Robotica (2007) volume 25, pp. 245–256. © 2007 Cambridge University Press
doi:10.1017/S0263574706003304 Printed in the United Kingdom

Towards multipurpose autonomous manipulation with the UJI service robot

M. Prats, P. J. Sanz, A. P. del Pobil, E. Martínez and R. Marín

Department of Computer Science and Engineering, Universitat Jaume I. Castellón, Spain

(Received in Final Form: December 18, 2006)

SUMMARY

This paper presents a modular control architecture that enables a mobile manipulator to be compliant and perform manipulation tasks in partially known everyday human environments. An impedance velocity/force controller that allows the execution of a great variety of tasks under the Task Frame Formalism (TFF) is implemented.

Tasks are represented as a net of basic abilities which are performed by the robot using the impedance controller. Mechanisms for switching between abilities according to the robot's perceptual state are defined.

We show the validity of our approach on the UJI Service Robot, making it to perform a common daily task such as opening a door. Finally, we apply this framework to make progress on the new version of the UJI Librarian Robot, making a great step forward in the way the robot manipulates books.

KEYWORDS: Service robotics; compliant manipulation; Human-Robot interface; Control architecture.

1. 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 multisensory 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;[3–5] 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 behaviours.

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 suggested in ref. [7] 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 capable of executing several tasks, which is 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 behaviours and the online modification or addition of new behaviours, making it amenable to incremental learning methodologies.

Our architecture has been implemented on the UJI Service Robot[8] (see Fig. 1). 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: turning a door handle and taking a book out of the bookshelf using a three-fingered hand, which is the basis of the new version of our UJI Librarian Robot.[8]

In Section 2 we briefly introduce the UJI Service Robot. Section 3 presents the compliant velocity/force impedance controller that guides our robot. Section 4 presents the software architecture, composed of the perception, action and abilities modules. The door opening task performed

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press



Fig. 1. The UJI Service Robot.

within this architecture by the real robot is described in Section 5, and performance results are presented. Then, in Section 6, the librarian robot application is outlined, and we report the progress that has been achieved with respect to older prototypes, thanks to the use of the new architecture. In particular, we focus on the task of grasping a book by imitating human movements. Finally, some conclusions and future directions are outlined in Section 7.

1.1. Related work

Service robotics is an area that is getting more and more attention with time. However, to the best of our knowledge, there are few service robots that are 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 ref. [9].

One major problem in designing *multitask* robots is that we need to have several controllers and then select which of them to use, depending on the task. This problem has been addressed in some relevant works such as in ref. [10], or more recently in refs. [7] and [11]. These works pursue similar goals as ours. The first makes use of their own architecture for opening a door with a mobile manipulator, a task that has been shown to require complex manipulation and coordination capabilities. The other two also define an architecture for task execution based on skill primitives,[12] although they are focused on industrial assembly rather than on service robots. However, they also obtain interesting results by 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.,[11] 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, according to the Task Frame. 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 3.

On the other hand, Petersson and Kragic[10] 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 for 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 before 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 behaviours and, more importantly, permits the online modification or addition of new behaviours. This last feature is necessary, for example, in a learning-by-demonstration framework, when we want the robot to learn more behaviours without having to shut it down and program them by hand.

In addition, all manipulation tasks are defined within the TFF,[6] which has been shown to constitute a powerful concept for the compliant execution of manipulation tasks.[7,13] Kröger et al. show in ref. [7] 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 among them at runtime.

With respect to the librarian robot application, it is noticeable that only two known different laboratories have investigated in this area, both with the aim to provide book browsing to the user. Thus, a teleoperated solution was presented by Tomizawa et al.,[14] from the University of Tsukuba, Japan. Two important tasks that need to be

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

improved in the authors' opinion[14] are optimization of the manipulation planning and improvement of the book recognition, which are the major topics of our research. The other system is controlled in an autonomous manner, by a laboratory from John Hopkins University. This work is known as Comprehensive Access to Printed Materials (CAPM).[15] Its main objective, presently under progress, is to enable real-time browsing of printed materials through a web interface. So, with this aim, an autonomous mobile robotic library system has been developed to retrieve items from bookshelves and carry them to scanning stations located in an off-site shelving facility.

The main contribution of our work with respect to these systems is that the environment is not modified for making things easier. Instead of placing the books into special boxes, which are easier to grasp, we try to grasp the books as they can be found in common libraries, without making strong assumptions. Instead of adapting the world to the robot, we rather try to adapt the robot to the world.

2. System Description

The UJI Service Robot (Fig. 1) 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 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 this paper deals with manipulation, the sensors and actuators of the mobile robot are also integrated into the architecture.

3. Compliant Controller

The compliant motion of robots for interaction control is a topic that has been extensively studied in the literature since the very beginning.[6,16–19] Although several variations have been presented, there are basically two methods for simultaneously controlling velocity and force of a manipulator: hybrid velocity/force control [16] and impedance control.[17]

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.[9,13]

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 20 N 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 such 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 20 N, 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.[17] 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 $\mathbf{k}$, according to the following equation:

$$\mathbf{v}' = \mathbf{v} + \mathbf{k}\mathbf{f} \tag{1}$$

where $\mathbf{v}$ is the reference velocity, $\mathbf{v}'$ is the modified velocity, and $\mathbf{f}$ is the current external force. $\mathbf{k}$ is the control gain, and it is normally set to a fixed value.[10] From Eq. 1, we can see that $\mathbf{v}'$ will be zero, when $\mathbf{k} = -\mathbf{v}/\mathbf{f}$. Thus, if we want to make the final velocity $\mathbf{v}'$ zero when the current force $\mathbf{f}$ reaches a maximum value, no matter what the reference velocity $\mathbf{v}$, then $\mathbf{k}$ must be updated at each iteration as $\mathbf{k} = -\mathbf{v}/\mathbf{f}_{\max}$, being $\mathbf{f}_{\max}$ the maximum force that the robot can apply. If $\mathbf{k}$ 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 $\mathbf{f} = \mathbf{f}_{\max}$, independently of the robot's velocity.

The schema of our implicit impedance controller is shown in Fig. 2. 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 the system 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 others. The

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

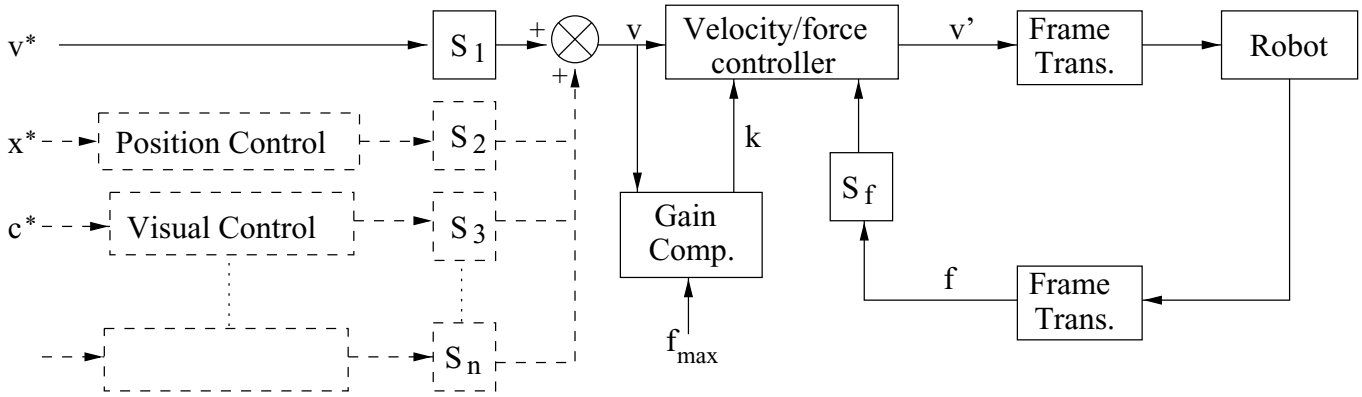


Fig. 2. The velocity/force controller schema.

directions managed by each controller can be selected with the corresponding selection matrices $\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_n}$. The result is a velocity vector expressed in the task frame, which is used for computing the new gain $\mathbf{k}$, and then modified according to current force perception. Another selection matrix, $\mathbf{S_f}$, is in charge of selecting which directions must also be force controlled. The resulting modified velocity $\mathbf{v}'$ is transformed into the effector frame and sent to the robot low-level controller.

As it can be deduced from Eq. 1, the resulting velocity $\mathbf{v}'$ depends linearly upon the current force $\mathbf{f}$. 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 $|\mathbf{f}| > |\mathbf{f}_{\max}|$. The behavior of the controller is depicted in Fig. 3, where the resulting velocity (for one dimension) is represented with respect to the current force, for three different reference velocities, and taking $\mathbf{f}_{\max} = 10\,\mathrm{N}$. We can see how, independently of the reference velocity, the control is always zero when $\mathbf{f} = \mathbf{f}_{\max}$.

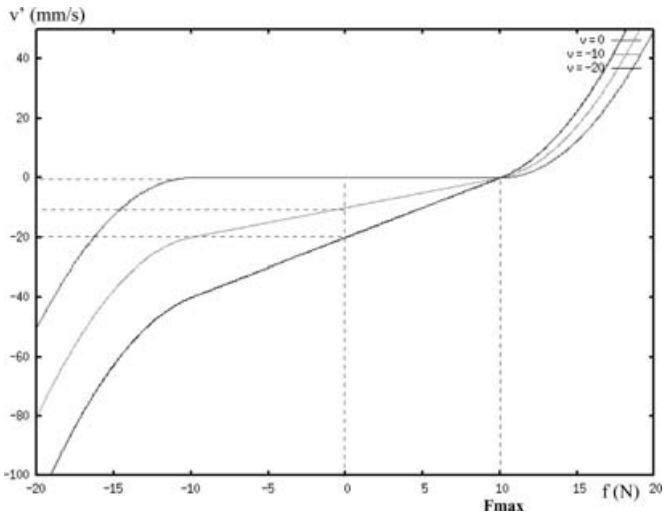


Fig. 3. The resulting velocity w.r.t current external force (only for one dimension).

4. Software Architecture

Several software architectures for mobile robots have been proposed in the literature.[20] However, there are few contributions in the area of manipulators.[7,21] As Sciavicco and Siciliano pointed out in ref. [22], 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 behaviour according to external information.

Many architectures implement some of these features, but the latter one is usually missing. This is particularly true for most of the behaviour-based architectures,[23] where a set of robot behaviours are pre-programmed for a particular task. Robots running under these architectures work really well when doing the tasks they have been designed for, but are unable to create new behaviours and doing other things.

From our point of view, this is a fundamental issue for future robot development. A robot must be able to execute the tasks it has been designed for, but also to modify or even to create new behaviours according to its experience, within a learning framework. Taking these goals into account, our architecture is structured into three main modules: perceptions, abilities and actions (see Fig. 4).

As perception, we understand any information that could be used to guide our actions. We consider both exteroceptive and proprioceptive perception as, for example, the perception of the force at the wrist or the robot's perception of its hand position. We also consider, as internal perceptions, the references, conditions and immediate goals of the robot. For example, the maximum force that the robot's fingers can support, the condition that compares the current force at the fingers with respect to the maximum allowed, or the position in space where the robot has to immediately move its hand.

The robot is also aware of conditions from most of these perceptions, as for example, whether the force at the fingers is greater than a given value. Many of the perceptions depend

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

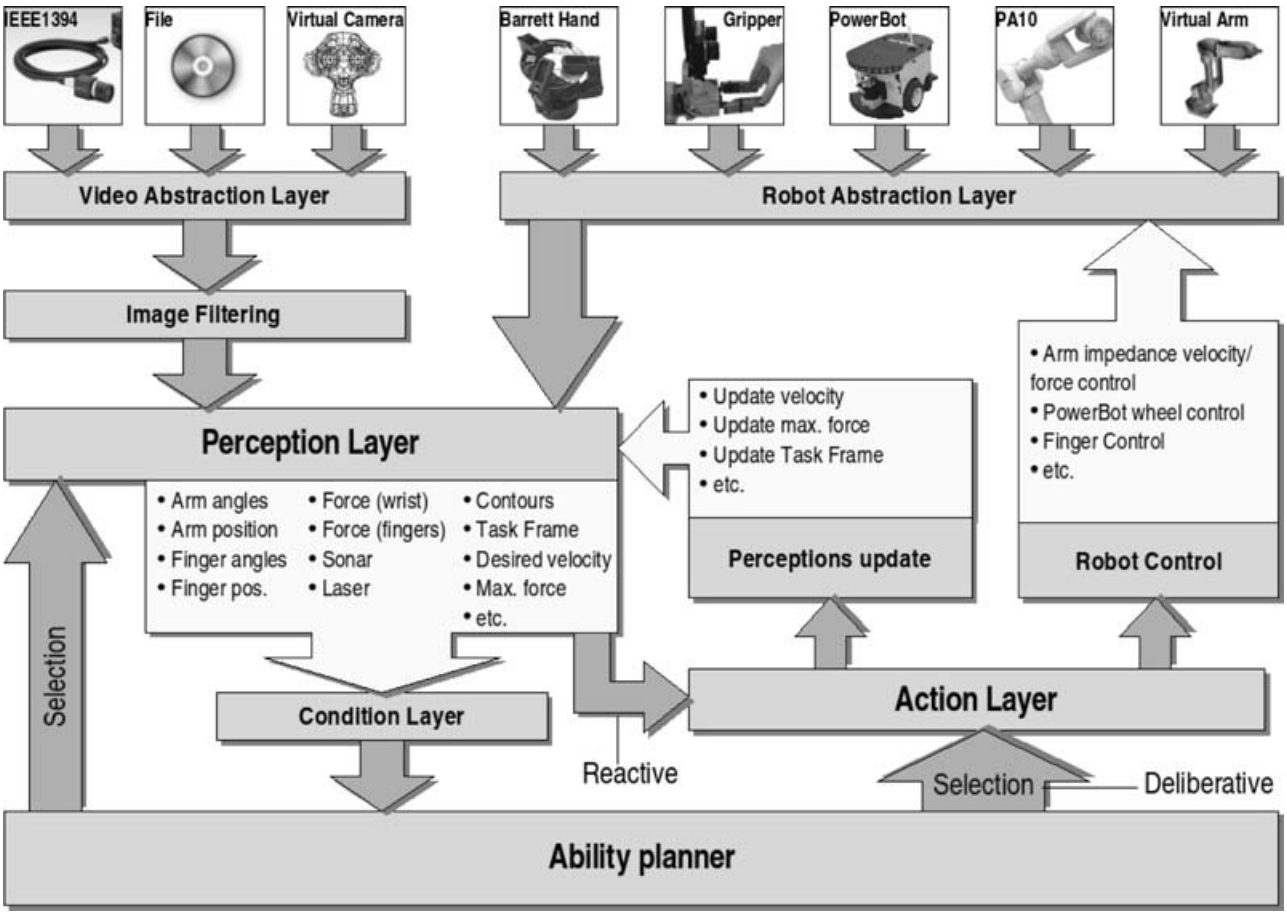


Fig. 4. Architecture for mobile manipulation (AMM).

upon others. For example, the perception that senses the force at the wrist 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 these kinds of relationships, and for easily integrating new sensors. Hardware abstraction, extensibility and scalability are important features for any architecture.[24]

Data coming from perceptions are used to make actions in a reactive manner.[23] The controller presented in Section 3, has been integrated into the action module of the architecture, along with other control algorithms that manage the fingers and the mobile robot's wheels.

An action takes as input a subset of perceptions and generates a control vector, which is sent to the robot. 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 damage
- The current force at the wrist.

Taking these inputs, the action uses the controller of Section 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 behaviour. 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 be immediately reflected in 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.

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.

The *Abilities* module is in charge of continuously selecting which perceptions to activate and which actions to perform according to these perceptions. It 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 or other abilities, and arcs are conditions, which are built on perceptions. The main difference with respect to other existing approaches, such as the skill primitive nets,[11,12] is that our nets are not only composed of manipulation primitives,[7,11] but also contain nodes that act on robot's internal perception. This clearly has

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

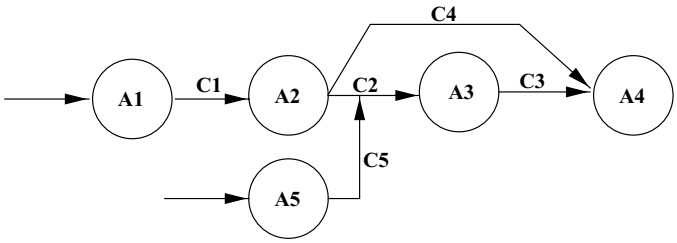


Fig. 5. An ability is the connection of actions by means of conditions.

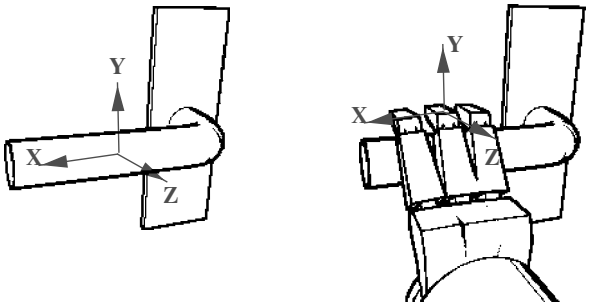


Fig. 6. Frame attached to the door handle (left) and Task Frame (right).

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 an 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 shown in 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 such as moving along a direction until a high force is detected. This ability could be integrated into a more complex one such 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.

5. Example of a Task

In this section we present a task that the robot can perform on the basis of the architecture explained in the last section. The aim of the task is to open a door. In ref. [10], a method for opening a door is presented, which focuses on the last part of the task: 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 Fig. 6. The robot's strategy lies in moving the hand in $-$Y direction as long as the absolute value of the opposite force is lower than 15 N, which was chosen experimentally and should be enough for turning any handle. 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 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 Fig. 6, modifying the direction in which the robot applies the force. The remaining degrees of freedom are also force-controlled in order to deal with modelling errors.

This strategy has been implemented as an ability in our architecture. The schema is shown in Fig. 7. Each node represents an atomic action. These actions are already implemented in the architecture as basic abilities. 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 3. When the force in Y direction exceeds the value of 15 N, we consider that the handle has reached its limit and the door can be opened. The value of 15 N has been set high enough to ensure that the robot will be able to turn any handle.

The whole sequence can be observed in Fig. 8, where the task frame is illustrated in red. 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 10 N, 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 15 N, which was set to be the maximum force. At this point, as expected, the velocity decreases to zero, and the task is considered 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 1 N.

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

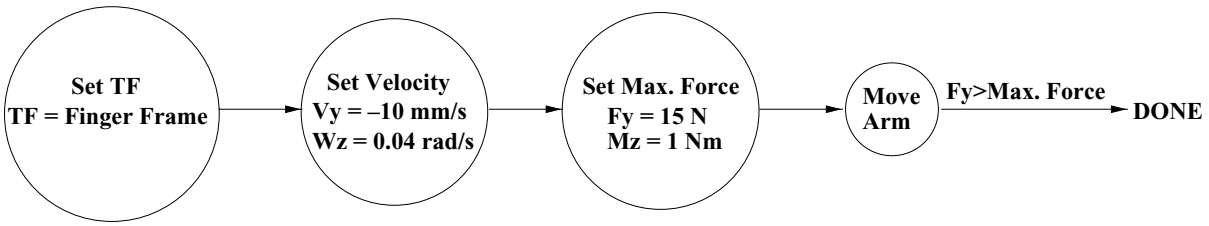


Fig. 7. Ability for turning a door handle.

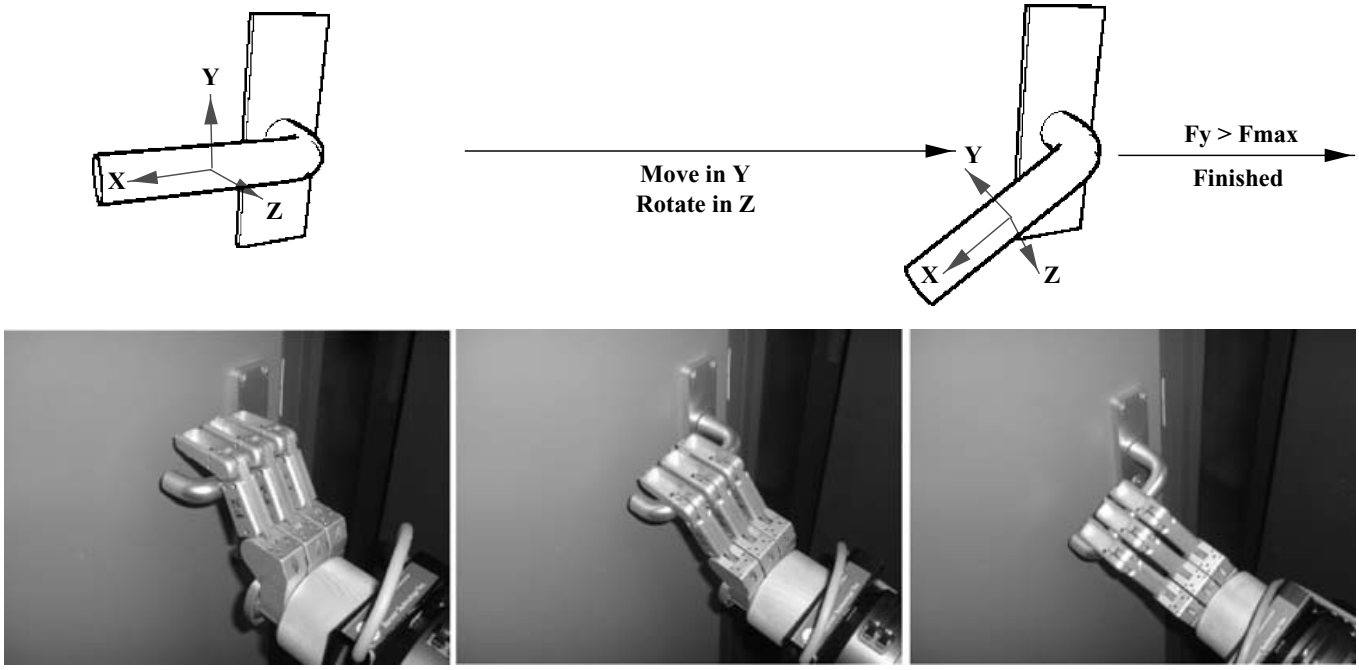


Fig. 8. Sequence of the robot turning the door handle.

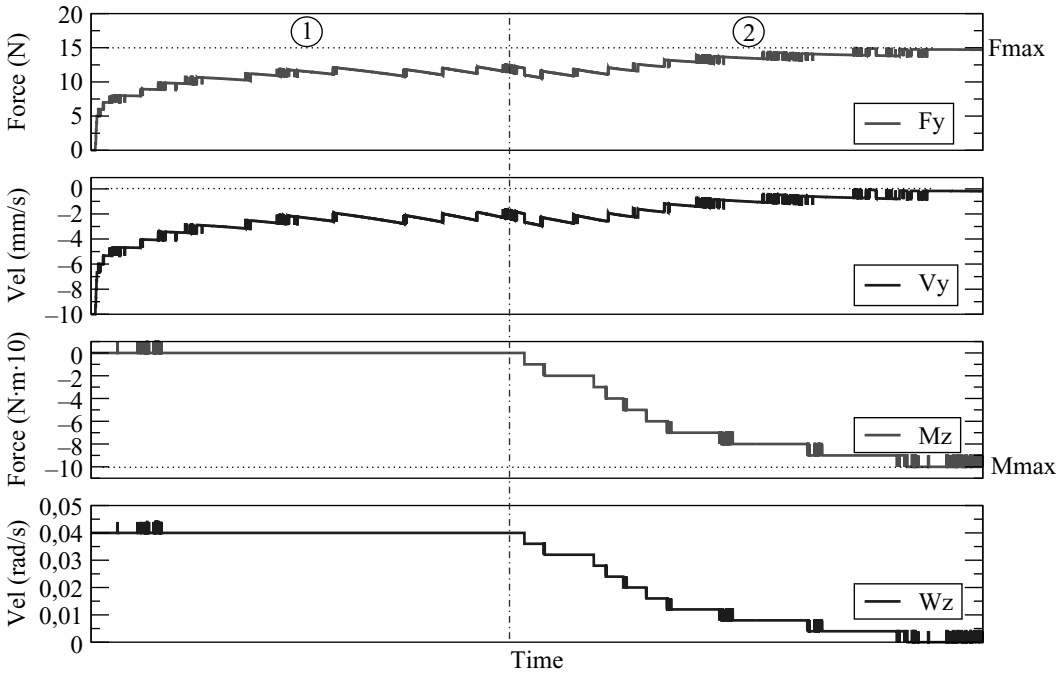


Fig. 9. Forces and velocities during the door handle task.

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press



Fig. 10. Old and the new hand.

The combination of both behaviours allows the robot to perform the task in a compliant manner, without having knowledge about any particular model of handle.

6. Librarian Robot Aplication

The controller, explained in Section 3, and the architecture of Section 4, have been integrated into a real robotic application: the Librarian Robot.

In our research laboratory, we are working on a librarian robot, designed to work in this partially known environment. We want the system to be able to search and retrieve a book requested by a user. The operation starts when the user requests a book by its name or code. The robot is then in charge of locating the book in an ordinary library, extract it and take it to the user. The only initial information is the book code, which appears written on a label on each book.

After several years using the Jaume robot for developing this application,[8,25] now we use the UJI Service Robot, introduced in Section 2, with the aim of creating a general system, able of doing several tasks, without being specifically programmed for a particular skill. The main difference of this robot with respect to the older prototype[25] is that a three-fingered Barrett Hand has been installed (see Fig. 10), that allows for more advanced manipulation capabilities.

6.1. Working environment

The proposed work environment is the library of the Universitat Jaume I in Castellón, Spain. We have defined a model of this library, allowing the robot to approximately know its structure. Although the model has been defined with concrete information of our library, it can be easily adapted to any other library by the following user-configurable parameters:

- The number of buildings
- The number of floors in each building
- The number of bookcases in each floor, with their respective location and content
- The number of modules and shelves that compose a bookcase.

We assume that books are classified using the Library of Congress Classification (LCC), which associates a code (also called signature) to each book. In libraries following this classification, books are normally grouped by their knowledge area, in such a way that, given a book code, we know where to find it approximately. We store this knowledge into the bookcase representation: in addition to the bookcase location, the signature of the first book that it contains is also stored. Knowing the first book of each bookcase, and assuming that books are correctely arranged, it is easy to detect which bookcase contains the book we are searching for.

6.2. Description of the application

For performing these tasks, several software modules have been implemented:

- User interface module, which is in charge of transferring user commands to the robot. It must allow the operator to remotely control the robotic scenario, as well as obtaining monitoring information through sensors and cameras. On the other hand, the user interface can be used in both offline and online ways. It means the 3D simulator would allow the implementation of robotic tasks in simulation and sending the commands to the real scenario afterwards.
- Navigation module, which guides the robot through the library for arriving to a desired point. This point could have been indicated by the remote operator, or computed automatically into autonomous operation mode.
- Vision module, which takes images from the camera and processes them in order to locate and recognize book labels.
- Grasping module, in charge of controlling the robotic arm and gripper for retrieving a book.

A snapshot of the operator interface is shown in Fig. 11. It shows a rendered model of the library that replicates the real world. The robot is continuously drawn at its current position. If it changes its position (or arm angles), the new configuration is automatically reflected into the user's view. With this utility, the operator can know at any moment the position of the robot. The interface also lets the operator to select between different predefined views, or to move the current view in any direction.

For the remote control of the robot, the right panel of the application can be used. The remote operator can also receive, if requested, a live video stream of the robot's camera, that is shown in a small window inside the application.

The robot autonomously locates the bookcase where the book is to be found, and plans a path to the target position. For details, see ref. [26]. When the robot is searching for a book in autonomous mode, it navigates through the library without hitting people and obstacles. For this, off-the-shelf algorithms included in the robot's programming libraries are used.

When the robot arrives in front of the bookcase that should contain the book, it does not know the exact position of the book. It uses vision for locating, reading and comparing book labels. The vision module includes algorithms for correct camera positioning, image segmentation, tracking of book labels and optical character recognition (OCR). These algorithms are detailed in ref. [8]. The robot starts searching for the book on the shelves and modules of the predicted bookcase. Instead of searching for the book one by one in the whole bookcase, an intelligent search algorithm has been

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

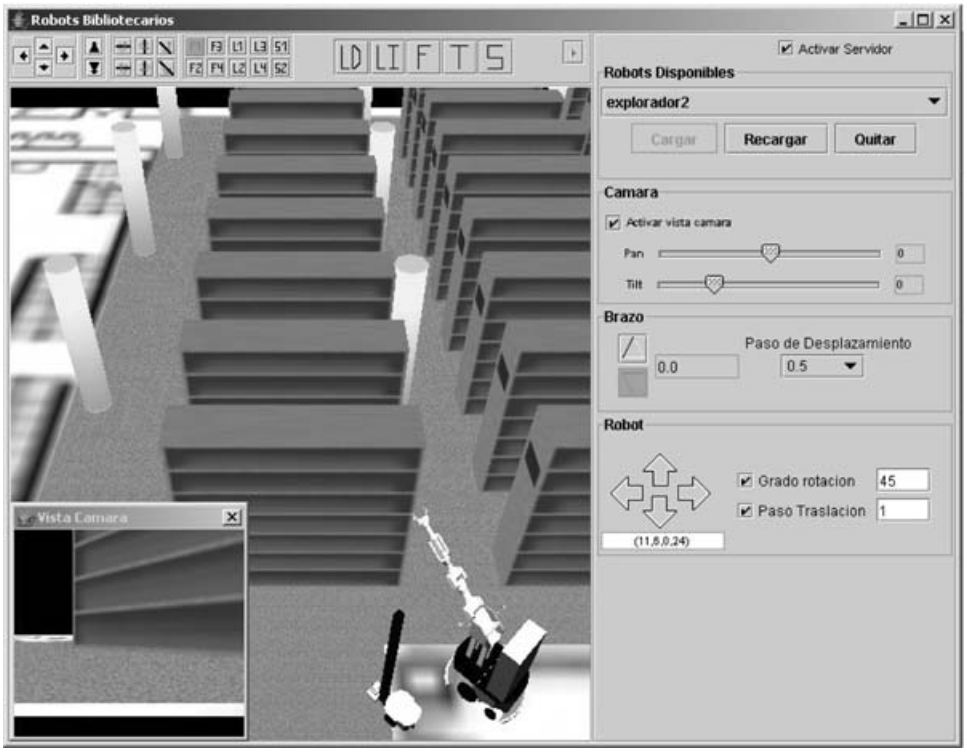


Fig. 11. Operator interface.

implemented[26] that takes advantage of the fact that books follow an order in its arrangement.

When the desired book is found in the image, the grasping process starts. With the parallel jaw gripper, the approach was to grasp the book from the front.[8] Using force feedback, both claws were opened until the distance between them fit with the width of the book. With the Barrett hand, a more natural approach is adopted by imitating human movements. This approach is explained in the next section, where we want to show how the proposed architecture is general enough for performing very different tasks.

Moreover, as introduced before, the user interface can be used in offline mode for those situations where we want to implement a sophisticated task, which the robots are not able to perform at that moment. In fact, the user interface provides a way for external applications sending commands to the simulation. As this interface goes on top of HTTP, moving a robot a position is as simple as opening a Uniform Resource Locator (URL). Here, we have an example of loading a given book into the 3D simulation scenario from MATLAB:

```
urlread('http://localhost/command', 'GET',
[{'name''loadBook'}{'param1'positionBook}]);
```

Then, by using a tool such as Matlab for programming robotic algorithms, we can send the results of these programs to the 3D simulator in a simple manner.

Figures 12 and 13 show snapshots of a MATLAB program that connects to the 3D simulator to look for a specific book in the library, using simple colour matching techniques. The search is performed in parallel by two mobile robots and once the book is located they call the robot manipulator to pick up the book.

6.3. Grasping a book

The architecture presented in Section 4 allows for the execution of complex tasks such as taking a book out of the bookshelf. We had already addressed this problem[8], by using a simple parallel jaw gripper, which limited the manipulation capabilities of the system. With the recently installed Barrett's Hand, we 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 14 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 Fig. 14 depicts, the initial task frame, drawn in red, 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 8 N, which is suitable for not damaging the book. At this point, the robot starts moving in X direction of the task frame, while still making 8 N of force along Z direction, and rotating around Y axis. This motion is managed by the compliant impedance controller of Section 3. When robot's perceptual modules detect that the angle with respect to the original position is higher than 15°, 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 15 represents the forces that appear during the task and the consequences on the robot's behaviour. The

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

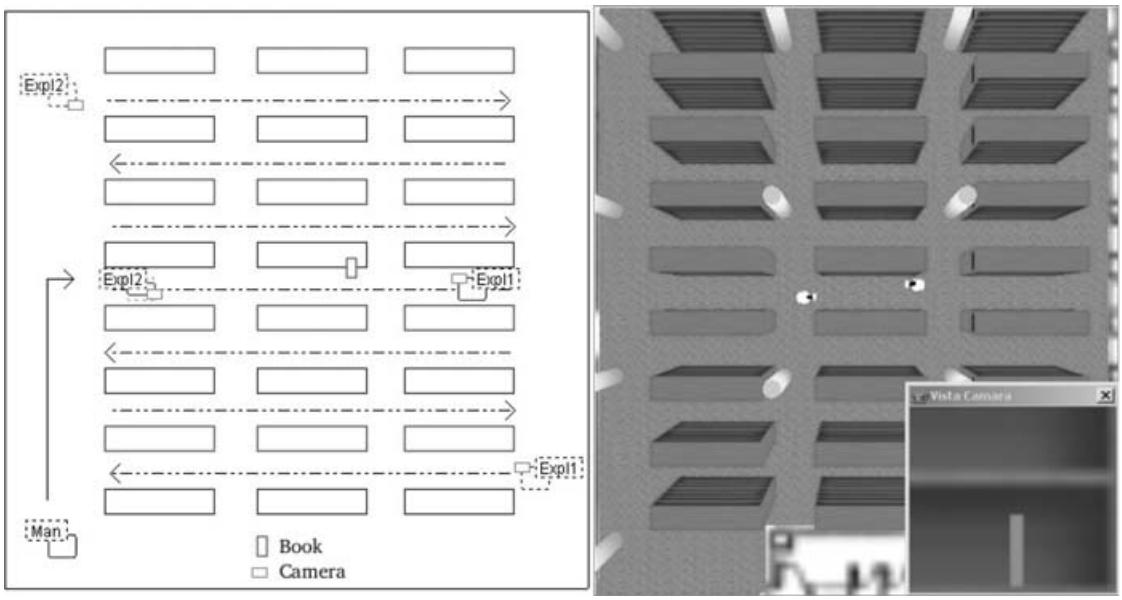


Fig. 12. Snapshots of the mobile robots searching for the book in simulation.

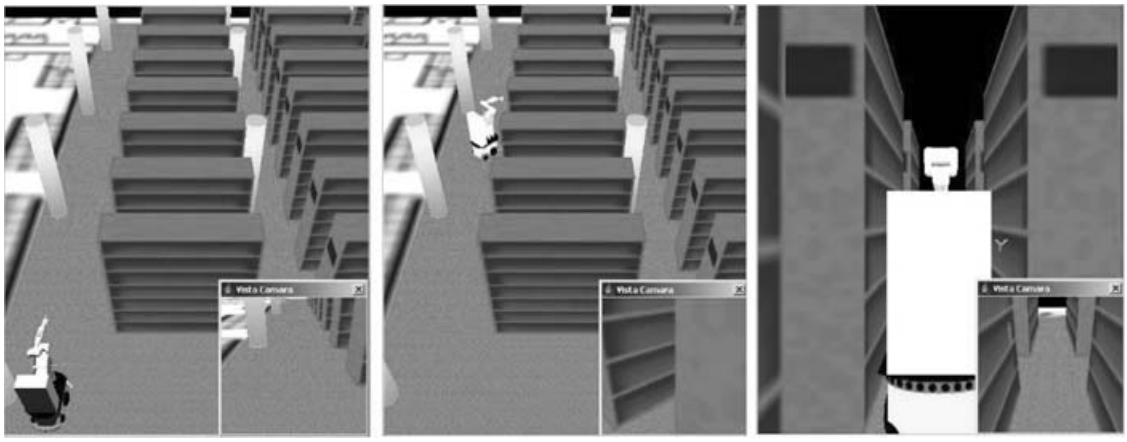


Fig. 13. Snapshots of the manipulator going towards the book in simulation.

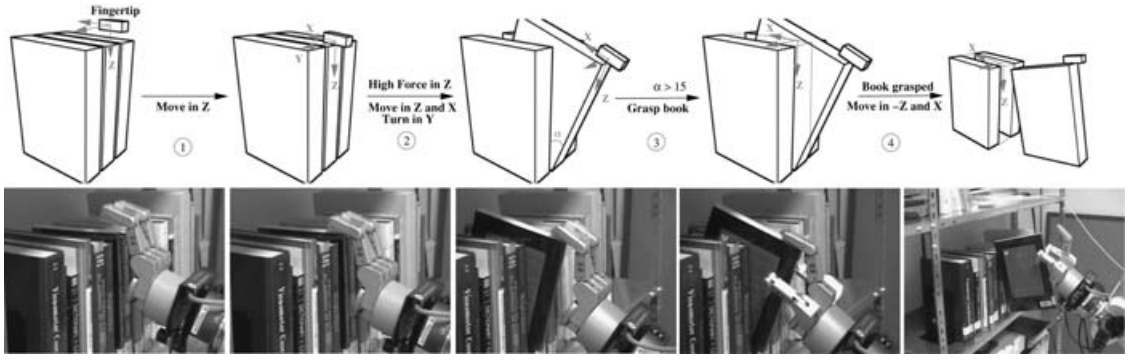


Fig. 14. Book extraction task.

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 8 N. At this point, the velocity along Z direction is approximately zero, and the robot starts moving along lit 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 8 N must be kept constant during the second stage, as is shown in the figure. Step 3 corresponds to the point where the fingers

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

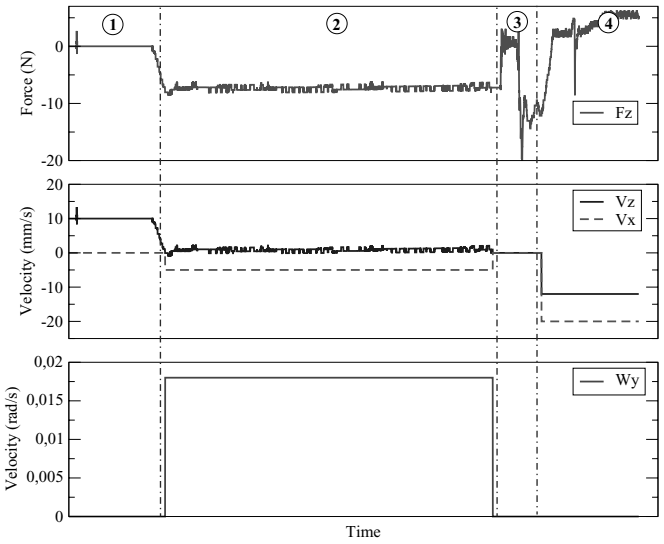


Fig. 15. Forces and velocities during the book task.

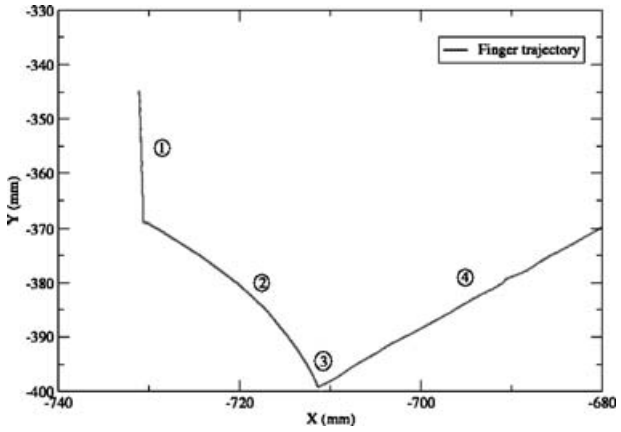


Fig. 16. Fingertip trajectory during the book task.

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 16 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.

7. Conclusion and Future Lines

In the execution of manipulation tasks, external forces play an important role, and must be taken into account for avoiding damage and failure. This is more important when the robot has to perform service tasks in everyday human environments, such as a library. 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 making the UJI Service Robot to turn a door handle. Results show how the robot smoothly adapts its behaviour according to the forces that appear during the execution of these tasks, making it robust to environmental changes or modelling errors. The architecture has been used for programming the new version of the UJI Librarian Robot, making important progress in the way the robot takes a book.

The two tasks presented here are human daily tasks, which are achieved by explicitly telling the robot what to do and trying to imitate what a human does (see Fig. 17). This fact has been possible due to the similarity between the human arm anatomy and the used system. Due to this similarity, the establishment of a correspondence between them and subsequent adaptation of movements has been possible. However, the inclusion of new skills in the robot with the aim of enabling complex behaviours, by means of observing the performance of the same actions carried out by humans, is an ongoing project for us at the moment. Initially, we are trying to follow the ideas of other experts in this matter like the paradigm of "task-level imitation"[27] based on learning how to perform a grasping action without imitating the exact movement of the demonstrator.

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 capable of grasping and moving objects, but also autonomously performing common daily tasks such as opening doors and drawers, switching on the light, grasping books, etc. in a dependable way.



Fig. 17. Book extraction task performed by a human.

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press

Acknowledgments

This paper describes the 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

1. A. Bicchi and V. Kumar, "Robotic Grasping and Contact: A review," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA (2000) pp. 348–353.
2. A. M. Okamura, N. Smaby and M. R. Cutkosky, "An Overview of Dexterous Manipulation," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA (2000), pp. 255–262.
3. C. Ferrari and J. Canny, "Planning Optimal Grasps," *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France (1992) pp. 2290–2295.
4. C. Borst, M. Fischer and G. Hirzinger, "A Fast and Robust Grasp Planner for Arbitrary 3D Objects," *Proceedings of the IEEE International Conference on Robotics and Automation* (1999) pp. 1890–1896.
5. A. T. Miller, S. Knoop, H. I. Christensen and P. K. Allen, "Automatic Grasp Planning Using Shape Primitives," *Proceedings of the IEEE International Conference on Robotics and Automation* (2003) pp. 1824–1829.
6. M. Mason, "Compliance and force control for computer-controlled manipulators," *IEEE Trans. Syst., Man, Cybern.* **11**(6), 418–432 (1981).
7. T. Kröger, B. Finkemeyer, U. Thomas and F. M. Wahl, "Compliant Motion Programming: The Task Frame Formalism Revisited," *Proceedings of Mechatronics & Robotics*, Aachen, Germany (Sep. 2004) pp. 1029–1034.
8. M. Prats, P. J. Sanz and A. P. del Pobil, "Model-Based Tracking and Hybrid Force/Vision Control for the UJI Librarian Robot," *Proceedings of the International Conference on Intelligent Robots and Systems*, Edmonton, Canada, 2005 pp. 1090–1095.
9. F. Marrone, F. Raimondi and M. Strobel, "Compliant Interaction of a Domestic Service Robot with a Human and the Environment," *Proceedings of the 33rd International Symposium on Robotics*, Stockholm (Oct. 2002).
10. D. Austin, L. Petersson and D. Kragic, "High-Level Control of a Mobile Manipulator for Door Opening," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Kagawa, Japan (Oct. 2000) pp. 2333–2338.
11. U. Thomas, B. Finkemeyer, T. Kröger, and F. M. Wahl, "Error-Tolerant Execution of Complex Robot Tasks Based on Skill Primitives," *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan (Sep. 2003) pp. 3069–3075.
12. T. Hasegawa, T. Suehiro and K. Takase, "Model-based manipulation system with skill-based execution," *IEEE Trans. Robot. Autom.* **18**(5), 535–544 (1992).
13. J. Baeten, H. Bruyninckx and J. De Schutter, "Integrated vision/force robotic servoing in the task frame formalism," *Int. J. Robot. Res.* **22**(10–11), 941–954 (2003).
14. T. Tomizawa, A. Ohya and S. Yuta, "Remote Book Browsing System Using a Mobile Manipulator," *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan (Sep. 2003) pp. 256–261.
15. J. Suthakorn, S. Lee, Y. Zhou, R. Thomas and S. Choudhury, "A Robotic Library System for an Off-Site Shelving Facility," *Proceedings of the IEEE International Conference on Robotics and Automation* (2002) pp. 3589–3594.
16. M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *ASME J. Dynam. Syst. Meas. Control* **77**, 126–133 (Jan. 1981).
17. N. Hogan, "Impedance control of industrial robots," *Robot. Comput.-Integr. Manuf.* **1**(1), 97–113 (1984).
18. J. De Schutter and H. Brussels, "Compliant robot motion. i. A formalism for specifying compliant motion tasks," *Int. J. Robot. Res.* **7**(4), 3–17 (1988).
19. J. Schutter and H. Brussels, "Compliant robot motion. ii. A control approach based on external control loops," *Int. J. Robot. Res.* **7**(4), 18–33 (1988).
20. G. A. Bekey, *Autonomous Robots* (MIT, Cambridge, MA, Jun. 2005).
21. L. Petersson, M. Egerstedt and H. Christensen, "A Hybrid Control Architecture for Mobile Manipulation," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, South Korea (Oct. 1999) pp. 1285–1291.
22. L. Sciavicco and B. Siciliano, "Modelling and Control of Robot Manipulators," **In:** Advanced Textbooks in Control and Signal Processing (Springer-Verlag, Berlin, Germany, 2000).
23. R. C. Arkin, *Behavior-Based Robotics* (MIT, Cambridge, MA, May 1998).
24. A. Orebäck and H. I. Christensen, "Evaluation of architectures for mobile robotics," *Autonom. Robots* **14**(1), 33–49 (Jan. 2003).
25. R. Ramos-Garijo, M. Prats, P. J. Sanz and A. P. del Pobil, "An Autonomous Assistant Robot for Book Manipulation in a Library," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Washington D.C., USA (Oct. 2003) pp. 3912–3917.
26. E. Martínez, M. Prats, A. P. del Pobil and P. J. Sanz, "Robots Behave in the Real World: Looking for Books in a Library," *Proceedings of the 9th IASTED International Conference on Artificial Intelligence and Soft Computing*, Benidorm, Spain (2005) pp. 325–330.
27. G. Simmons and Y. Demiris, "Imitation of Human Demonstration Using a Biologically Inspired Modular Optimal Control Scheme," *IEEE-RAS/RSJ International Conference on Humanoid Robots*, Los Angeles, USA (2004) pp. 215–234.

https://doi.org/10.1017/S0263574706003304 Published online by Cambridge University Press