuminations, shadows, or even
fire smoke. Therefore, a unit to assist the vision module to
enhance the robustness of the object pose estimation has
been developed. After reaching the pose estimated by the per-
ception module, the external forces are sensed and translated
from the joint to the Cartesian space as in (10) where \( J^+T \)
represents the transpose of Jacobian pseudoinverse matrix
and \( \tau_{q,ext} \) the joint external torques vector. If \( \Delta F_{ext} \neq 0 \)
on axis \( i \), a contact with the object has been established on
that axis and there is no need of a further movement in
that direction. On the contrary, if \( \Delta F_{ext} = 0 \) on axis \( i \), no
contact is detected and a movement forward on that axis is
performed until \( \Delta F_{ext} \neq 0 \), ensuring that the end-effector
is ready to grasp the object.

\[ F_{c,ext} = J^+T \cdot \tau_{q,ext} \]  \hspace{1cm} (10)

Performing end-effector movements against the object to
sense the mentioned forces could lead to an accidental fall of
the object, if it is placed in a precarious position, but since
these adjustments are made with a very high compliance,
the end-effector will not displace the object from its original
pose. Note that the fusion of data coming from visual
localization and external forces sensing is done sequentially.

### E. Finite State Machine

The described units are linked together by means of a
Finite State Machine, depicted in Fig. 3, that allows
the framework to be completely autonomous. Each state
represents a motion primitive while the outgoing arrows
represent the possible feedback that can be given in response
to the performed subtask outcome. Solid arrows stand for
successful actions while dashed arrows are directed to safe
states in case a fail in the task occurs.

The FSM initial state is identified by the “Homing” state,
that is also the safe exit state for the primitives involving arm
movements. In this recovery state, the robot end-effector is
in the center of the workspace. The natural control flow leads
firstly to a “Reach” state, where the end-effector approaches
the object depending on the position and the orientation
provided by the object localization module. During this
phase, the robot enters in the interaction field and the self-
tuning impedance controller is enabled.

To double-check if the pose given by the vision object lo-
calization unit is accurate, the FSM switches to the “Adjust”
state where the sensor fusion unit can move the end-effector
forward on the axes where no forces are detected. Once the
end-effector palm makes contact with the object surface, the
“Grasp” primitive closes the robot hand and allows the object
grasping. Consequently, the object has to be taken away in a
safe space and released. To accomplish this goal, four other
primitives were designed. In the first place, the object is
brought in front of the robot (“Pick” state) and then moved
away at its side (“Move Away” state). Before being released
in the “Ungrasp” state, we want to make sure that the object
falls down in the space where it does not hinder future robot
operations. Therefore, in the “Place Down” state, the object

Fig. 3: The system is controlled by a Finite State Machine. Each
state represents a motion primitive.
upper-end is first rotated in a way that it points away from
the robot and the intended future operational space, and then
placed down until it makes contact with the ground. In this
state, the self-tuning impedance controller is enabled again,
so that when the object makes contact with the ground, the
robot becomes compliant since $\Delta F_{\text{ext},t}$ will be positive.

III. ROBOT INTERFACES

The software architecture of the robot relies upon XBot-
Core [21] (Cross-Bot-Core) – a recently developed open-
source\(^2\) and light-weight Real-Time (RT) platform for
robotics, designed to be both an RT robot control framework
and a software middleware. XBotCore satisfies RT require-
ments, ensuring 1 kHz hard RT control loop even in complex
multi-DOF systems, moreover it provides a simple and easy-
to-use middleware Application Programming Interface (API),
for both RT and non-RT external control frameworks. This
API is completely flexible with respect to the framework a
user wants to utilize. It is also possible to reuse the code
written using XBotCore API with different robots (cross-
robot feature) thanks to the Robot Hardware Abstraction
Layer (R-HAL) introduced in [22].

The R-HAL permits to seamlessly program and control
any robotic platform providing a relatively uniform abstrac-
tion that hides the specifics of the underlying hardware.
Consequently, this allows the robot software developers to
easily transfer/reuse their code on different robots. All the R-
HAL implementations are built as a shared library loaded at
runtime according to what specified in a configuration file. In
particular, the R-HAL implementation for the KUKA IIWA
has been provided relying on the FRI (Fast Robot Interface)
to transfer data between the robot controller and the external
computer. It is important to mention that the control was
done in non-RT mode because the FRI API communication
provided by KUKA is intrinsically not RT safe. However,
a porting of the KUKA API to the RT domain could be
possible with no particular issues ensuring always no changes
in the XBotCore plugin developed. The control behavior is
implemented using two different XBot plugin. One of them
is used to realize the Cartesian impedance controller. The
second plugin implements the FSM behavior exchanging
data (pose to reach, external force detected) with the other
plugin.

IV. EXPERIMENTS

We conducted experiments using a KUKA LBR IIWA 14
robotic arm equipped with an underactuated hand, i.e., the
Pisa/IIT SoftHand [23] in a debris removal setup. An ASUS
Xtion Pro RGB-D sensor was attached in the front part of
the robot arm base and calibrated with respect to the end-
effector. In the default mode, the diagonal components
of the stiffness matrix ($K_c$) were set to 100 N/m on the trans-
lational components, and to 20 Nm/rad on the rotational
components. The choice of update rate parameters $\alpha$ and
$\beta$ in (6) and (7) was done experimentally with the aim of
minimizing the Cartesian error within the trajectory period of
the current state. Different debris lengths and weights were
used in our experiments.

Fig. 4 illustrates the different phases of the task execution
with reference to the FSM states. The leftmost figure at the
top (4.1), shows the robot in the “Homing” state, i.e., its
initial configuration. The robot end-effector then reaches the

\(^2\)https://github.com/ADVRHumanoids/XBotCore
Fig. 5: Experiment 1 - External forces in the Cartesian space (a), the Cartesian displacement between desired and actual position (b), the curve trend of the Cartesian stiffness (c), FSM interaction value (d), interaction field value (e) and interaction expectancy value (f). The dashed vertical lines highlight when the different FSM states are performed referring to Fig. 4.

debris grasp pose received by the object localization module (Fig. 4.2, “Reach” state), entering the interaction field that surrounds the debris area and therefore activating the interaction field value as shown in Fig. 5.e. The interaction expectancy value (Fig. 5.f) is not yet activated since the other component of the logical AND, i.e., the FSM interaction value (Fig. 5.d), is still False and will be active in future states (“Pick” and “Place Down”). The sensor fusion of force and visual localization unit then checks if the grasp is ready to be executed (Fig. 4.3, “Adjust” state). Once the debris is in contact with the end-effector, the grasping action is performed (Fig. 4.4, “Grasp” state) and the object is picked (Fig. 4.5, “Pick” state). Since the latter state enables a positive FSM interaction value, the Cartesian stiffness (Fig. 5.c) is now subject to changes: $\Delta F_{\text{ext},t} < 0$ on z-axis, so $K_c(z)$ is increased following (6) until the end of the current state when the FSM interaction value is deactivated.

At the end of the “Pick” state the FSM interaction value becomes False and so does the interaction expectancy value. To show that the Cartesian stiffness would not change even if unexpected external interactions occur, the framework leaves the robotic manipulator in the current configuration for 10 seconds allowing a person to interact with it. In Fig. 4.6 a subject perturbs the robot in all the Cartesian axes as the external forces plot (Fig. 5.a) shows from time 20.5s to time 30.5s and the Cartesian stiffness is not affected by such changes, regardless of the force and position displacements.

After these manual perturbations, the FSM restarts its natural control flow to move away the debris (Fig. 4.7, “Move Away” state) and place it down (Fig. 4.8, “Place Down” state). In this state, the FSM interaction value becomes again True and, since the object is still in the interaction field, the interaction expectancy value is activated. The Cartesian stiffness (Fig. 5.c) is again subject to changes, this time $\Delta F_{\text{ext},t} > 0$ on z-axis, so $K_c(z)$ is decreased following (7) until a lower bound set to 100 N/m (as the initial value) is reached. This behavior results in the object being gently placed on the ground. At the end of the “Place Down” state the FSM interaction value is set back to False and the “Ungrasp” state opens the end-effector and releases the object (4.9). As soon as the debris exits the interaction field, the relative interaction field value is set to False. Fig. 4.10 shows the final setup with the object laying on the ground and the robot back to the “Homing” state ready to start the
task over.

A similar experiment was conducted with another object of different weight ($0.7\,Kg$) and length, to demonstrate the system flexibility to varying task conditions. The results of the first phase, where the interaction field was active, are shown in Fig. 6 (“Pick” state). $K_c(z)$ reaches a maximum value of $\approx 1330\,N/m$ in comparison with the $\approx 1900\,N/m$ of the first experiment where the weight of the object was $2\,Kg$.

Due to the similarity of the interaction expectancy profiles, the plots are not repeated here.

V. DISCUSSION AND CONCLUSION

In this work, a novel self-tuning impedance controller was proposed to improve robot autonomous adaptability to varying task conditions. The selective adaptation of the Cartesian impedance parameters was carried out based on local force and vision sensory data. To improve the situation-awareness of the robot in responding to expected interactions and complying with the external disturbances, the interaction expectancy module combined two interaction values from vision and FSM to make robot controller respond appropriately and accordingly. The choice of update rate parameters $\alpha$ and $\beta$ were carried out experimentally in this work, that must be improved to enhance robot autonomy. In this direction, the parameters will be tuned based on the task and on the object material, retrieved by the perception module. The effectiveness of the proposed control framework in adapting to varying task conditions, in the presence of external disturbances, was experimentally carried out in a debris removal task.

REFERENCES


