A Control Architecture for Compliant Execution of
Manipulation Tasks

Mario Prats and Angel P. del Pobil and Pedro J. Sanz
Robotic Intelligence Lab
Universitat Jaume I
Castellón, Spain
Email: [mprats,pobil,sanzp]@icc.uji.es

Abstract—This paper deals with the problem of dependable physical interaction through manipulation in partially-known everyday human environments. We present a modular software architecture that allows the definition and compliant execution of manipulation tasks under the Task Frame Formalism. We show the details of several software modules implemented within this architecture, that enable higher levels of adaptability and robustness, as well as the incremental incorporation of more complex skills in a modular fashion.

The whole system is validated making a real robot arm with a three-finger hand perform two different manipulation tasks: opening a doorhandle and taking a book out of a bookshelf. Results show how the presented framework is suitable for easily defining and performing a great variety of manipulation tasks.

I. INTRODUCTION

With the paradigm shift of robotics moving from a specific industrial technology to the consumer, home and service markets, fundamental challenges arise. New robot systems are called for that are more intelligent, flexible, modular, dependable and robust. Particularly, physical interaction in everyday human environments requires flexible robot systems endowed with rich sensory-motor skills and multi-sensory feedback based on modular software architectures for perception and control that enable them to exhibit higher levels of adaptability, through learning and multimodal acquisition of skills, for achieving goals in incompletely known environments.

The research presented in this paper addresses some of these issues in the context of physical interaction through manipulation in partially-known everyday human environments. Manipulation is a well-known problem in robotics [1] [2]. In the community there has been a focus on grasping [3] [4] [5], however, physical interaction through manipulation in our daily life even for simple and common tasks goes well beyond grasping for picking and placing. Examples of those tasks are: switching on the light, taking a book out of a bookshelf, opening a door or a drawer, pushing objects, turning on a tap, etc. These are basic skills that need to be incorporated into future service robots in a robust and flexible way.

Some of those tasks have already been addressed in the literature exhibiting a reasonable performance for the particular tasks for which the systems were designed. However, they typically lack adaptability and fail miserably when small changes are introduced in the task. Among the features that usually account for this fact we can mention, among others, the impossibility of controlling the velocity of a force-controlled direction, the use of a fixed value of the controller gain for a given task, or lacking the capability of online modification or addition of new behaviors.

In this paper we present a software architecture that allows the definition and compliant execution of complex robot manipulation tasks within the Task Frame Formalism (TFF) [6]. It includes the three different ways of specifying the task frame and allows to switch between them at runtime. Along with the architecture, we show several perceptual and action modules for the robot to sense the environment and act in consequence. For safety and performance reasons, the robot is endowed with a compliant impedance velocity/force controller that takes care of external forces, making it robust to environmental changes or modelling errors. Its gain is computed online making the robot able of executing several tasks that are not possible to perform with other types of velocity/force controllers. We also introduce the notion of ability, as a module in charge of recursively defining and supervising tasks. Perceptual modules are independent of the actions resulting in a few general motor actions that can be used in a modular way in many manipulation tasks, after performing the suitable perceptual actions. Moreover, due to the recursive definition of abilities, it is possible to construct more complex abilities incrementally. Our approach allows the parallel execution of robot behaviors and the online modification or addition of new behaviors.
making it amenable to incremental learning methodologies.

Our architecture has been implemented on the UJI Service Robot [7]. The whole system provides a general framework for defining and executing daily manipulation tasks in a modular and flexible way. In particular, this paper describes two tasks, concerning the extraction of a book from a bookshelf (see Figure 1) and turning a door handle, though the architecture provides an interface for easily defining new tasks.

In Section II we briefly introduce the robot used in our experiments. Section III presents the software architecture along with the perception, action and abilities modules that have been implemented. Two tasks performed within this architecture by a real robot are described in Section IV, and performance results are presented. Finally, some conclusions and future directions are outlined in Section V.

A. Related work

Service robotics is an area that is getting more and more attention as time goes by. However, at the best of our knowledge, there are few service robots able to perform more than one different manipulation task in a home environment. Robots are usually programmed to perform a given, specialized task, and have difficulties when doing other things. One example is the cleaning robot described in [8].

One major problem in designing multitask robots is that we need to have several controllers and to select which of them to use, depending on the task. This problem has been addressed in some relevant works such as [9], or more recently in [10] [11]. These works pursue similar goals to ours. The first makes use of their own architecture for opening a door with a mobile manipulator, which has been shown to require complex manipulation and coordination capabilities. The other two, also define an architecture for task execution, although they are focused on industrial assembly rather than on service robots. However, they also obtain interesting results in autonomously inserting a bulb into a bayonet socket.

One main drawback of the existing approaches is that they need to specify the task at a very low level for the robot to perform it successfully. In the work of Thomas et al. [10], the robot performs the task using a hybrid velocity/force control [6]. With this controller, the space of possible effector motions is divided into velocity-controlled directions and force-controlled directions. This means that we cannot control the velocity of a force-controlled direction, which can be necessary under certain service tasks, as we will explain in section III.

On the other hand, Petersson & Kragic [9] make use of an impedance controller which is more suitable for service tasks. However, they set the gain of the controller to a fixed value for a given task. When performing another task, they would have to find another gain experimentally, which might present a problem on robots that need to learn new tasks autonomously.

We make further improvements in this direction by using an impedance velocity/force controller whose gain is computed automatically online, given the current robot velocity, and the maximum force that the robot can apply without damaging the environment or breaking its fingers. These constitute more natural magnitudes for defining a task. Moreover, our architecture allows the parallel execution of robot behaviors and, more important, permits the online modification or addition of new behaviors. This last feature is necessary, for example, in a learning-by-demonstration framework, when we want the robot to learn more behaviors without having to shut it down and program them by hand.

In addition, all manipulation tasks are defined within the TFF, which has been shown to constitute a powerful concept for the compliant execution of manipulation tasks [11] [12]. Kröger et al. show in [11] three different ways of specifying the task frame, for the robot to be able to execute any sensor-guided command. Our architecture implements these three cases and, in addition, allows to switch between them at runtime.

II. System description

The UJI Service Robot (Figure 2) is a mobile manipulator consisting of a Mitsubishi PA-10 arm mounted on an ActivMedia PowerBot mobile robot. The manipulator is endowed with a three-fingered Barrett Hand (Figure 1) and a JR3 force/acceleration sensor that is mounted at the wrist, between the hand and the end-effector. A small IEEE1394 camera is also attached to the wrist of the robot. The manipulator, hand and camera controllers, along with the control computer are also mounted on the mobile platform and connected to the PowerBot batteries. The computer is a Pentium 4 running at 3 GHz, with 512 Mb of RAM.

The architecture described here has been programmed in C++ making extensive use of class abstraction in order to provide an intuitive way for adding new parts. It has been
designed for being used with both the arm and the mobile platform, although we are now concerned with manipulation, and only the arm sensors have been implemented as perception modules. The architecture is very general and the integration of the mobile platform in the future should be straightforward.

III. SOFTWARE ARCHITECTURE

Several software architectures for mobile robots have been proposed in the literature [13]. However, there are few contributions in the area of manipulators [11] [14]. As Sciavicco and Siciliano pointed out in [15], a control architecture for manipulators should be endowed with:

- The manipulation ability, in charge of acting on the environment.
- The sensory ability, capable of obtaining information from the world.
- The data processing ability, which processes data of system activity.
- The intelligent behavior ability, capable of modifying system behavior according to external information.

Many architectures, also in mobile robots, implement some of these features, but the latter one is usually missing. This is particularly true for most of the behavior-based architectures [16], where a set of robot behaviors are preprogrammed for a particular task. Robots running under these architectures work really well when doing the tasks they have been designed for, but are unable of creating new behaviors and doing other things.

From our point of view, this is a fundamental issue for future robot development. A robot must be able of executing the tasks it has been designed for, but also to modify or even to create new behaviors according to its experience, within a learning framework. Taking these goals into account, our architecture is structured into three main modules: perceptions, actions and abilities.

A. Perceptions

As perception, we understand any information that could be used to guide our actions. We consider both exteroceptive and proprioceptive perception. An example of exteroceptive (or external) perception could be the perception that the robot has about the force at the wrist. This information must clearly be used to guide the robot’s action avoiding undesirable results, as well as any other input from sensors that deal directly with the environment. The proprioceptive (or internal) perception is as important as the external one, and makes the robot be aware of information that does not come directly from the external world, but is still necessary for performing tasks. One example could be the robot’s perception of its hand position. If the robot senses that its hand is too far from its body, it might consider moving its body to a more comfortable position.

We also consider as internal perceptions the references, conditions and immediate goals of the robot. For example, the robot must be aware of the maximum force that its fingers can support. This is a reference that must be encoded into the robot’s perception. A condition could be the perception that the current force at the fingers is higher than the maximum allowed. The activation of this perception should immediately trigger an appropriate robot response. With respect to immediate goals, we refer to spatio-temporal references that the robot might immediately use, as for example, the position in space where it has to immediately move its hand.

In particular, the most important perceptions that the architecture implements are the following:

1) **Proprioceptive perceptions**: The robot is aware of the current configuration of its fingers, the position of its hand with respect to its base, the distance of its hand w.r.t another given frame, the time, the maximum force that it can resist, the task frame, the velocity at which it must move, the particular parameters of its hand (as the maximum grip size), the hand configuration that it should adopt and the point in the space where it should immediately move.

2) **Exteroceptive perceptions**: The robot is able to perceive images from the world and contours from these images. It can also sense the force present at the wrist, and the force at each of the fingers.

The robot is also aware of conditions on most of these perceptions, as for example, whether the force at the fingers is greater than a given value. Many of the above mentioned perceptions depend upon others. For example, the module that perceives contours, needs as input one image, that is actually perceived by another module. Another example is the perception that senses the force at the wrist. This one makes use of the perception that stores the current effector position, in order to compute and to balance the gravity effects that the hand’s mass introduces into the force sensor. The architecture provides very simple mechanisms for defining this kind of relationships.

It is worth noting that, although we have implemented the particular perceptions that the robot needs for doing manipulation tasks, the architecture provides common interfaces that allow the easy integration of new sensors, as it could be a new camera, or the range laser of the mobile robot. Hardware abstraction, extensibility and scalability are important features for any architecture [17].

B. Actions

A robot that needs to interact with the environment has to know how to control its actuators for achieving a desired goal. The action module of the architecture provides facilities for the design and integration of control laws. Using these facilities, we have endowed our robot with two different control modules. One that is in charge of controlling finger motions, and the other, more sophisticated, which controls arm motions in a compliant way.

The compliant motion of robots for interaction control is a topic that has been extensively studied in the literature since the very beginning [6] [18] [19] [20] [21]. Although several variations have been presented, there are basically two methods for simultaneously controlling velocity and force of a manipulator: hybrid velocity/force control [18] and impedance control [19].
Hybrid velocity or position/force control is usually used when the robot has to perform some kind of task, because it fits very well with the TFF [6]. Within this framework, the whole space of possible task directions is divided into two orthogonal subspaces: one that stores force-controlled directions (position-constrained), and the other that represents velocity-controlled directions (force-constrained). A great number of tasks can be performed within this framework. The only condition is that it must be possible to decompose the task into force and velocity-controlled directions [8] [12].

This is usually true for interaction tasks in industrial environments, as cleaning or polishing a surface. With this kind of tasks, the hybrid control allows us to explicitly control the force along the constrained directions. For a polishing task we can command, for example, to exert a force of 20N over the surface, while freely moving along the tangent plane. The controller would work flawlessly under these circumstances.

However, when considering a simple daily task as opening a drawer, we realize that it is difficult to perform it with a hybrid controller. The reason is that, now, we do not want to exert a given force, but to produce a given movement. We cannot explicitly control the force along the opening direction. It could happen that the drawer is too heavy, and the force that we control is not enough. Alternatively, the drawer could be too light for the force we are controlling, and the robot would open it too quickly, perhaps breaking it.

For this kind of task, we do not want to control the force explicitly. Instead, the velocity is more important. We do not say: “open the drawer at 20N, no matter the velocity!” It is more natural to say: “open the drawer slowly”. Of course, we still need to have into account the task forces, but in a different way. The robot must try to maintain a given velocity, adapting its force accordingly, in order to maintain it between the range that its fingers (or arm) can support. If the drawer is locked, the robot would try to open it exerting more and more force, until its force limits would be reached.

An impedance controller has explicit control of velocity in all directions [19]. There is an outer loop that modifies the velocity references depending on measured forces. For the drawer example, this controller is valid as long as the force loop cancels any velocity reference, when the measured force is the maximum that the robot can apply. With an impedance controller, the force at which the velocity is cancelled depends upon a gain $k$, according to the following equation:

$$v' = v + kf$$

(1)

where $v$ is the reference velocity, $v'$ is the modified velocity, and $f$ is the current external force. $k$ is the control gain, and it is normally set to a fixed value [9]. From equation 1, we can see that $v'$ will be zero, when $k = -\frac{v}{f}$. Thus, if we want to make the final velocity $v'$ zero when the current force $f$ reaches a maximum value, no matter the reference velocity $v$, then $k$ must be updated at each iteration as $k = -\frac{v}{f_{\text{max}}}$, being $f_{\text{max}}$ the maximum force that the robot can apply. If $k$ is not updated online, the robot will stop its motion at different forces depending on its velocity. Instead, we want to always stop the motion when $f = f_{\text{max}}$, independently of the robot’s velocity.

The schema of our implicit impedance controller is shown in Figure 3. It takes as input a desired velocity expressed in the task frame. This velocity can be the coupling of the outputs of several controllers, which allows to control each direction with different sensory inputs. For example, we might want to do visual servoing in one direction, while position or velocity control in the others. The directions managed by each controller can be selected with the corresponding selection matrices $S_1, S_2, \ldots, S_n$. The result is a velocity vector expressed in the task frame, which is used for computing the new gain $k$, and then modified according to current force perception. Another selection matrix, $S_f$, is in charge of selecting which directions must be also force-controlled. The resulting modified velocity $v'$ is transformed into the effector frame and sent to the robot low-level controller.

As it can be deduced from equation 1, the resulting velocity $v'$ depends linearly upon the current force $f$. It could be interesting to have a better robot response in extreme cases, where the current force exceeds the maximum force that the robot can support. It is for that reason that we have imposed an exponential increase to the final robot’s velocity, in cases where $|f| > |f_{\text{max}}|$. The behavior of the controller is depicted in Figure 4, where the resulting velocity is represented with respect to the current force, for three different reference velocities, and taking $f_{\text{max}} = 10N$. We can see how, independently
of the reference velocity, the control is always zero when \( f = f_{\text{max}} \).

We have integrated this controller, along with the controller that manages finger motions, into the action structure of the architecture. From this point of view, an action module takes as input a subset of perceptions (from those explained in section III-A) and generates a control vector, which can be a six-dimensional vector for the case of the robot’s effector motion, or a three-dimensional vector for finger control.

In particular, the action that manages the compliant motion of the arm, takes as input the following perceptions:

- The velocity at which the robot must move.
- The task frame, where velocities are defined.
- The maximum force that the robot can resist, for avoiding damages.
- The current force at the wrist.

Taking these inputs, it uses the controller of Figure 3 for computing a control vector that will be finally sent to the robot. This direct connection between perception and action, provides the robot with a reactive behavior. Information coming from perceptual modules does not need to visit higher cognition modules before being used for action. Perceptual changes, including modifications in the task frame or maximum allowed velocity, will immediately reflect into the robot’s action.

The architecture also allows for the execution of actions which do not control robot motion. Instead, these actions can act on the robot’s perception, as for example, modifying the velocity reference, the task frame, etc. In conclusion, an action, within this architecture, represents a process that affects the perceptual or motor state of the robot.

C. Abilities

Although perception may be used to guide our actions at the lowest level, it is also used for making a mental representation of what is happening and planning future actions. Thus, perceptual information must also flow to high level modules where plans are made and supervised.

In our architecture, the Abilities module is in charge of defining and supervising manipulation tasks. Within this framework, we define an ability as a set of actions or other abilities connected by conditions. An ability can be seen as an automaton, where nodes are actions (see section III-B), or other abilities, and arcs are conditions, which are built on perceptions (see section III-A). The main difference with respect to other existing approaches, as the skill primitive nets [10], is that our nets are not only composed of manipulation primitives [10] [11], but also contain nodes that act on robot’s internal perception. This clearly has the advantage that the task information is not encoded within the action code. Instead, it is stored in perceptual modules that are independent of the action. This feature allows us to have few and general motor actions, that can be used in any manipulation task, after performing the suitable perceptual actions.

Figure 5 depicts an example of ability that shows all the possible connections that the architecture allows. The nodes represent actions, whereas the arcs represent conditions. The architecture allows the parallel execution of actions. In the example of the figure, there are two initial nodes, A1 and A5. Thus, these two actions will execute concurrently. If condition C1 occurs when executing A1, the robot will start to execute A2 while still performing A5. If at this point condition C2 holds true, action A2 will finish but only action A5 will be active, because C2 will be waiting for C5 to trigger. If this happens, A3 will start its execution. If, instead, the condition that triggers when doing A2 is C4, then A4 will start its execution immediately, without waiting for C5 or C3. So, apart from the connections between actions, the architecture also provides facilities for synchronization.

Moreover, due to the recursive definition of ability, as the connection between actions or other abilities, we are able to integrate existing abilities into others. With this mechanism it is possible to construct more complex abilities incrementally. For example, a robot could know a simple ability as moving along a direction until a high force is detected. This ability could be integrated into a more complex one as pushing a button, which in turn could belong to another one, and so on. This offers a framework for robot learning that we would like to address in future work: the robot could incrementally learn new abilities on the basis of the skills that it already knows.

IV. EXAMPLE TASKS AND RESULTS

In this section we present two tasks that our robot can perform in the basis of the architecture explained in the last section. These are: turning a door handle and the extraction of a book from the bookshelf.

A. Turning a door handle

In [9], a method for opening a door is presented, which focuses on the last part of the task: that of following the appropriate arc when opening the door. Here we focus on the first part aimed at turning the door handle in a compliant manner.

We assume that the robot’s fingers have successfully been placed over the door handle, so that the task frame corresponds to the frame depicted in Figure 6. The robot’s strategy lie in moving the hand in -Y direction as long as the absolute value of the opposite force is lower than 15N. At the same time, it must slowly turn its wrist, as long as the torque in Z direction is lower than 1 Nm. If the rotation velocity is not suitable for the particular case, then a torque around direction Z will appear, and the robot will update its wrist rotation in

![Fig. 5. An ability is the connection of actions by means of conditions.](image-url)
Fig. 6. Frame attached to the door handle.

Fig. 7. Ability for turning a door handle.

consequence, trying to maintain the torque under 1 Nm. It is worth noting that the rotation of the wrist also affects the task frame of Figure 6, modifying the direction in which the robot applies the real force.

This strategy has been implemented as an ability in our architecture. The schema is shown in Figure 7. Each node represents an atomic action. These actions are already implemented in the architecture. The task is defined by selecting which actions to use, and setting the input values. The first node is in charge of defining the task frame: it is set to the Barrett’s Hand central finger frame. Then, the desired velocities and allowed maximum force are defined. Finally, the robot executes the Move Arm action, which corresponds to the compliant controller explained in section III-B. When the force in Y direction exceeds the value of 15N, we consider that the handle has reached its limit and the door can be opened. The value of 15N has been set high enough to ensure that the robot will be able to turn any handle.

The whole sequence can be observed in Figure 8. Figure 9 shows the forces that appear during the execution of the task, and how the robot reacts to them. During the first phase, the force in Y direction increases until an approximate value of 10N which corresponds to the resistance of the particular door handle. Consequently, the hand’s velocity along this direction decreases, because the current force is approaching the maximum allowed force. During this stage, and for this particular case, there is no torque around Z direction and the wrist is rotating at 0.04 rad/s. During the second phase, the handle starts offering more resistance, and the opposite force finally reaches the value of 15N, which was set to be the maximum force. At this point, as expected, the velocity decreases to zero, and the task is considered as finished. It is worth noting that, during this second stage, the torque around Z axis starts increasing (in negative direction). The robot, then, modifies its velocity around this axis, and stops the motion when the absolute value of the torque reaches the maximum allowed value of 1N.

The combination of both behaviors allows the robot to perform the task in a compliant manner, without having knowledge of any handle’s model. Figure 10 depicts the trajectory followed by the hand, represented in the robot’s base frame.
B. Taking a book out of the bookshelf

The architecture allows for the execution of more complex tasks, as for example, taking a book out of the bookshelf. In this case, finger motion must also be controlled. We have already addressed this problem [7], but using a simple parallel jaw gripper, which limited the manipulation capabilities of the system. Now we have installed a Barrett’s Hand, and consider taking the book by imitating human movements.

For taking out a book, we usually place one finger on the top and make it turn with respect to the base. Then, with the help of other fingers, we finally grasp it. This is the strategy that we have imitated with the robot. Figure 11 shows the whole sequence. The task frame at each step is drawn in red. The arcs represent transitions between different actions. Above the arc, the condition that triggers the transition is written. The next action to perform appears below the arc.

As Figure 11 depicts, the initial task frame is set to the robot’s fingertip. The robot starts moving in Z direction of the task frame, until it is making a force of about 8N. At this point, the robot starts moving in X direction of the task frame, while still making 8N of force along Z direction, and rotating around Y axis. This motion is managed by the compliant impedance controller of section III-B. When robot’s perceptual modules detect that the angle with respect to the original position is higher than 15 degrees, another action is triggered, which closes the fingers and grasps the book. At this point, the task frame is modified online, and set to the original position where the first contact was made. The book is finally taken out by moving the hand with positive velocity in X and negative velocity in Z direction of the new task frame.

Figure 12 represents the forces that appear during the task, and the consequences on the robot’s behavior. The first stage corresponds to the initial movement, when the fingertip is moving along Z direction, searching for contact. Thus, no forces are present, and the robot moves with a velocity of 10 mm/s along Z axis. When contact is made, the opposite force suddenly increases, until it reaches the maximum value of 8N. At this point, the velocity along Z direction is approximately zero, and the robot starts moving along X direction, and around Y axis. This makes the robot perform the rotation movement of the book. In order to avoid the sliding of the book during this stage, the force of 8N must be kept constant during the second stage, and so is done as the figure shows. Step 3 corresponds to the point where the fingers grasp the book. This action introduces some instabilities in force readings, so all velocities are set to zero during this phase. Finally, during the fourth phase, velocities are set to constant values, and the book is actually extracted from the bookshelf. The positive force that appears during this step is due to the book’s weight.

Figure 13 shows the trajectory followed by the fingertip during the execution of this task, represented within the XY plane of the robot’s base coordinate system.

V. CONCLUSION AND FUTURE LINES

In the execution of manipulation tasks, external forces play an important role, and must be taken into account for avoiding damages and failure. This is more important when the robot has to perform service tasks in unstructured everyday
human environments. In this paper, we have presented a control architecture that allows the compliant execution of this kind of tasks. The architecture implements an impedance velocity/force controller with a gain that is computed online, taking into account the maximum force that the robot can perform. The architecture offers a great variety of perception modules which can be combined with robot actions by means of abilities.

Within this framework, tasks can be easily defined and performed by the robot. We have tested the approach with the UJI service robot for two complex manipulation tasks, namely: turning a door handle, and taking a book out of the bookshelf. Results show how the robot smoothly adapts its behavior according to the forces that appear during the execution of these tasks, making it robust to environmental changes or modelling errors.

Due to the fact that the architecture allows for the online modification and addition of robot abilities, we plan to use it for training the robot in the context of a learning approach. The robot could learn more and more abilities, each based on the previous one. We also plan to use visual perception combined with force and tactile sensing for a multimodal acquisition of skills. Our aim is to have a service robot, that is not only able of grasping and moving objects, but also of autonomously performing in a dependable way common daily tasks such as opening doors and drawers, switching on the light, etc.

ACKNOWLEDGMENT

This paper describes research carried out at the Robotic Intelligence Laboratory of Universitat Jaume-I. The authors would like to thank the Spanish Ministry (MEC), under project DPI2004-01920, and Generalitat Valenciana, under projects CTBPRB/2005/052 and GV05/137 for their invaluable support in this research.

REFERENCES