Autonomous Robots manuscript No.
(will be inserted by the editor)

A Framework for Compliant Physical Interaction

The grasp meets the task

Mario Prats · Pedro J. Sanz · Angel P. del Pobil

Received: date / Accepted: date

Abstract Although the grasp-task interplay in our daily life is unquestionable, very little research has addressed this problem in robotics. In order to fill the gap between the grasp and the task, we adopt the most successful approaches to grasp and task specification, and extend them with additional elements that allow to define a grasp-task link. We propose a global sensor-based framework for the specification and robust control of physical interaction tasks, where the grasp and the task are jointly considered on the basis of the *task frame formalism* and the *knowledge-based approach* to grasping. A physical interaction task planner is also presented, based on the new concept of task-oriented hand preshapes. The planner focuses on manipulation of articulated parts in home environments, and is able to specify automatically all the elements of a physical interaction task required by the proposed framework. Finally, several applications are described, showing the versatility of the proposed approach, and its suitability for the fast implementation of robust physical interaction tasks in very different robotic systems.

M. Prats
Computer Science and Engineering Department
Jaume-I University, Castellón, Spain
E-mail: mprats@icc.uji.es

P.J. Sanz
Computer Science and Engineering Department
Jaume-I University, Castellón, Spain
E-mail: sanzp@icc.uji.es

A.P. del Pobil
Computer Science and Engineering Department
Jaume-I University, Castellón, Spain
and
Department of Interaction Science
Sungkyunkwan University, Seoul, South Korea
E-mail: pobil@icc.uji.es

1 Introduction

Robotic manipulation is still far away from the robustness and versatility offered by human infants, or even primates. Even though decades of research in robotic grasping and arm control have lead to fruitful results, very few of the robots currently found in research labs are able to perform tasks that go beyond pick and place in a dependable fashion. In the last years a remarkable number of mobile manipulators and complex humanoid robots have been developed, some of them endowed with advanced arms and hands. However, even though most of these robots have been designed with the purpose of assisting people at their homes, effort seems to be focused on locomotion, leaving manipulation almost unaddressed.

Several initiatives for building a multipurpose assistive mobile manipulator exist. One example is the PR-1 robot prototype (Wyrobek et al, 2008), developed by the Willow Garage company in its Personal Robotics Program. This robot is the first of a series of personal robots designed for assisting people in human environments. The STAIR (STanford AI Robot) project (Quigley et al, 2007) is aimed at building a mobile manipulator that can navigate through home and office environments, interacting with objects (Saxena et al, Feb 2008), and opening doors (Petrovskaya and Ng, 2007). The El-E helper robot, developed at Georgia Institute of Technology, mimics the capabilities of service dogs by grabbing hold of a towel to manipulate doors and drawers (Nguyen and Kemp, 2008). The same group was previously inspired by helper monkeys for grasping (Kemp et al, 2008). Care-O-bot II, designed for elderly care, included a manipulator that allowed him to perform vision-based fetch and carry tasks (Graf et al, 2004). Although research in humanoid robots is mostly

focused on locomotion aspects, some of these robots have also been used for manipulation purposes. Two outstanding examples are the Armar-III robot (Asfour et al, 2006) and the HRP-2 (Kaneko et al, 2004). Most of the previous robots are part of a long-term project for building a useful robotic assistant. However, robustness and versatility seem to be two important limiting factors.

Among the applications that have been addressed in the literature, it is worth mentioning solutions for cleaning (Marrone et al, 2002), contour following (Baeten et al, 2003), assembly (Thomas et al, 2003), and, especially, door opening (Niemeyer and Slotine, 1997; Petersson et al, 2000; Ott et al, 2005; Petrovskaya and Ng, 2007). However, most of the applications found in the literature only provide specialized controllers for specific actions, which cannot easily scale to deal with different tasks or systems.

The grasp-task connection has been rarely considered in the literature. In general, approaches to grasping consider only pick and place tasks, and approaches to task control consider that a suitable grasp has been already performed. There are different approaches to grasp planning, task planning and sensor-based control. They have never been considered as a related problem, nor have they been addressed from a common framework.

In our opinion, one of the reasons for the little number of robots actually using their hands for performing physical interaction tasks in real environments is the lack of a well-established methodology allowing for both grasp and task specification, planning and real-time dependable control. In this article, we propose a more general approach to manipulation, where the grasp and the task are jointly considered, in a global framework based on multisensor information, that allows for real-time and real-life dependable physical interaction. The second main contribution is a physical interaction task planner, which specifies automatically all the required elements of a physical interaction task, based on the proposed framework. Finally, several real applications of the framework and the planner are described, involving different robotic systems, and focusing on the interplay between the grasp and the task.

1.1 Outline of our approach

Physical interaction is a term that will be used throughout this article in order to refer indistinctly to the grasp and the task. Throughout this paper, a *grasp* is understood as any set of contacts between the hand and the object, either constraining the object motion in all the directions, known as *prehensile grips* in the taxonomy of Cutkosky and Wright (1986), or constraining only some degrees of freedom (DOF's) in the case of *non-prehensile grasps*. Both the grasp and the task are addressed as *physical interaction* tasks.

The purpose of this paper is to address the following fundamental questions:

- How can everyday physical interaction be specified in a common framework, including both the grasp and the task, and supporting sensor-based control?
- How can a robot autonomously plan a physical interaction task, making use of this framework?
- What sensors are necessary, and how can a robot combine these sensors and control its motors for performing physical interaction tasks in a robust manner?

The ultimate goal is to develop a practical framework, appropriate for real-life experimentation, and to show its suitability for robust sensor-based implementation of physical interaction tasks in human environments.

This article is organized as follows: section 2 presents a survey on grasp and task planning; section 3 describes in detail the *Task Frame Formalism* and the *Knowledge-based approach* to grasping. Both of them represent the background of our framework, that is detailed in section 4; section 5 presents a physical interaction planner used for the automatic specification of physical interaction tasks, making use of the proposed framework; finally, section 6 describes several real applications, and discussion and conclusions are reported in sections 7 and 8.

2 State of the art

In this section, the state of the art in grasping and task planning and control is reviewed, paying special attention to the interplay between them.

2.1 Grasping

In our daily life, hands are used for restraining objects (Bicchi and Kumar, 2000), also called fixturing or prehensile manipulation; for manipulating objects with fingers, known as dexterous manipulation (Okamura et al, 2000), and for non-prehensile manipulation (Napier, 1956). The three cases have been addressed in robotics, although more attention has been put to prehensile manipulation. Two main approaches for object fixturing exist: the *contact-level approach* and the *knowledge-based approach*.

The contact-level approach considers the grasp as a set of contacts, each one transmitting a force-torque to the object. The purpose of a contact-level grasp synthesis algorithm is to find a set of contacts so that the space of all the possible forces/torques that can be applied to the object through the contacts, contains the space of all the possible disturbance wrenches. The space of all the possible disturbance wrenches depends on the task. In most of the cases, this space is modelled with a sphere, representing a generic task where disturbance forces and torques can appear in any direction.

Some researchers have argued that pure contact-level approaches usually do not take the hand constraints into consideration (Wren and Fisher, 1995; Miller and Allen, 1999; Morales et al, 2006; Huebner and Kragic, 2008), producing a set of contacts which are not reachable in practice. In fact, the lack of accurate sensors and errors in the models and robot-object positioning put difficulties to practical implementation. This argument is supported by the small number of articles that apply the contact-level approach with real robotic hands under realistic conditions.

An alternative approach is to grasp with predefined hand postures, which is called the *knowledge-based approach* to grasping (Stansfield, 1991; Okamura et al, 2000), or also the *qualitative approach* (Morales et al, 2006). These approaches consider a set of predefined prehensile patterns and classify them according to its suitability for the object geometry and for the task. The grasp planning problem is then reduced to the selection of the grasp preshape which best adapts to the particular geometry and task.

Some researchers have reported the importance of designing task-oriented grasping algorithms, defined as those which take into account the task requirements (Li and Sastry, 1987). After a detailed study of the human grasps, Cutkosky and Wright (1986) reported that the choice of a grasp depends mainly on the task to be performed, even more than on the object geometry. According to Li and Sastry (1987), stability is only a necessary condition for a good grasp, but is not sufficient: a good grasp should be task-oriented, i.e. able to generate body wrenches that are relevant to the task.

The first task-oriented grasp quality measure following the contact-level approach was presented by Li and Sastry (1987). The authors introduced the concept of *task ellipsoid* for modelling a non-spherical *task wrench space* (TWS), and defined a task-oriented grasp quality measure as the radius of the largest task ellipsoid which is contained in the *grasp wrench space* (GWS). Borst et al (2004) proposed a method to compute the task ellipsoid from the TWS, and to scale the GWS and TWS in order to transform the ellipsoid into a sphere, thus transforming the problem into the "sphere fitting" problem, which can be solved efficiently. Haschke et al (2005), approximated the task ellipsoid with a polytope, employing a set of task wrenches along the principal axis of the ellipsoid. Even though these approaches provide significant advances in task-oriented grasping, none of them have been applied in real robotic systems.

Regarding the knowledge-based approach to grasping, the task-oriented nature of the grasp has been rarely considered in practical robotics works. However, very important aspects of how humans use task information during grasping have been reported, for example, by Mackenzie and Iberall (1994). Lyons (1985) was one of the first authors in considering the task requirements into the selection of robotic grasps. However, only distinction between stability and precision requirements were made. One work which is close to our research is that of Bekey et al (1993), where the authors proposed a knowledge-based approach for grasp planning taking into account the object geometry, modelled with shape primitives, and the task requirements in terms of a set of heuristics taken from the human behavior. Our research extends this previous work in different ways. First, we do not only propose a grasp planner, but also a grasp-task specification approach and several sensor-based control methods. The physical interaction planner is just one part, built on top of a specification framework that allows to translate the grasp plan into a set of frames that can be directly used for control, and this is validated with several real experiments. Furthermore, the planner does not only consider grasping, but also how to interact with the object after the grasp has been performed. In fact, one of the main aspects of our research is that not only task constraints are taken into account during grasping, but also grasp-related aspects are considered during task execution. Both grasp and task controllers cooperate in order to robustly perform real manipulation tasks.

2.2 Task execution

Research in robotic task execution can be divided into global and local approaches. The global approach has been usually adopted by the motion planning communities (Latombe, 1991), which address the task planning problem as finding a joint space path for a high dimensional kinematic chain performing a desired end-effector motion in an environment with obstacles. In contrast, the local approach, normally adopted by the control communities, aims at computing an arm/hand control signal based on sensor information, for an instantaneous, possibly constrained, hand motion (Mason, 1981; Khatib, 1987). Whereas the former is suitable

for finding global and optimal solutions, it normally requires a perfect knowledge of the environment and intensive computational time. The local approaches are suitable when limited knowledge is available. The general procedure is to compute instantaneous joint motion based on sensor information, although they are subject to local minima and suboptimal motion.

At the control level, several concepts have been developed in order to assist the programmer in the task specification and control problem. Mason (1981) introduced the concept of *compliance frame*, as a coordinate system related to the task and aligned with the object *natural constraints*. *Position/force hybrid control* was introduced for the implementation of different control modes on each frame direction (Raibert and Craig, 1981). Khatib (1987) developed the *operational space formulation*, where the overall dynamic control of the robot is decomposed into *task behavior* and *posture behavior*. Bruyninckx and De Schutter (1996) formalized the original Mason's concepts into the *Task Frame* (TF) and *Task Frame Formalism* (TFF), which has been the most accepted methodology for specifying sensor-based control tasks. De Schutter et al (2007) realized that the TFF only applies to some task geometries, for which control modes can be assigned independently to each direction, and developed a more general approach, based on *feature frames*, where control references and constraints can be assigned to arbitrary Cartesian directions, and where several types of geometric uncertainty can be taken into account.

In general, the task planning and control approaches consider that a suitable grasp has been performed and that it remains suitable during the task. However, in practice, the grasp can be subject to important errors. In addition, the task forces may affect the state of the grasp during execution. The grasp is the link that allows to transform robot motion into environment motion. Thus, it is very important to take it into consideration from the task control point of view.

3 The bases of physical interaction

The most significant advances on the reciprocal relationship between the grasp and the task have been described in the previous section. Although the grasp-task interplay in our daily life is unquestionable, very little research has been performed in this line in robotics. Task planning and grasp planning communities have worked independently, and its intersection has received very little attention.

In order to fill the gap between the grasp and the task, we adopt the most successful approaches to grasp and task specification, and extend them with additional elements that allow to define a grasp-task link. As can be concluded from the previous section, these approaches are the Task Frame Formalism in the case of task specification, and the knowledge-based approach in the case of grasping. This section presents a detailed view of both techniques.

3.1 Task Frame Formalism

Compliant motion is a concept that refers to the motion of a robot manipulator when it is constrained by the task geometry. Opening a drawer, turning a door knob or polishing a surface are all examples of compliant motion tasks where the robot motion is constrained by a prismatic joint, a revolute joint and a planar contact respectively. In general, any compliant motion involves the lost of some degrees of freedom at the end-effector.

A robotic manipulator intended for compliant motion needs a task representation and specification approach that allows the programmer to define the task. The most accepted specification formalism for compliant motions tasks is the *Task Frame Formalism* (TFF).

The first contribution towards the TFF was proposed by Mason (1981), motivated by the increasing interest of finding an automatic method for the synthesis of control strategies for compliant motion, specially for automatic assembly applications. However, Mason only introduced the basic concepts of the formalism. It was Bruyninckx and De Schutter (1996) who proposed a clear and formal description of the TFF, including practical examples and reporting its limitations. This formalism establishes an intuitive approach to model a motion constraint, and to specify the desired forces and motions in a compatible and controller-independent manner.

Without considering the dynamic effects (i.e. from a kinetostatic point of view), constrained motion can be modelled as the relative instantaneous motion between two objects which does not generate any force other than frictional forces. One example of constrained motion could be the relative rotation between a door knob and the door itself, through the revolute joint that links them. Let $\mathbf{v} = (v_x, v_y, v_z, w_x, w_y, w_z)^T$ be a kinematic screw representing the relative velocity between the two linked objects, and $\mathbf{f} = (f_x, f_y, f_z, m_x, m_y, m_z)^T$ a wrench containing the forces and torques generated through the contact, a constrained motion in a frictionless environment must satisfy the following expression, known as the reciprocity condition:

$$\mathbf{v}^T\mathbf{f} = 0 \tag{1}$$

All the possible kinematic screw vectors holding the reciprocity condition compose the *twist space*. Similarly,

the *wrench space* is composed of all the possible wrench vectors reciprocal to all the velocities. For any kind of joint, it holds true that the twist and wrench spaces are disjoint sets. In addition, their union is always a six-dimensional vector space.

The Task Frame (TF) is an orthogonal basis that allows to specify all the possible reciprocal vectors. It must be set by the task programmer according to the task geometry, so that part of its axis (the velocity-controlled directions) are a basis for the twist vector space, and the rest of its axis (the force-controlled directions) are a basis for the wrench vector space. According to (Bruyninckx and De Schutter, 1996), this is a requirement, called *Geometric Compatibility*, that any TF should satisfy, although it can be violated in some cases for the sake of simplicity in the specification. In addition, the TF must always remain geometrically compatible during the task execution, as considered by the *Time-invariance* requirement of Bruyninckx and De Schutter (1996), which may require *tracking* the TF position and orientation during the task. An elementary task can be described, in terms of the TF, as a set of velocity references on the velocity-controlled axis, and force references on the force-controlled directions.

Our framework for describing physical interaction tasks is based on the TFF, mainly because of the following reasons:

- The TFF is a simple and intuitive approach to task specification, but still powerful.
- The TFF is suitable enough for the kinds of tasks that a home assistive manipulator should perform. Most of these tasks involve interacting with home appliances and articulated furniture, which normally require acting on prismatic and revolute joints.
- The TFF is suitable for autonomous planning of physical interaction tasks. A high-level task description on an articulated object can be easily transformed automatically into a TFF-based specification.

3.2 Grasp preshapes and shape primitives

The concept of *grasp preshapes*, also called *hand preshapes*, *hand postures*, or *prehensile patterns*, was firstly studied by Schlesinger (1919), in Germany, who proposed a simple grasp taxonomy for classifying the prehensile functionalities of prosthetic hands from an anatomical point of view.

A prehensile pattern is a hand configuration useful for a grasp on a particular shape and for a given task. The Schlesinger's classification was based primarily on the object shape, without considering the task to be performed with the object. Napier (1956) was the first to consider the intended task as key factor for selecting a suitable hand posture, together with the shape of the object, its size and miscellaneous factors such as weight and temperature.

These concepts were later adopted by the mechanics and robotics communities, the former for designing versatile robotic hands, and the latter as a useful way of reducing the complexity of planning grasps for a dexterous robotic hand. Cutkosky and Wright (1986) started from the original power and precision patterns of Napier's classification and developed a very complete taxonomy, designed to codify the knowledge required for manipulation tasks in a manufacturing environment. In the Cutkosky's taxonomy, several prehensile patterns were classified according to its task-related and object-related properties. This taxonomy was the first providing a quite complete systematic mapping between the task requirements on a given object shape and an appropriate grasp.

Since the publication of the Cutkosky's taxonomy, several researchers in the robotics community have adopted the grasp preshapes as a method for efficient and practical grasp planning in contrast to contact-based techniques. This new approach has received the name of *knowledge-based approach* to grasping (Stansfield, 1991). From the robotics point of view, a grasp preshape is a set of finger postures adopted as the wrist moves towards the object. The grasp is performed by moving the wrist to a suitable position close to the object, and then closing the fingers until contact is made.

A *shape primitive* is a simple geometry used to approximate an object shape, and over which a grasp preshape can be easily planned. A shape primitive, together with the intended task, determines a prehensile pattern. This idea was already introduced by Stansfield (1991) and Bard et al (1995), which modelled objects with elliptical cylinders, over which power grasps were easily computed. Miller et al (2003) proposed four different shape primitives which could be combined for modelling any object geometry: spheres, cylinders, cones and boxes. The authors developed a grasp planner which was later used by other authors for planning preshapes for a humanoid hand (Morales et al, 2006), or grasp planning on box-based object shape approximations (Huebner and Kragic, 2008), among others.

Grasp preshapes and shape primitives have shown to be an intuitive, efficient, practical and powerful approach to grasp planning, in contrast to contact-based approaches which rarely consider hand kinematics and task constraints. In addition, this approach is strongly based on medical studies of the human hand prehension functionalities, from the beginning of the 20th century

until nowadays. For all of these reasons, the part of our research devoted to grasp planning and control is based on the knowledge-based approach.

4 The grasp meets the task: a framework for compliant physical interaction

The TFF and the knowledge-based approach to grasping are tools that have been used independently by the task planning and grasp planning communities, but the relationship between them has been never considered. A joint framework linking both approaches would allow an integrated specification of the grasp and the task.

Based on these well-established theories, we develop a framework for the specification and robust control of physical interaction tasks, where the grasp and the task are jointly considered on the basis of multisensor information. In this section, the TFF and the knowledge-based approach to grasping are linked through a set of *physical interaction frames*. A physical interaction task is then described by a suitable placement of these frames, and the relationships between them. We study how sensors can be used for tracking the physical interaction frames, and show some conceptual examples of daily physical interaction tasks specified with the proposed approach.

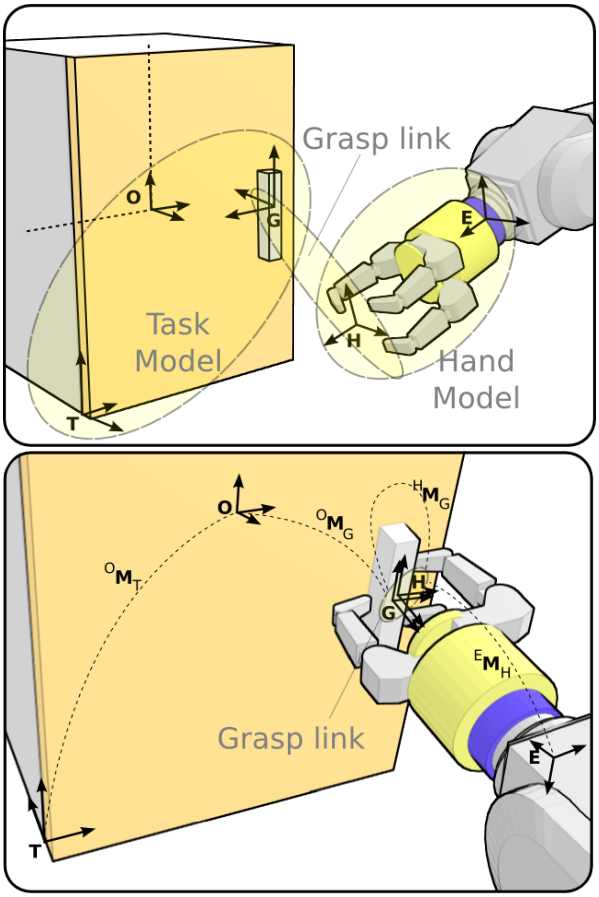


Fig. 1 The physical interaction frames are (top): the object frame (O), the end-effector frame (E), the task frame (T), the hand frame (H) and the grasp frame (G). The task motion must be transformed into robot coordinates through the kinematic chain formed by T, G, H and E (bottom). The *grasp link* is the relative pose between frames H and G, and represents the bridge between the task and the hand.

4.1 The physical interaction frames

In order to assist the specification of a physical interaction task, five auxiliary frames are used, as shown in Figure 1:

- The *Object frame (O)*, which is the origin where the object is defined.
- The *End-effector frame (E)*, where the control of the robot is performed. We focus on Cartesian control of the end-effector, because it has a direct mapping with the constrained motion, which is also specified in the Cartesian space. An inverse kinematics controller will be needed in order to transform Cartesian position/velocities into a suitable motion of the arm and the mobile platform.
- The *Task frame (T)*, where the task is specified according to the TFF. The programmer, or task planner, has to choose a suitable task frame according to the requirements originally described by Bruyninckx and De Schutter (1996).
- The *Hand frame (H)*, which is a coordinate system attached to the robot hand (or tool). It depends on the adopted hand posture and control strategy used for making contact. For control purposes, it is necessary to link the hand frame with the robot end-effector frame. This can be normally done through robot hand kinematics. In an analogy with the feature frames defined by De Schutter et al (2007), the hand frame can be placed on a *physical entity*, like the fingertip surface, or on an *abstract entity*, like the middle point in the imaginary line joining the thumb and the index fingers.
- The *Grasp frame (G)*, given in object coordinates, and related to the *task frame* through the object geometry, or through an user-defined transformation. This frame must be set to the part of the object which is suitable for grasping and task execution. It can also be placed on a *physical entity*, like a button surface, or on an *abstract entity*, like the symmetry axis of a handle.

Figure 1 shows an illustration of the different frames and its relationships, before and after the grasping action. The relationship between the physical interaction frames is represented with the following homogeneous transformation matrices: ${}^{E}\mathbf{M}_{H}$, ${}^{O}\mathbf{M}_{G}$, ${}^{O}\mathbf{M}_{T}$ and ${}^{H}\mathbf{M}_{G}$, relating, respectively, the end-effector frame to the hand frame, the object frame to the grasp frame, the object frame to the task frame and the hand frame to the grasp frame. Each transformation is composed of a rotation matrix and a translation vector, i.e. ${}^{i}\mathbf{M}_{j} = \begin{bmatrix} {}^{i}\mathbf{R}_{j} & {}^{i}\mathbf{t}_{j} \\ 0 & 1 \end{bmatrix}$, where ${}^{i}\mathbf{R}_{j}$ is the 3×3 rotation matrix between frames i and j, and ${}^{i}\mathbf{t}_{j}$ represents the position of frame j with respect to frame i.

The object, task and grasp frames compose the *task model*, whereas the end-effector and hand frames establish the *hand model*. It is worth noting that the specification of a task model does not necessarily require an object geometric model. The homogeneous transformations ${}^{O}\mathbf{M}_{T}$ and ${}^{O}\mathbf{M}_{G}$, can be set from an object model, but also from a user-defined specification not involving a geometric model, or from robot sensors, like the interactive perception approach of Katz and Brock (2008). Section 6 will show some examples of physical interaction tasks executed without relying on geometric models. The same concept applies to the hand model: ${}^{E}\mathbf{M}_{H}$ can be computed from hand kinematics, but also from visual feedback or a user-defined transformation. Thus, the framework is defined in an object-independent manner. The physical interaction frames can be set with or without using object geometric models.

The relative pose between the robot hand and the part of the object being manipulated, ${}^{H}\mathbf{M}_{G}$, depends on the particular execution and can be subject to important geometric uncertainties. This transformation matrix represents the link for transforming the task description into robot coordinates and must be continuously estimated by the robot sensors. We will refer to it indistinctly as the *grasp link*, the *hand-grasp link*, *hand-grasp transformation* or the *hand-grasp relationship*.

4.2 A physical interaction task

A physical interaction task is composed of the grasping action and the constrained interaction with the environment. Both the grasp and the task specification are detailed in the following points, based on the physical interaction frames defined previously. Some conceptual examples are provided.

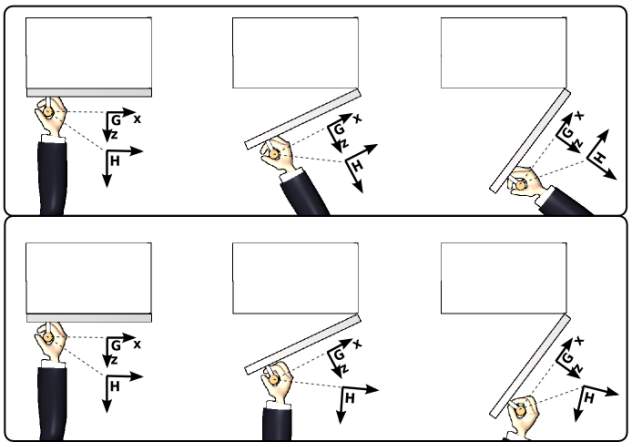


Fig. 2 A full-constrained grasp (top row) vs. an under-constrained grasp (bottom row). An under-constrained grasp is characterized because the grasp and the hand frames are not rigidly attached during the task execution.

4.2.1 The grasp

The grasping action is specified as moving the hand frame towards a desired relative positioning with respect to the grasp frame, i.e., taking the grasp link towards a desired configuration. Constrained and free degrees of freedom for the grasp must be indicated. For the constrained DOF's, the hand frame must completely reach the desired relative pose with respect to the grasp frame. However, for free degrees of freedom, there is no particular relative pose used as reference. Instead, the robot controller can choose a suitable pose, according to different criteria such as manipulability, joint limit avoidance, etc. We refer to these directions as *grasp-redundant DOF's*. A grasp with grasp-redundant DOF's is called an *under-constrained grasp*. An example of an under-constrained grasp is shown in Figure 2: the rotation around the handle axis is a grasp-redundant DOF that can be used by the controller in order to achieve a secondary task.

Let $\mathcal{P} = \{m_0, m_1, \ldots, m_n\}$ represent a hand preshape (either prehensile or non-prehensile), where m_i is the desired value for each of the n DOF's of the hand. The grasp is then defined as:

$$\mathcal{G} = \left\{\mathcal{P}, H, G, {}^{H}\mathbf{M}_{G}^{*}, \mathbf{S_c}\right\} \tag{2}$$

where ${}^{H}\mathbf{M}_{G}^{*}$ is an homogeneous transformation matrix that contains the desired relationship between the hand and the grasp frame, whereas $\mathbf{S_c}$ is a 6×6 diagonal selection matrix which indicates the Cartesian degrees of freedom constrained by the grasp. A value of 1 at the diagonal element i indicates that the corresponding DOF is constrained by the grasp, whereas a value of 0 indicates that it is not. Therefore, the grasp is specified as a desired relative pose (possibly under-constrained)

between the hand frame and the grasp frame, i.e. a desired state of the grasp link.

4.2.2 The task

The task requires performing compliant motion, following a set of velocity/force references defined in the task frame, according to the TFF. It is defined as follows:

$$\mathcal{T} = \{T, \mathbf{v}^*, \mathbf{f}^*, \mathbf{S_f}\} \quad (3)$$

$\mathbf{S_f}$ is a 6×6 diagonal selection matrix, where a value of 1 at the diagonal element i indicates that the corresponding DOF is controlled with a force reference, whereas a value of 0 indicates it is controlled with a velocity reference. This matrix is equivalent to the *compliant selection matrix* of Raibert and Craig (1981). A velocity reference is suitable for tasks where a desired motion is expected, whereas a force reference is preferred for dynamic interaction with the environment, where no object motion is expected, but a force must be applied (for polishing a surface, for example). $\mathbf{v}^*$ and $\mathbf{f}^*$ are, respectively, the velocity and force reference vectors given in the task frame, T.

4.2.3 Examples

Figure 3 shows three daily physical interaction situations that can be specified with the proposed framework. The first is an example of a task where a dynamic interaction with the environment is desired. Instead of specifying a velocity, the task is described as a desired force to apply to a button, along Z axis of the task frame T. The hand frame, H, is set to the fingertip of a *one-finger preshape*. The grasp frame, G, is set to the button surface so that the desired hand-grasp transformation can be set to the identity, which corresponds to the case where the fingertip is in contact with the button. This is the most common example of a non-prehensile grasp. For this case, the robot may choose the most suitable rotation around Z axis of the hand frame. Thus, Z direction is set to be a free DOF.

In the second example, a rotation velocity around Z axis of the task frame, T, is desired in order to turn on the tap. The grasp frame, G, is set to a part suitable for grasping, whereas the hand frame is set to the middle point between the thumb and the index finger in a *pinch preshape*. For performing the grasp, the hand frame must match with the grasp frame, up to a rotation about Y axis, which is set to be a grasp-redundant DOF.

Finally, the third example shows an ironing task where both a velocity and a force reference are needed.

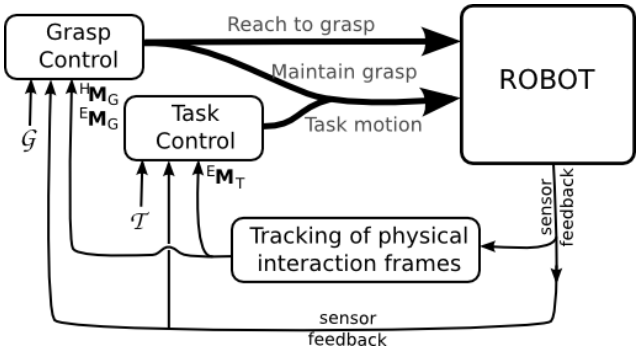


Fig. 4 The general control approach, composed of the *grasp controller* and the *task controller*, both relying on the sensor-based tracking of the physical interaction frames.

The Z axis of the task frame is force-controlled in order to produce some force against the ironing board. At the same time, axis X and Y are velocity-controlled in order to follow a particular trajectory, $\mathbf{f}(t)$. Regarding the grasp, a *cylindrical power preshape* is adopted, with a free DOF around Y axis of the hand frame, H.

In all of the cases, the velocity and force references have been manually set to a suitable value. When applying this framework to a real robot, these references must be set automatically by a planner, depending on the particular case.

4.3 Execution

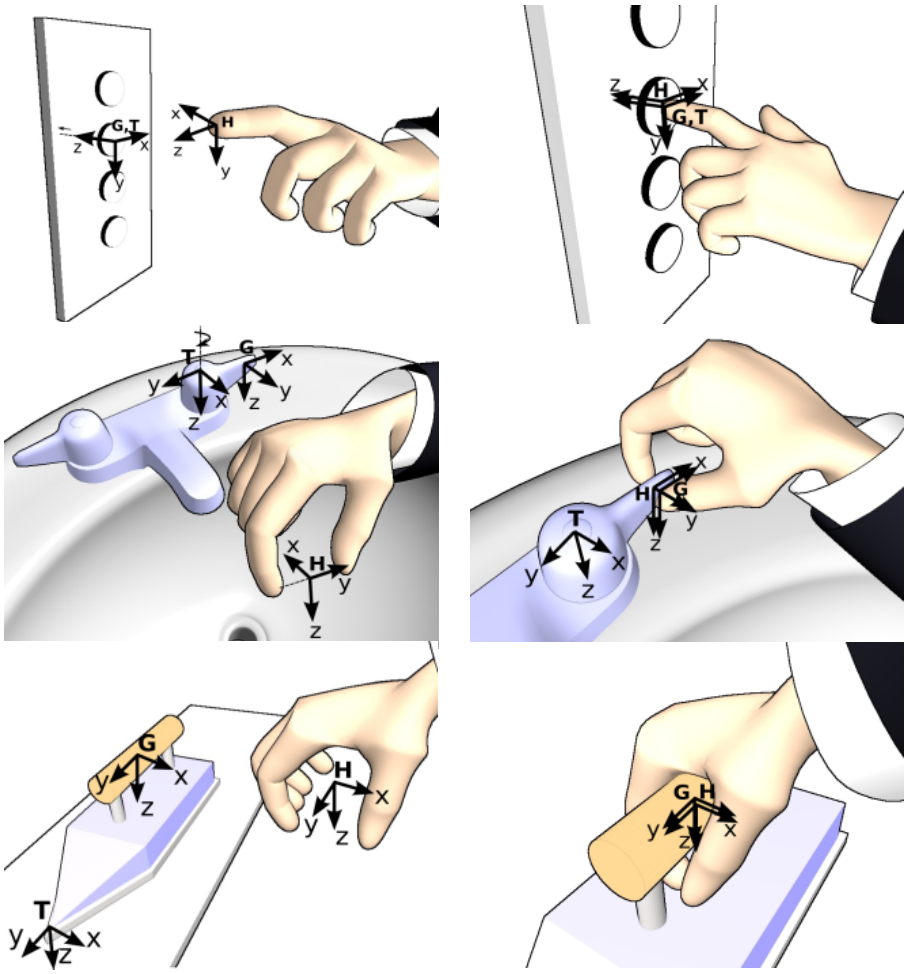
In the previous points we have defined the necessary elements for the specification of physical interaction tasks. In the following, some control guidelines are provided.

The execution of a physical interaction task can be divided into two steps:

- A *non-contact* phase, corresponding to the reach to grasp action, where the hand of the robot must be moved towards the object until the grasp is executed successfully.
- An *interaction* phase, where the hand is in contact with the object and the task constrained motion must be performed through robot motion, while keeping an appropriate contact situation.

During interaction, it is very important to maintain a good grasp on the object, specially for non-prehensile grasps which do not constrain all the relative hand-object motions. A suitable controller should perform the motion that ensures the execution of the task, but also auxiliary motion aimed at constantly improving, or at least maintaining, the grasp link.

Instead of defining a detailed control law at this level, we rather present the guidelines that a suitable



$\mathcal{P}$ = "*one-finger preshape*"
${}^{H}\mathbf{M}^{*}_{G} = \mathbf{I}_{4\times4}$
$\mathbf{S_c} = \text{diag}\,(1,1,1,1,1,0)$
$\mathbf{S_f} = \text{diag}\,(0,0,1,0,0,0)$
$\mathbf{v}^{*} = (0,0,0,0,0,0)$
$\mathbf{f}^{*} = (0,0,10\text{ N},0,0,0)$

$\mathcal{P}$ = "*pinch preshape*"
${}^{H}\mathbf{M}^{*}_{G} = \mathbf{I}_{4\times4}$
$\mathbf{S_c} = \text{diag}\,(1,1,1,1,0,1)$
$\mathbf{S_f} = \text{diag}\,(0,0,0,0,0,0)$
$\mathbf{v}^{*} = (0,0,0,0,0,0.01\text{ rad/s})$
$\mathbf{f}^{*} = (0,0,0,0,0,0)$

$\mathcal{P}$ = "*cylindrical power preshape*"
${}^{H}\mathbf{M}^{*}_{G} = \mathbf{I}_{4\times4}$
$\mathbf{S_c} = \text{diag}\,(1,1,1,1,0,1)$
$\mathbf{S_f} = \text{diag}\,(0,0,1,0,0,0)$
$\mathbf{v}^{*} = \mathbf{f}\,(t)$
$\mathbf{f}^{*} = (0,0,10\text{ N},0,0,0)$

Fig. 3 Some physical interaction situations described with the framework. First: pushing a button, with a force reference. Second: turning on a tap, with a velocity reference. Third: ironing task, with a velocity and force reference.

controller for physical interaction should follow. We propose a general control scheme, composed of two simultaneous controllers, as shown in Figure 4: the *grasp controller* and the *task controller*. The grasp controller is in charge of the reaching to grasp action, but also of the auxiliary motion required for maintaining an appropriate grasp condition during the task motion. Regarding the task controller, it is involved in the execution of the compliant motion required for the task. Both controllers rely on sensor information about the location of the physical interaction frames.

4.3.1 Grasp control

The grasp controller must ensure that the desired relationship between the hand and the grasp frame is achieved, at the same time that fingers are controlled to reach the desired hand posture. Using sensor feedback and an estimation of the state of the grasp link, the grasp controller generates control signals for the arm and for the hand (see Figure 5). The arm motion is in charge of approaching the hand to the target object, whereas the finger motion actually performs the preshaping of the hand.

For the arm motion, the simplest case is to design a proportional controller moving the hand in a straight line towards the target. Let ${}^{H}\mathbf{M}^{*}_{H} = {}^{H}\mathbf{M}_{G} \cdot \left({}^{H}\mathbf{M}^{*}_{G}\right)^{-1}$. Then, a proportional position-based control can be performed with the following equation, where $\mathbf{h}^{*}_{H}$ is a pose vector (with axis-angle representation of the orientation) build from the homogeneous matrix ${}^{H}\mathbf{M}^{*}_{H}$, λ_p is a control gain, and $\mathbf{v}^{G}_{H}$ is the resulting velocity of the grasp controller, given in the hand frame:

$$\mathbf{v}^{G}_{H} = \lambda_p \mathbf{h}^{*}_{H} \tag{4}$$

When the hand is far from the target, this controller is the simplest case of reaching. It can be done in open loop if a good estimation of the target pose with respect to the hand is available (Petrovskaya and Ng, 2007), but closed loop is more adequate if we want to deal with the uncertainties of non-structured environments in a sensor-based approach (Prats et al, 2008a). In closed

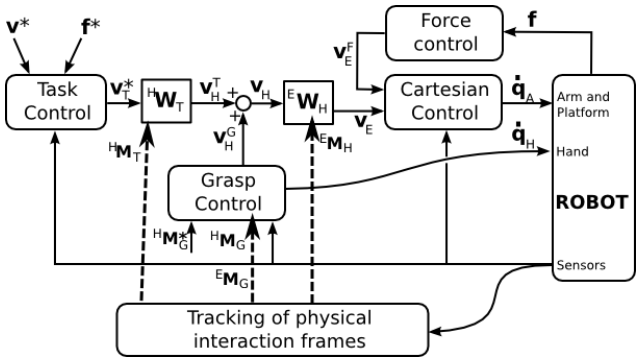


Fig. 5 A more detailed view of the control approach.

loop, the grasp link, ${}^{H}\mathbf{M}_{G}$, must be computed at each control cycle from the robot sensor feedback.

During task execution, the grasp controller can correct the relative misalignment's between the hand and the object, with the purpose of keeping a suitable grasp. Misalignment's can appear due to errors in the geometric models, robot calibration, or a TF specification violating the *Geometric Compatibility* requirement, for example. It is of utmost importance to keep an appropriate hand-grasp relationship during interaction, because the successful execution of the task depends on it.

Other more advanced control solutions are possible. In an environment where obstacles are modelled, a path planning algorithm could provide a feasible collision-free path, not necessarily in a straight line. It is also worth noting that the control equation 4 does not consider grasp redundancy, whereas it may be necessary for the success of some tasks. For an example of a controller considering grasp redundancy, refer to (Prats et al, 2008c). Our purpose in this section is only to present some general guidelines concerning implementation of the grasp controller. Section 6 will outline specific controllers, based on the general control scheme proposed here, combining force, visual and tactile feedback.

4.3.2 Task control

During the interaction phase, the robot hand is in contact with the environment, and any kind of uncertainty may produce considerable forces that can damage the environment or the robot. When the robot is in contact with the environment, it is extremely important to design a controller that can deal with unpredicted forces and adapt the hand motion accordingly.

The task controller must be able to transform a velocity/force reference, $\mathbf{v}^*$ and $\mathbf{f}^*$, given in the task frame, into a hand velocity, $\mathbf{v}_H^T$, and finally into joint motion, as shown in Figure 5.

The first step is to transform the task velocity/force reference, into a velocity reference given in the task frame, $\mathbf{v}_T^*$. The TF directions which are velocity-controlled can take directly the velocity reference, $\mathbf{v}^*$. However, those which are force-controlled need to transform a force reference, $\mathbf{f}^*$, into a position or velocity set-point.

This can be done with the *generalized spring*, which establishes the following relationship between force and position/velocity ($\mathbf{K}$ is the *stiffness matrix*):

$$d\mathbf{X} = \mathbf{K}^{-1}(\mathbf{f} - \mathbf{f}^*) \tag{5}$$

Therefore, it is possible to transform a velocity/force reference, $\mathbf{v}^*$ and $\mathbf{f}^*$, into one *force-based* velocity, $\mathbf{v}_T^*$, as:

$$\mathbf{v}_T^* = \mathbf{S}_f\mathbf{K}^{-1}(\mathbf{f} - \mathbf{f}^*) + (\mathbf{I} - \mathbf{S}_f)\mathbf{v}^* \tag{6}$$

The second step requires the transformation of the velocity reference, $\mathbf{v}_T^*$, given in the TF, to another velocity reference given in the frame where the Cartesian control of the robot is performed, i.e. the end-effector frame. This can be done by following the kinematic chain composed of the task, grasp, hand and end-effector frames:

$$\mathbf{v}_E = \underbrace{{}^{E}\mathbf{W}_H \cdot {}^{H}\mathbf{W}_G \cdot {}^{G}\mathbf{W}_T}_{{}^{E}\mathbf{W}_T} \cdot \mathbf{v}_T^* \tag{7}$$

where ${}^{i}\mathbf{W}_j$ is the 6×6 screw transformation matrix associated to ${}^{i}\mathbf{M}_j$.

Finally, the end-effector velocity must be transformed, by a low-level controller, from Cartesian space into joint space through the mobile manipulator jacobian. Force-torque feedback is a necessary condition (and, in some cases, sufficient) for robust manipulation. Whereas some tasks can still be performed when deprived of visual and tactile feedback, the force feedback cannot be removed if success in manipulation is desired. Thus, we assume that a force-torque sensor is always present, and that it performs the end-effector motion required for avoiding undesired forces.

4.3.3 Sensor-based tracking of the physical interaction frames

The general grasp and task controllers described in the previous points, depend completely on the relative pose between the physical interaction frames. The grasp controller needs to know the grasp link, ${}^{H}\mathbf{M}_{G}$, which is also required by the task controller, as part of the transformation ${}^{E}\mathbf{M}_{T}$ that links the task with the robot end-effector, i.e. ${}^{E}\mathbf{M}_{T} = {}^{E}\mathbf{M}_{H} \cdot {}^{H}\mathbf{M}_{G} \cdot {}^{G}\mathbf{M}_{T}$.

We support the idea that robot perception must continuously provide information about the relative pose

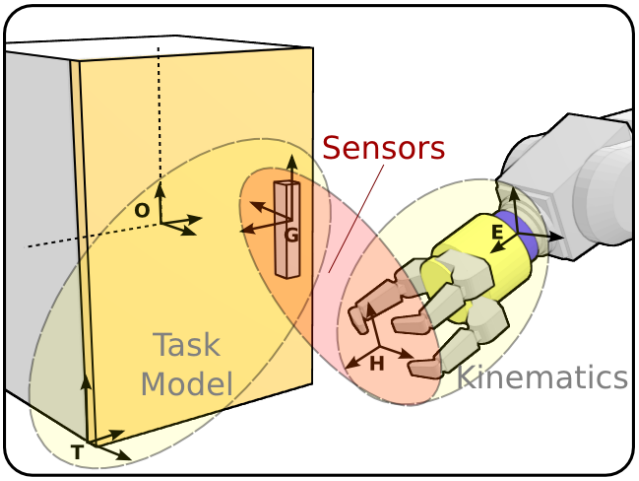


Fig. 6 The use of sensor feedback is specially important for the estimation of the grasp link.

between the physical interaction frames. In most of the cases, ${}^{E}\mathbf{M}_{H}$ and ${}^{G}\mathbf{M}_{T}$ can be computed from hand kinematics and the task model respectively. One exception, considered in detail in the next section, is the use of tools. In this case, the hand frame can be placed on the tool so that its position with respect to the end-effector cannot be computed directly from the hand encoders. Additional sensors would be needed in order to estimate the ${}^{E}\mathbf{M}_{H}$ transformation in this case.

But the transformation that must be definitively estimated through robot sensors is the grasp link, ${}^{H}\mathbf{M}_{G}$, because it can be subject to important execution uncertainties such as bad positioning, poor sensory information, sliding, etc. The robot must always estimate the hand-grasp transformation during task execution in order to assist the grasp and task controllers. This transformation is necessary for tracking the task frame position and orientation during interaction. The estimation of ${}^{H}\mathbf{M}_{G}$ is the key for relating the task frame to robot coordinates, thus allowing the transformation of the task constraints into suitable robot motion (see Figure 6).

The best sensor input to estimate this relationship is vision. A robot could be observing its hand and the object simultaneously, while applying model-based pose estimation techniques. Another interesting sensor is a tactile array, which provides detailed local information about contact, and could be used to detect grasp mistakes or misalignment's. In general, the best solution is to combine several sensor modalities for getting a robust estimation. Section 6 will describe several applications combining vision, force and tactile sensors with the purpose of monitoring the grasp link.

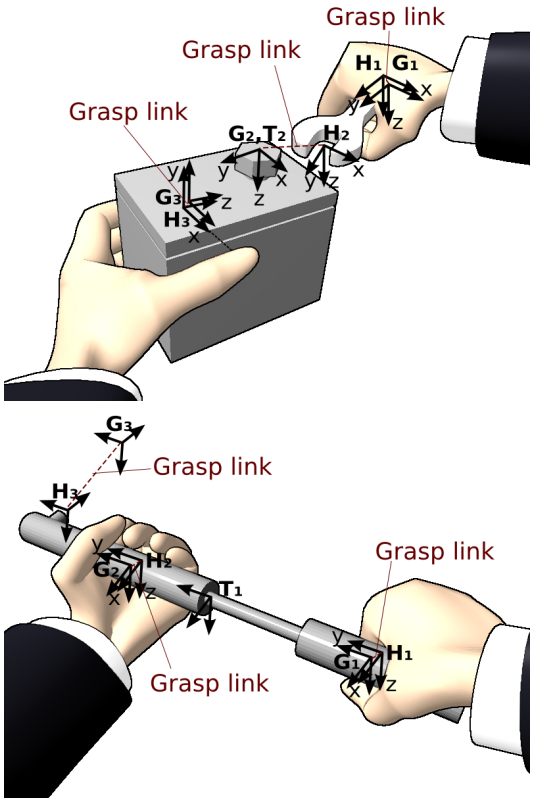


Fig. 8 Two examples of two-handed manipulation under the proposed framework. Top: one hand is holding an object while the hand uses a tool on it. Bottom: one hand holds a bicycle pump while the other performs the inflating task.

4.4 Use of tools

The proposed framework also supports the specification of physical interaction tasks involving tools. We can consider the tool as an extension to the hand, so that the hand frame can also be set to parts of the tool, and the framework can be applied normally. However, in this case, the relationship between the end-effector frame and the hand frame cannot be computed only through hand kinematics. It depends on the exact position of the tool in the hand, and must be captured by sensors once the utensil has been grasped. Tactile and vision integration can play an important role here.

Thus, the use of tools consists of two different instances of physical interaction tasks: one, involving a grasp for a transport task, and another one, involving the actual use of the tool.

In order to bring the tool to the object, the robot must first grasp the tool and pick it up. This task can be easily described with the proposed framework, by choosing a suitable hand preshape and grasp frame depending on the particular tool and the task to perform, as shown in Figure 7 (left).

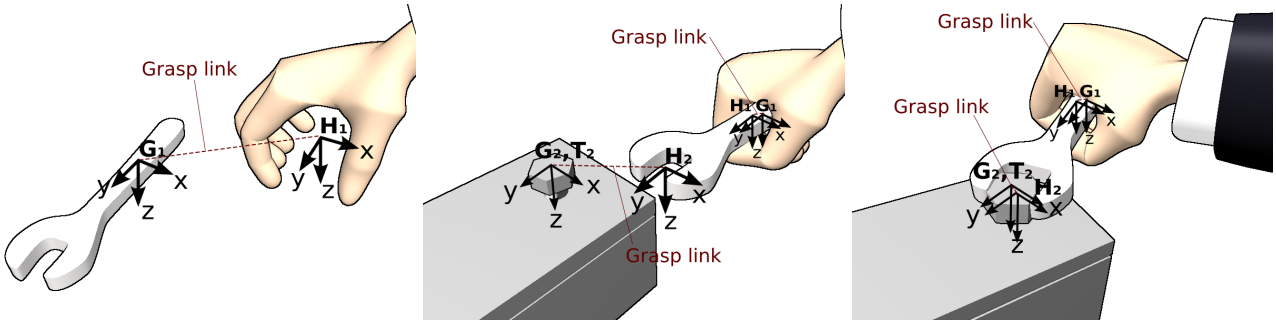


Fig. 7 An example of tool grasping and use based on the proposed formalism. Left: grasping the tool. Middle and right: using the tool

Once the tool is grasped, it can be considered as an extension to the robot hand, and the proposed formalism still applies. Using the tool requires moving the tool to the object (reaching) and performing the task motion. During the reaching phase, the hand frame can be set to a part of the tool suitable for making contact with the object. Therefore, reaching can be defined in terms of the proposed formalism, as an alignment between the hand frame, placed on the tool in this case, and the grasp frame, as shown in Figure 7 (middle and right).

When the tool is aligned with the object part, the task can be transformed from object coordinates to tool coordinates, and finally to end-effector coordinates, passing through two different grasp links. Sensor feedback must contribute in order to compute an estimate of the relative pose between, first, the object and the tool, and second, the tool and the hand. Therefore, the use of tools requires the application of the proposed formalism simultaneously for two different subsystems: hand-tool and tool-object.

4.5 Two-hand manipulation

In the case of physical interaction with two hands, we can even have three different subsystems running simultaneously under the proposed framework. For example, one hand could be holding an object, whereas the other hand could be holding a tool and acting on the object, as shown in Figure 8 (top). In this case, one physical interaction task would be needed for one hand (hand-object), whereas two physical interaction tasks would be specified for the other hand (hand-tool and tool-object). Another situation could be an inflation task, where a hand is holding a tool acting on an object, whereas the other hand works the tool (see Figure 8, bottom).

5 Planning of physical interaction tasks

In the previous section, we have proposed a framework for the joint specification of the grasp and the task. The elements necessary for the specification of the physical interaction task can be defined by a task programmer, but it is desirable to build those elements automatically with a physical interaction task planner.

From a given task description, the robot must be able to autonomously plan the physical interaction task, including the grasping action and subsequent compliant interaction, even for new objects. The robot must be able to adapt to the particular situation without being specifically programmed for it. In this section, we describe a physical interaction planner based on the framework for physical interaction. We focus on interaction in home environments, which includes manipulation of furniture and home appliances: opening doors and drawers, switching lights, turning knobs, etc.

In our way to describe the planner, the concept of *task-oriented hand preshapes* is introduced, as an extension to the classical understanding of hand preshapes. We define a set of *ideal task-oriented hand preshapes* for an *ideal hand*, and then propose the *hand adaptors*, as a method for the instantiation of the ideal preshapes in real robotic hands. The main advantage of this approach is that the same planning algorithms can be used for different hands, just by defining a suitable mapping between the ideal hand and a real one.

In order to assist automatic planning, we propose to represent an object as a set of parts that are linked kinematically. Each part is approximated with a box shape primitive, and labelled with a class name according to its function. Then, the concept of *object action* is introduced, as an object-centered description of a physical interaction task. Each object class provides a set of possible actions that can be performed on it. An *object task* is then described as a sequence of one or more object actions. Finally, we show how a physical interaction task can be automatically specified, taking as input an

object description and the action to perform on a given object part.

It is worth noting that the planner proposed in this section only considers a limited set of tasks. We present one specific planner for enabling interaction with articulated parts in home environments, and considering very basic geometry and a limited set of grasp preshapes. Other planners are possible that take into account other tasks, more complex geometry, additional preshapes, etc.

5.1 Task-oriented hand preshapes

In this article, the knowledge-based approach to grasping is adopted in a task-oriented manner. We extend the classical concept of hand preshape, as it is usually considered in robotics, for the inclusion of a task-oriented entity: the hand frame. As defined by the framework for physical interaction described in section 4, the hand frame is used to establish a link between the grasp and the task. Therefore, we introduce a task-oriented entity into the grasp definition with the purpose of enabling task-oriented grasping.

We define a *task-oriented hand preshape* as a hand preshape augmented with a hand frame. The hand frame is not only used for the grasp link specification, but also for describing the part of the hand that will be used for the physical interaction task. For example, pushing a button requires a contact with the fingertip, in contrast to a power grasp where contact is primarily done with the palm and the inner surface of the fingers. In the first case, the hand frame would be set to the fingertip, whereas in the second case it could be placed on the palm. One original hand preshape can generate several task-oriented hand preshapes, depending on the placement of the hand frame.

We define a set of task-oriented hand preshapes for an *ideal hand*, which is an imaginary hand able to perform all the human hand movements. The task-oriented hand preshapes defined for this hand are called the *ideal task-oriented hand preshapes*, or *ideal hand preshapes* for simplification. Figure 9 shows the ideal task-oriented hand preshapes considered by the planner. They are:

Hook power. The hand adopts a hook posture. Contact is performed with the inner part of the fingertips, near the palm. The thumb finger does not contribute to the grasp. This grasp is useful when the object can be enveloped with the fingers, and the task requires making force along one or two directions (Z and/or X in the hand frame). One example is turning a handle and pulling back.

Hook precision. The hand adopts a hook posture, as in the previous case, but the hand frame is placed at the fingertips. This grasp is suitable for pushing along one direction (Z in the hand frame), when it is not possible to envelope the object with the fingers. Pushing objects, or opening a sliding door are two examples for which this grasp could be needed.

Cylindrical power. The hand takes a cylindrical configuration, where the thumb is in opposition to the rest of the fingers. The hand frame is placed on the center of the polyhedron generated by taking the fingertips and the palm as vertex. This preshape is useful for enveloping objects and applying forces in all the directions.

Cylindrical precision. The hand takes a cylindrical configuration, as in the previous case. The hand frame is placed on the centroid of the polygon generated by taking all the fingertips as vertex. This preshape is useful for grasping small objects and applying forces along the X direction of the hand frame.

One-finger frontal. This preshape is useful for pushing small objects like buttons. Only the index finger is used, with the hand frame placed at the fingertip. Forces can be applied mainly along the positive sense of the Z axis.

One-finger precision. Similar to the previous case, but with the hand frame placed on the inner part of the fingertip. This preshape is useful for sensitive tasks, where tactile receptors at the fingertip play an important role. One example is grasping a book from a bookshelf, as we will see in Section 6.

Pinch. The pinch preshape is similar to the cylindrical precision preshape, but only the index and thumb fingers are used. It is suitable for grasping small objects or interacting with controls attached to prismatic joints (such as a volume bar, for example).

Lateral. The grasp is performed with the thumb finger in opposition to the lateral part of the index finger. The preshape is appropriate for precision grasps that require applying considerable forces and torques, such as unscrewing a bottle cap, turning a knob, etc.

Each of these grasp preshapes is suitable for a particular set of tasks. Precision preshapes are suitable for applying forces along well-known directions, in contrast to power preshapes, which can apply and resist forces in a wide range of directions. Therefore, if the interaction with the environment requires pushing along one known direction, *hook precision*, *cylindrical precision*, *pinch*, *one-finger frontal* and *one-finger precision* could be valid preshapes, depending on the object geometry. In contrast, interaction tasks requiring a wider force

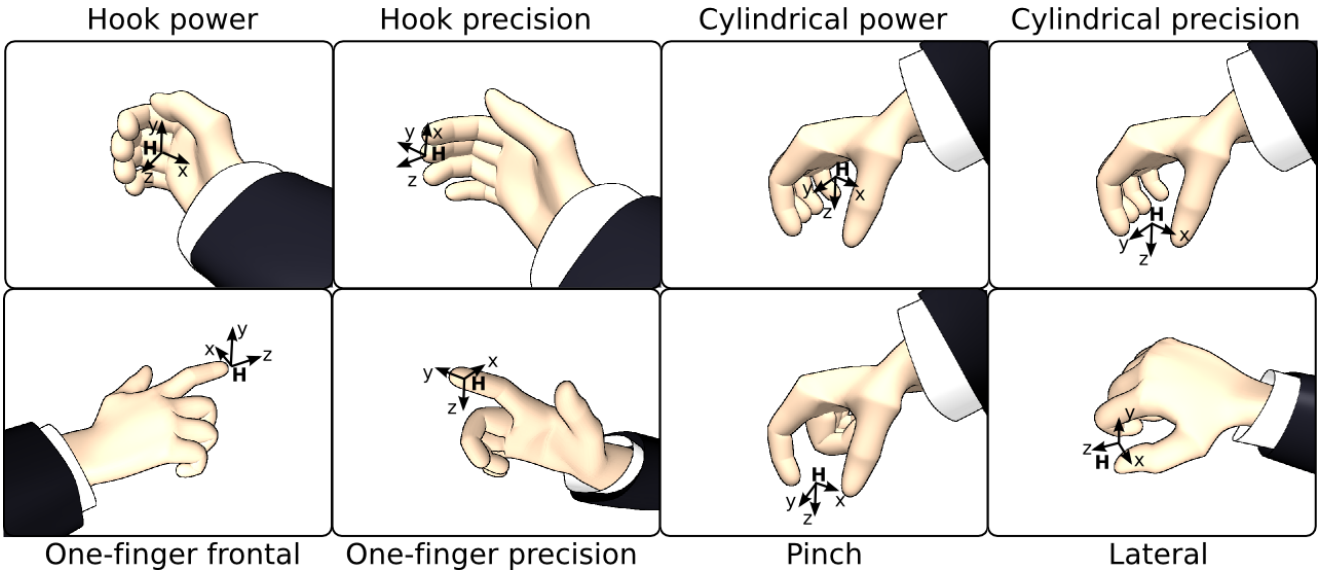


Fig. 9 The ideal task-oriented hand preshapes considered by the physical interaction task planner.

range would need a *hook power* or *cylindrical power* preshape. The *lateral preshape* is in the middle between precision and power, and should be used for interaction tasks that require a considerable force, when the object geometry does not allow a power grasp.

The physical interaction planner must choose the most suitable ideal preshape for the given object geometry and the intended task. For its application in a real robot, the ideal hand preshapes must be instantiated into real postures on a robotic hand. *Hand adaptors* are in charge of this. A hand adaptor is able to map an ideal preshape into a real robot preshape. The main advantage of planning on an ideal hand and adapting the results to the real case is that the physical interaction planner can be independent of a particular robotic hand. The result of the planner is general and can be easily applied in different robotic systems, just by defining the corresponding hand adaptor.

As an example, Figure 10 shows the adaptation of the ideal task-oriented preshapes to the Barrett Hand. Each of the ideal hand preshapes has a correspondence with a Barrett hand preshape, with the exception of pinch and lateral. These postures require further dexterity and must be replaced by one of the feasible configurations. Concretely, in the case of the Barrett Hand, the pinch and lateral preshapes are mapped to the cylindrical precision posture.

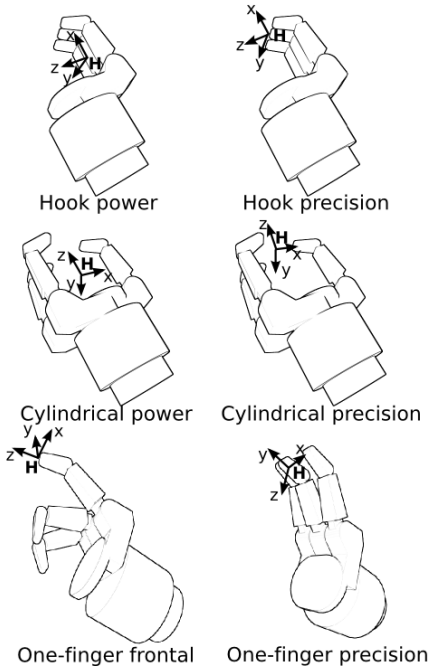


Fig. 10 The Barrett Hand task-oriented hand preshapes.

5.2 Object representation

The physical interaction planner needs a suitable task and object representation. From a description of the object and the task, the planner must be able to specify automatically all the elements of a physical interaction task. This section focuses on the object representation that we have adopted, whereas the next section will be devoted to the task description.

In our approach, the object is represented as a kinematic chain of shape primitives. Each object part is classified into an *object class*, according to its function. The object class can be part of the object description, or can be assigned automatically according to the geometry and type of the joint.

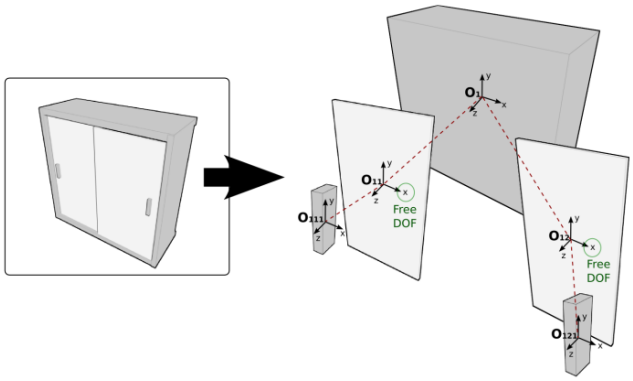


Fig. 11 An example of the structural model of a cabinet.

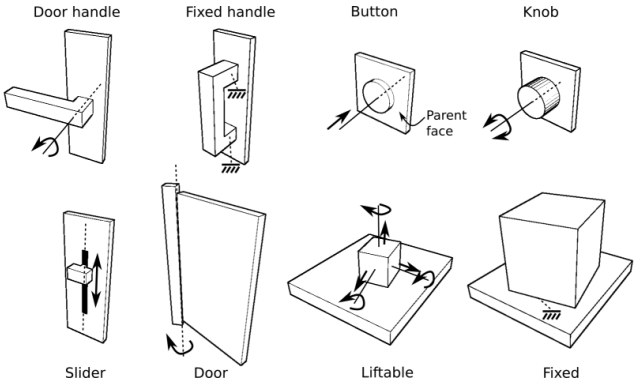


Fig. 12 The object classes considered by the physical interaction planner. Each object part can be automatically categorized into one class according to its geometry and joint type.

5.2.1 Object structural model

The object structural model consists of geometric and kinematic information. It is composed of several shape primitives, linked in a tree hierarchy. Each node corresponds to an object part, approximated with a shape primitive. Each part is defined on its own reference frame, which is independent from the other parts. Each link represents the relative position between two object parts, including a motion constraint modelled by one of the following joints: prismatic, revolute and fixed. A prismatic joint allows one translational DOF between two object parts. A revolute joint involves a rotational DOF, whereas a fixed joint represents that the two object parts are rigidly linked. Therefore, an object is recursively defined as the union of several, possibly articulated, sub-objects.

Only box shape primitives are considered at this moment, because of the following reasons:

- A box is simple to describe geometrically, and can be used efficiently. The most common physical properties as the mass center, volume, etc. can be computed easily.
- Any object geometry can be approximated by a bounding box. In addition, a box shape is specially well-suited for approximating furniture geometry.
- A box approximation is easier to obtain from sensor data than other geometries. See, for example, the approach of (Huebner et al, 2008), where arbitrary object geometries are approximated by box primitives from a 3D point cloud obtained by vision or range sensors. The box-shape approximation is later used for planning grasps (Huebner and Kragic, 2008).

Figure 11 shows the structural model of a cabinet furniture. There is a base box corresponding to the cabinet structure, which contains two sliding doors as child objects. Each of the doors contain a handle that is rigidly linked to them. Each box is defined in its own reference frame. The pose of any part with respect to the parent is represented by an homogeneous transformation matrix. The motion constraints are indicated as free DOF's in the local frames. For example, each of the doors can be moved freely along the X axis of their local frames.

5.2.2 Classification of object parts

The classification of object parts into categories has several advantages. First, the actions can be specified in an object-oriented way. Each type of object allows a limited set of actions. All the possible tasks that can be performed on an object are encoded in the object itself, without the need of an additional database encoding object-task relationships. Second, it allows to establish the link between a high-level task description and low-level actions. A high-level task description can be transformed into object actions, which are directly linked with the object degrees of freedom. And, third, the physical interaction planning problem can be divided into small sub-problems, one for each object class.

Each object part can be classified into one of the categories depicted in Figure 12, according to the object geometry and the type of joint to which they are attached:

Door handle. An elongated box primitive attached to the parent object through a revolute joint.
Fixed handle. A hand-size box primitive rigidly attached to the parent object.
Button. A compact shape linked to the parent through a prismatic joint whose axis is perpendicular to the parent face.
Knob. A compact shape linked to the parent through a revolute joint.

Slider. A compact shape linked to the parent through a prismatic joint whose axis is parallel to the parent face.

Door. A box with two dimensions considerably larger that the other one, and attached to the parent object through a revolute or prismatic joint with the joint axis along one of the largest dimensions.

Liftable. A small object that is not linked to a parent, and, thus, can be subject to free motion.

Fixed. A large object, not necessarily attached to a parent object, but for which motion is impossible due to its size. For example: the cabinet structure.

The object category can be part of the object description or can be performed by an automatic classifier. The former applies to an object recognition framework, where any object information can be stored in a database and accessed after the object is recognized. The latter can be used under an exploration framework, where the robot obtains a model on its own, without receiving external information.

Object class	Object action	Description
Door handle	Turn	Turns the handle in clockwise or counter-clockwise direction
	Push	Push the handle forwards of backwards (pull)
Fixed handle	Push	Push along one of the following directions: forwards, backwards, right, left, up and down
Button	Push	Activates the button
Knob	Turn	Rotate the knob in clockwise or counter-clockwise direction
Slider	Move	Translates the object part along the sliding mechanism
Liftable	Lift	A grasp on the object for transport purposes

Table 1 The object actions associated to each object class and its description.

5.2.3 Object Models

The planner described in this section assumes there is an object structural model available, even though it can be described by simple box-shape primitives. The use of a formal model allows the implementation of a general planner valid for different geometries and tasks. Even though a model is used by the planner for setting the physical interaction frames, it is worth noting that it represents only a specific planner, and that other solutions could be possible without using object models.

We would also like to clarify that it is not necessary to have the complete object model in order to apply the physical interaction planner. For example, one could have a vision algorithm for recognizing door handles. From the output of this algorithm, it could be possible to build a simple box-shape model with the handle size, and classify it with the *Door handle* label. This simple model could be used as input to the physical interaction planner. The same idea is applied to other object types, such as knobs, buttons, etc.

5.3 Task description

Each class name is associated with a set of actions: the *object actions*. Each object action has a direct correspondence with a physical interaction task. Table 1 shows the actions defined for each object class, and its low-level description.

Object actions provide an intuitive way to command physical interaction tasks at a mid-level language. However, commands can be given at a higher level. We define an *object task* as a name given to a sequence of one or more object actions. For example, "increase volume" is an object task involving the object action of "turn knob", "Switch on the television" corresponds to "push button", etc.

Thus, object tasks have a correspondence with object actions, and object actions correspond to low-level physical interaction tasks. The link between object actions and physical interaction tasks is given by the physical interaction planner. The correspondence between object tasks and object actions must be stored in the robot by rule based programming, or either learnt by demonstration, experimentation, etc.

Some object classes, like the *door* class, may not have object actions associated with them. Object parts categorized in such classes, can be used for describing high-level tasks, although the object action is associated with another part. For example, "open door" could be a high-level task description of the "pull handle" action.

5.4 Planning

There is a direct mapping between an object action and a physical interaction task. The physical interaction planner is in charge of this transformation. Taking as input an object representation and an action to perform on a given object part, the physical interaction planner defines the elements necessary for the specification of the grasp and the task.

5.4.1 The grasp

According to expression 2, the grasp can be specified with a grasp preshape, $\mathcal{P}$, a hand frame, H, a grasp frame, G, the desired hand-grasp relationship, ${}^{H}\mathbf{M}_{G}^{*}$, and a matrix which selects the constrained DOF's, $\mathbf{S}_c$.

The grasp preshape and the hand frame are jointly specified through the selection of a suitable task-oriented hand preshape. The most suitable task-oriented hand preshape is chosen according to the object action and the task geometry. Table 2 shows the correspondence between an object action and the corresponding task-oriented hand preshape, depending on three geometric properties of the object part and the task: if there is enough space for enveloping the object with the fingers, whether the task direction is known or unknown, and whether the geometry of the box primitive is classified as elongated, compact, large or small.

The grasp frame is selected according to the following rules (see also Figure 13):

- In the case of prehensile grasps (cylindrical, pinch, lateral), its origin is placed on the center of the box primitive. In the case of non-prehensile grasps (hook, one-finger), its origin is set on the centroid of a box face having its normal vector opposite to the task direction.
- The Z axis of the grasp frame must indicate the approaching direction. In the case of non-prehensile grasps, the approaching direction corresponds to the task direction.
- The rest of the axes are chosen so that the desired state of the grasp link can be specified with the identity matrix, i.e. ${}^{H}\mathbf{M}_{G}^{*} = \mathbf{I}_{4\times4}$.

The DOFs which are not constrained by the grasp depend on the selected preshape. In general, power preshapes (cylindrical power and hook power) do not constrain rotation around Y axis of the hand frame, whereas precision preshapes (cylindrical precision, hook precision, pinch and lateral) allow a rotation around X axis, with the exception of one-finger preshapes (one-finger frontal and one-finger precision), which allow rotation around Z axis.

5.4.2 The task

According to expression 3, the task specification is composed of the task frame, T, the velocity reference, $\mathbf{v}^*$, the force reference, $\mathbf{f}^*$, and the compliant selection matrix, $\mathbf{S}_f$.

The planner allows for two possibilities in the placement of the task frame. The first is to place it at the joint axis, as specified by the geometric compatibility requirement of the TFF. The other possibility is to set it at the same location as the grasp frame. The first case has the advantage that it is more coherent with the task geometry. The second case has the advantage of specification simplicity at the expense of control complexity. It is possible to configure the desired behavior of the planner depending on the available knowledge about the object, but, in general, we opt for the second case, because in some cases it could be desired to perform a door handle turning or pushing operation where only the handle has been recognized and no information is available about the door geometry and kinematics. Therefore, instead of relying on a motion model, we rely on robust sensor-based control in order to adapt the hand motion to the object kinematics.

Suitable velocity references are set in all the cases, with the exception of the *button push* action, where a force reference is set. The compliant selection matrix is defined accordingly.

6 Application to real robots

The framework and the planner proposed in this article have been developed incrementally, as the result of several applications in three different robotic systems: the UJI Service robot (Prats et al, 2007b) at the Robotic Intelligence Lab in Castellón (Spain), a mobile manipulator (Lee et al, 2007) at the Intelligent Systems Research Center in Sungkyunkwan University (South Korea), and the Armar-III Humanoid Robot (Asfour et al, 2006), in Karlsruhe University (Germany). The tasks involved:

- Turning a door handle with force feedback.
- Different strategies for pull opening several doors and drawers: using only force feedback, by vision-force control with markers, and by vision-force-tactile integration in a markerless environment.
- Two different strategies for book grasping, one using only force feedback, and another, integrating tactile and force information.

The variety of the tasks that have been implemented shows the versatility of the proposed approach, and its suitability for the fast implementation of robust physical interaction tasks in very different robotic systems. In this section, we present a survey of each task, focusing on their specification in terms of the framework for physical interaction. In addition, all of them are automatically planned with the physical interaction planner, with the exception of the book grasping task which is manually specified, but still supported by the framework. The sensor-based controllers are specific for

Object action	Gap	Direction	Size	Preshape
Door handle turn	Yes	Unknown		Cylindrical power
	Yes	Known		Hook power
	No	Unknown		Cylindrical precision
	No	Known		Hook precision
Fixed handle push	Yes	Unknown	Elongated	Cylindrical power
	Yes	Known	Elongated	Hook power
	No	Unknown	Elongated	Cylindrical precision
	No	Known (perpendicular)	Elongated	Cylindrical precision
	No	Known (parallel)	Elongated	Hook precision
	No		Compact	Lateral
Button push				One-finger frontal
Knob turn				Lateral
Slider move		Unknown		Lateral
		Known		One-finger precision
Liftable lift			Large	Cylindrical power
			Small	Cylindrical precision

Table 2 Task-oriented preshapes assigned to each object action depending on the task parameters and object geometry.

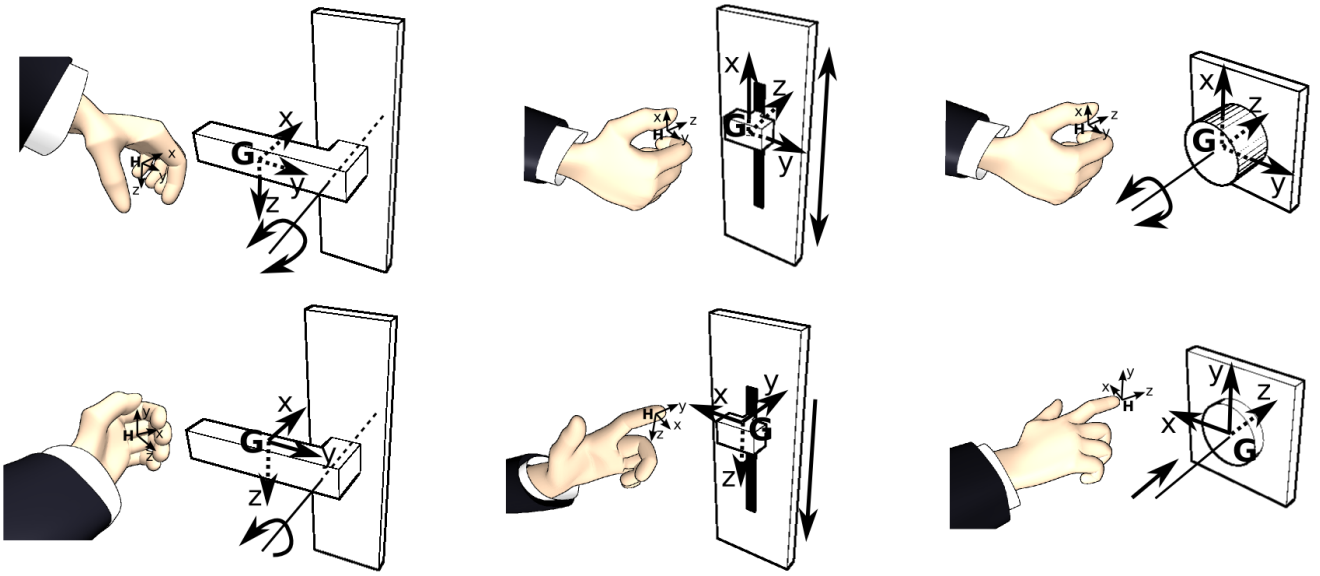


Fig. 13 Some examples of the grasp frame specification according to the task description and selected preshape. For power preshapes, the grasp frame is placed on the center of the box primitive, with Z axis indicating the approaching direction. For precision preshapes, it is set to the box face which normal is opposite to the task direction.

each example, but all of them are following the general guidelines given in sections 4.2.1 and 4.2.2. For low-level details about each task, the reader is referred to the corresponding specific publication.

6.1 Turning a door handle with the UJI Service Robot

In this experiment, the UJI Service robot is requested to turn a door handle, as shown in Figure 14. The physical interaction planning algorithm of Section 5 decides that the hook power preshape is the most suitable task-oriented posture for the given task and geometry, and automatically sets the grasp frame to the middle of the handle upper face, with Z direction pointing downwards.

A grasp controller following the approach presented in section 4.2.1 first moves the robot's hand to a point over the handle, and then reaches the handle through the approaching direction. When contact is detected, the task controller (section 4.2.2) performs the task by means of velocity/force references, while the grasp controller updates the hand orientation in order to keep a suitable contact configuration. The switching between the reaching phase and the interaction is implemented for this specific example as an automata in our software architecture.

Fig. 14 The UJI Service robot turning a door handle through velocity/force control.

This leads to a very robust behavior which is able to succeed in most of the times. The only problem comes from the particular resistance that the handle offers during turning. The force that must be applied to the handle cannot be automatically planned, since it is specific for a particular door. Therefore, we have to manually set a force reference which is suitable for the given handle. For this task, we set it to a value of 15N, which was shown to be suitable experimentally. As future work, we would like to address the problem of how these parameters can be automatically acquired through the previous experience. More details are given in (Prats et al, 2007a).

6.2 Pulling from fixed handles

6.2.1 Force control with the Armar-III humanoid robot

In this experiment, we focus on tasks that involve force-guided robust physical interaction of a humanoid robot with articulated furniture found in the kitchen, such as opening doors and drawers. The Armar-III humanoid robot (Asfour et al, 2006), built up by the Collaborative Research Center 588 in Karlsruhe, has been used for these experiments. A total 8 DOF are used in the Armar-III humanoid robot: 7 in the arm, and 1 in the hip (yaw). The humanoid robot makes use of two different kinds of redundancy for continuously adopting a comfortable grasp during task execution: joint and grasp redundancy.

In the kitchen environment where Armar-III is operating, all the handles have the same shape. The physical interaction planner selects the hook power task-oriented preshape, which is mapped to a power preshape by the hand adaptor for the Armar-III hand, as shown in Figure 15. The task and grasp frames are then automatically set to the handle, and the task velocity is set to a negative value along Z axis (the exact value is set manually). This avoids the use of the particular mechanism

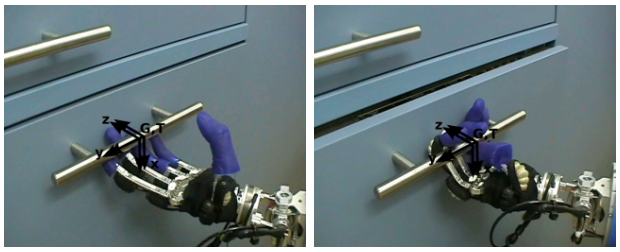


Fig. 15 The Armar-III humanoid robot pulling open the door of the dishwasher.

model, allowing the robot to use the same task description for the different doors and drawers, independently of the particular size or hinge position.

Here, position-force information is used in order to find the motion direction that minimizes the external forces. The robot starts by pulling the handle. The mechanism of the particular door, as well as the error in the estimation of the grasp link, generates small forces in the hand. The robot tries to minimize these forces by updating its position following an impedance force control approach. A history of the hand trajectory is stored and the task and grasp frames are aligned with the vector tangent to this trajectory, thus updating the hand-grasp link estimation. With this approach, the robot autonomously adapts to the particular door, without having any model, and even without knowing the particular mechanism. In fact, it was able to successfully deal with four different kitchen furniture while using the same control approach. Details are given in Prats et al (2008c).

6.2.2 Vision/force control with the ISRC mobile manipulator

In this example, the framework is applied to the task of pulling open the door of a wardrobe, combining vision and force in a mobile manipulator composed of an Amtec 7DOF ultra light weight robot arm mounted on an ActivMedia PowerBot mobile robot. The hand of the robot is a PowerCube parallel jaw gripper. This robot belongs to the Intelligent Systems Research Center (ISRC, Sungkyunkwan University, South Korea), and is already endowed with recognition and navigation capabilities (Lee et al, 2007), so that it is able to recognise the object to manipulate and to retrieve its structural model from a database.

In this case, the physical interaction planner selects a cylindrical precision preshape, which is mapped through the corresponding hand adaptor to the parallel jaw gripper. The automatically planned physical interaction frames are shown in Figure 16. The planner is configured to avoid the use of the door model,

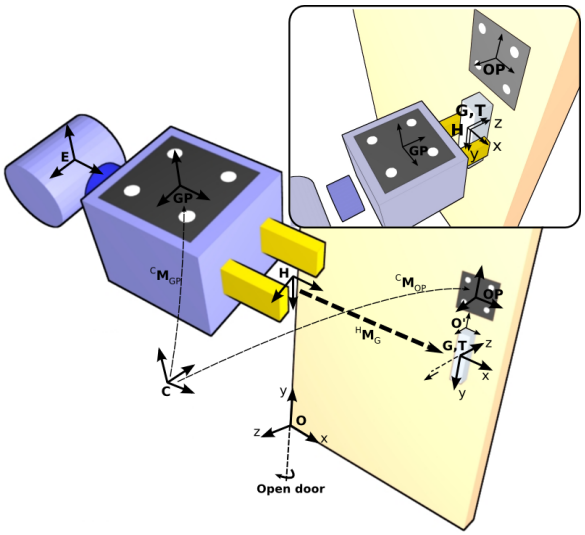


Fig. 16 The specification of the door opening task with the ISRC mobile manipulator.

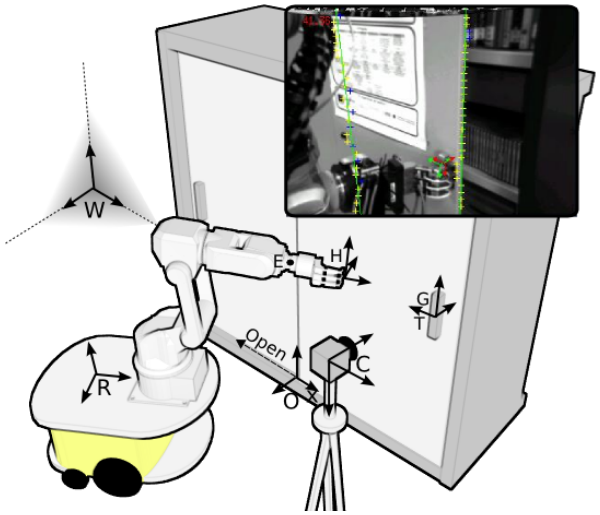


Fig. 17 The specification of a sliding door opening task with the UJI Service robot.

and, thus, it also sets the task frame to the handle, so that the interaction task is automatically specified as a negative velocity reference along Z axis. Again, the exact velocity magnitude is chosen manually to a suitable value.

Vision and force sensors are used in order to track the geometrically incompatible task frame. An external camera is used to track the object and the robot hand with the help of special markers. As all the measurements are relative to the hand and the object, camera motion can be performed without affecting to the overall behavior. The grasp link is then estimated directly with vision, and any misalignment is corrected through the grasp controller, which implements a position-based visual servoing control law. The grasp controller is in charge of the reach to grasp action, and of maintaining the grasp link desired state during interaction. In order to deal with unexpected forces, we adopt an external vision-force control law (Mezouar et al, 2007), where the current force vector is used to create a new vision reference. The robot succeeded in all of the attempts, as long as the visual features remained visible. In order to ensure this condition, the camera was manually moved during execution so that a suitable view was always available. Details are given in Prats et al (2008a), including additional experiments with a fridge door.

6.2.3 Vision-force-tactile control with the UJI Service robot

In this case, the task is to open a sliding cabinet door by combining vision, force and tactile sensors. The robot was manually moved in front of a cabinet as shown in Figure 17. The camera was placed in order to get a view of the cabinet door, and a coarse estimation of the homogeneous matrix describing the relationship between the camera frame and the robot base frame ($^{R}\mathbf{M}_{C}$) was calibrated. Note that this step would not be necessary in a humanoid system, for example, where the eye-to-hand relationship can be approximately computed through robot kinematics.

The physical interaction planner automatically selects a hook precision preshape for the Barrett Hand and the grasp frame is set accordingly. A model-based virtual visual servoing approach is used to track the door pose with respect to the camera, without using special markers. The door pose is then used for estimating the hand-grasp relationship. During reaching, this estimation is used by the grasp controller for making contact. During interaction, the grasp controller makes use of additional information coming from force and tactile sensors in order to keep a suitable grasp, while the task controller performs the opening motion.

In order to study the benefits that tactile sensors provide, the door opening task was reproduced several times, with three different sensor combinations and manually set rotational errors of up to 5 deg. in all of the axes of the initial estimation of the object pose. For each error (positive and negative), on each axis, the task was executed, first by only using the force sensor, then adding the vision modality, and finally by a combination of vision, force and tactile sensors. Therefore, a total of 18 trials were performed, 6 for force-alone, 6 for vision-force and 6 for vision-force-tactile. A trial was considered as a failure when the robot was unable to open the door over 25 cm. Simple force control succeeded in the 50% of the cases, whereas vision-force completed the task in 5 experiments, and vision-force-

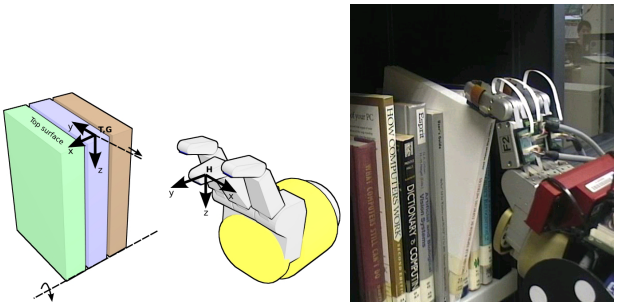


Fig. 18 Specification of the book grasping task.

tactile performed well in all the 6 cases. In addition, the vision-force-tactile combination was the only one able to avoid undesired forces in directions other than the task direction. More details can be found in Prats et al (2009).

6.3 Book grasping with the UJI Service robot

In this example, the compliant physical interaction framework is applied to the task of taking out a book from a bookshelf, using the UJI Service Robot. The approach is to do it as humans do: only one of the fingers is used, which is placed on the top surface of the target book and pulls it back, making the book turn with respect to the base, as shown in Figure 18. As this is a very specific action, it is not contemplated in the list of actions supported by the planner, and, thus, it was manually specified following the framework guidelines.

Two control strategies have been implemented: one using only force feedback, and another one using force-tactile integration (Prats et al, 2008b). In both cases, the force-torque sensor is used to apply a force towards the book and to avoid sliding. In the second case, a tactile array provides detailed information about the contact, and helps in estimating the hand-grasp relationship.

As shown in Figure 18, a one-finger precision preshape is adopted for making contact with the top surface of the book. The task frame is set to the same place that the grasp frame, so that the interaction task can be specified as a negative velocity reference in Y axis, and a force reference in Z. Simultaneously, a grasp controller is able to update the fingertip orientation in order to always keep a suitable contact with the book.

The typical failure in this case is when the books are too pressed in the shelf so that the fingers slide on the book surface without moving it. The solution is to increase the pressure that the finger makes on the book. However, this value is also set manually and may not be valid for situations in which the books are more pressed. Unfortunately, this variable is very difficult to measure and makes it impossible to perform a rigorous empirical study of the performance of this approach. In any case, automatic methods for the selection of control references should be developed.

7 Discussion

Currently, the number of interaction tasks that the planner supports is limited by the set of object classes that have been considered. It would be possible to add new object classes and to define object actions for them, or for the already existing ones. For example, the door class could allow a *knock* action. This action would require a new task-oriented hand preshape which is still not considered by the planner: a *fist preshape*. Most of the object parts found in our everyday environments can be classified into one of the proposed categories, but additional classes could be needed in other environments.

The planner makes use of object models built with box-shape primitives, although a complete model is not necessary in order to plan a physical interaction task. For planning an action on a handle, it would be enough to recognize the handle with a vision system, for example, and approximate its shape with a bounding box, without the need to know the mechanics or the geometry of the object to which it is attached. In the examples of this article, we have assumed that the object has already been successfully recognized and approximated with a bounding box.

It is worth noting that the task control does not necessarily rely on an object model. In our examples, the tasks have been executed without knowledge of the object mechanism. Instead, sensor feedback has been used in order to track the physical interaction frames and adapt the grasp and task motion to the particular case. In all the applications, the velocity and force references have been manually set to suitable values. As future work, we would like to address the problem of automatically setting the velocity and force references for a given task, based on previous experience.

Finally, the proposed framework and planner are based on the Cartesian control of the mobile manipulator end-effector. Thus, they are independent of the particular control strategy. We assume that a suitable controller, able to perform the desired Cartesian velocity exists. In the applications shown in this article, the Cartesian controllers have involved only the manipulator. The mobile platform has not been considered when executing the physical interaction tasks, although it would be possible to have another Cartesian controller taking it into consideration without affecting the specification of the task.

8 Conclusions

In this article, a framework for the integrated specification of the grasp and the task has been proposed, based on well-established techniques adopted from the task control and grasping communities: the task frame formalism, and the knowledge-based approach to grasping.

The framework is based on the suitable specification of the *physical interaction frames*. Among them, the hand frame and the grasp frame establish the link between the grasp and the task.

The execution of the physical interaction task is performed by two simultaneous controllers: the grasp and the task controller. The grasp controller approaches the hand towards a desired relative configuration with respect to the part of the object to be grasped, and tries to keep it during the task. The task controller transforms the task specification from the task frame to the end-effector frame, passing through the hand-grasp link.

The control framework assumes that an estimation of the hand-grasp relationship exists. This relationship must be estimated during execution in order to allow the transformation of the task specification, from object coordinates to robot coordinates. We propose to use robot perception for this estimation..

More complex tasks, such as using tools or two-handed manipulation, can be described as a set of individual physical interaction tasks and, thus, specified through different instantiations of this framework.

The second contribution of this paper is a planning algorithm that automatically specifies physical interaction tasks, taking as input an object description and the task to perform. The planner sets all the required elements that compose the definition of the grasp and the task on the basis of the physical interaction framework.

The new concepts of *task-oriented hand preshapes*, *ideal hand* and *hand adaptors* have been introduced as a way to provide a general task-oriented planning algorithm that can be instantiated on different robotic hands.

The framework and planner formalized in this article have been applied to several real situations in three different robotic systems, including two mobile manipulators and a humanoid robot. The variety of the tasks that have been implemented shows the versatility of the proposed framework, and its suitability for the fast implementation of robust physical interaction tasks in very different robotic systems. The juxtaposition of sensor-based grasp and task motion allows to robustly perform different tasks at the same time that the grasp state is constantly monitored and improved.

Acknowledgements We are very grateful to Profs. Sukhan Lee, Rüdiger Dillmann and Philippe Martinet, as well as Tamim Asfour and Steven Wieland, for their support in the experimental validation of this work. This research was partly supported by the Korea Science and Engineering Foundation under the WCU (World Class University) program funded by the Ministry of Education, Science and Technology, S. Korea (Grant No. R31-2008-000-10062-0), by the European Commission's Seventh Framework Programme FP7/2007-2013 under grant agreement 217077 (EYESHOTS project), by Ministerio de Ciencia e Innovación (DPI-2008-06636), by Fundació Caixa Castelló-Bancaixa (P1-1B2008-51) and by Universitat Jaume I.

References

Asfour T, Regenstein K, Azad P, Schroder J, Bierbaum A, Vahrenkamp N, Dillmann R (2006) Armar-iii: An integrated humanoid platform for sensory-motor control. In: IEEE-RAS International Conference on Humanoid Robots, pp 169–175

Baeten J, Bruyninckx H, De Schutter J (2003) Integrated vision/force robotic servoing in the task frame formalism. International Journal of Robotics Research 22(10-11):941–954

Bard C, Laugier C, Milési-Bellier C, Troccaz J, Triggs B, Vercelli G (1995) Achieving dextrous grasping by integrating planning and vision-based sensing. International Journal of Robotics Research 14(5):445–464

Bekey G, Liu H, Tomovic R, Karplus W (1993) Knowledge-based control of grasping in robot hands using heuristics from human motor skills. IEEE Transactions on Robotics and Automation 9(6):709–722, DOI 10.1109/70.265915

Bicchi A, Kumar V (2000) Robotic grasping and contact: A review. In: IEEE International Conference on Robotics and Automation, San Francisco, CA, pp 348–353

Borst C, Fischer M, Hirzinger G (2004) Grasp Planning: How to Choose a Suitable Task Wrench Space. In: IEEE International Conference on Robotics and Automation, New Orleans, USA, pp 319–325

Bruyninckx H, De Schutter J (1996) Specification of force-controlled actions in the 'task frame formalism': A synthesis. IEEE Transactions on Robotics and Automation 12(5):581–589

Cutkosky M, Wright P (1986) Modeling manufacturing grips and correlations with the design of robotic hands. In: IEEE International Conference on Robotics and Automation, San Francisco, CA, vol 3, pp 1533–1539

De Schutter J, De Laet T, Rutgeerts J, Decré W, Smits R, Aertbeliën E, Claes K, Bruyninckx H (2007) Constraint-based task specification and estimation for sensor-based robot systems in the presence of geo-

metric uncertainty. International Journal of Robotics Research 26(5):433–455

Graf B, Hans M, Schraft R (2004) Care-o-bot ii-development of a next generation robotic home assistant. Autonomous Robots 16(2):193–205

Haschke R, Steil J, Steuwer I, Ritter H (2005) Task-oriented quality measures for dextrous grasping. In: IEEE Conf. on Computational Intelligence in Robotics and Automation, Espoo, Finland, pp 689–694

Huebner K, Kragic D (2008) Selection of robot pre-grasps using box-based shape approximation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 1765–1770

Huebner K, Ruthotto S, Kragic D (2008) Minimum volume bounding box decomposition for shape approximation in robot grasping. In: IEEE International Conference on Robotics and Automation, pp 1628–1633

Kaneko K, Kanehiro F, Kajita S, Hirukawa H, Kawasaki T, Hirata M, Akachi K, Isozumi T (2004) Humanoid robot hrp-2. In: IEEE International Conference on Robotics and Automation, New Orleans, USA, vol 2, pp 1083–1090

Katz D, Brock O (2008) Manipulating articulated objects with interactive perception. In: IEEE International Conference on Robotics and Automation, Pasadena, USA, pp 272–277

Kemp CC, Anderson CD, Nguyen H, Trevor AJ, Xu Z (2008) A point-and-click interface for the real world: laser designation of objects for mobile manipulation. In: ACM/IEEE International Conference on Human Robot Interaction, New York, NY, USA, pp 241–248

Khatib O (1987) A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal of Robotics and Automation 3(1):43–53

Latombe JC (1991) Robot Motion Planning. Kluwer Academic Publishers

Lee S, Lee S, Lee J, Moon D, Kim E, Seo J (2007) Robust recognition and pose estimation of 3D objects based on evidence fusion in a sequence of images. In: IEEE International Conference on Robotics and Automation, Rome, Italy, pp 3773–3779

Li Z, Sastry S (1987) Task oriented optimal grasping by multifingered robot hands. In: IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, vol 4, pp 389–394

Lyons D (1985) A simple set of grasps for a dextrous hand. In: IEEE International Conference on Robotics and Automation, vol 2, pp 588–593

Mackenzie C, Iberall T (1994) The Grasping Hand. North-Holland

Marrone F, Raimondi F, Strobel M (2002) Compliant interaction of a domestic service robot with a human and the environment. In: 33rd Int. Symposium on Robotics, Stockholm, Sweden, pp 7–11

Mason M (1981) Compliance and force control for computer-controlled manipulators. IEEE Transactions on Systems, Man, and Cybernetics 11(6):418–432

Mezouar Y, Prats M, Martinet P (2007) External hybrid vision/force control. In: International Conference on Advanced Robotics, Jeju, Korea, pp 170–175

Miller AT, Allen PK (1999) Examples of 3D grasp quality computations. In: IEEE International Conference on Robotics and Automation, Detroit, Michigan, pp 1240–1246

Miller AT, Knoop S, Christensen HI, Allen PK (2003) Automatic grasp planning using shape primitives. In: IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp 1824–1829

Morales A, Asfour T, Azad P, Knoop S, Dillmann R (2006) Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, pp 5663–5668

Napier J (1956) The prehensile movements of the human hand. Journal of Bone and Joint Surgery 38-B(4):902–13

Nguyen H, Kemp CC (2008) Bio-inspired assistive robotics: Service dogs as a model for human-robot interaction and mobile manipulation. In: 2nd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics, pp 542–549, DOI 10.1109/BIOROB.2008.4762910

Niemeyer G, Slotine JJ (1997) A simple strategy for opening an unknown door. In: IEEE International Conference on Robotics and Automation, Albuquerque, NM, USA, vol 2, pp 1448–1453

Okamura AM, Smaby N, Cutkosky MR (2000) An overview of dexterous manipulation. In: IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, pp 255–262

Ott C, Buml B, Borst C, , Hirzinger G (2005) Employing cartesian impedance control for the opening of a door: A case study in mobile manipulation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems workshop on Mobile Manipulators: Basic Techniques, New Trends & Applications, Edmonton, Canada

Petersson L, Austin D, Kragic D (2000) High-level control of a mobile manipulator for door opening. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Takamatsu, Kagawa, Japan, pp

vol 3, pp 2333–2338

Petrovskaya A, Ng A (2007) Probabilistic mobile manipulation in dynamic environments with application to opening doors. In: Int. Joint Conf. on Artificial Intelligence, Hyderabad, India

Prats M, del Pobil AP, Sanz P (2007a) Task-oriented grasping using hand preshapes and task frames. In: IEEE International Conference on Robotics and Automation, Rome, Italy, pp 1794–1799

Prats M, Sanz PJ, del Pobil AP, Martínez E, Marín R (2007b) Towards multipurpose autonomous manipulation with the UJI service robot. ROBOTICA 25(2):245–256

Prats M, Martinet P, del Pobil AP, Lee S (2008a) Robotic execution of everyday tasks by means of external vision/force control. Intelligent Service Robotics 1(3):253–266

Prats M, Martinet P, Sanz PJ, Lee S (2008b) Compliant physical interaction based on external vision-force control and tactile-force combination. In: IEEE International Conference on Multisensor Fusion and Integration, Seoul, South Korea, pp 405–410

Prats M, Wieland S, Asfour T, del Pobil AP, Dillmann R (2008c) Compliant interaction in household environments by the armar-iii humanoid robot. In: IEEE-RAS International Conference on Humanoid Robots, Daejeon, South Korea, pp 475–480

Prats M, Sanz PJ, del Pobil AP (2009) Vision-tactile-force integration and robot physical interaction. In: IEEE International Conference on Robotics and Automation, Kobe, Japan, pending publication

Quigley M, Berger E, Ng AY (2007) Stair: Hardware and software architecture. AAAI 2007 Robotics Workshop

Raibert M, Craig J (1981) Hybrid position/force control of manipulators. ASME Journal of Dynamic Systems, Measurement and Control 102(2):126–133

Saxena A, Driemeyer J, Ng AY (Feb 2008) Robotic grasping of novel objects using vision. International Journal of Robotics Research 27(2):157–173

Schlesinger G (1919) Der mechanische Aufbau der kunstlichen Glieder in Ersatzglieder und Arbeitshilfen. Springer Berlin

Stansfield S (1991) Robotic grasping of unknown objects: A knowledge-based approach. International Journal of Robotics Research 10(4):314–326

Thomas U, Finkemeyer B, Kröger T, Wahl F (2003) Error-tolerant execution of complex robot tasks based on skill primitives. In: IEEE International Conference on Robotics and Automation, Taipei, Taiwan, vol 3, pp 3069–3075

Wren D, Fisher R (1995) Dextrous hand grasping strategies using preshapes and digit trajectories. In: IEEE International Conference on Systems, Man and Cybernetics, Vancouver, BC, Canada, vol 1, pp 910–915

Wyrobek K, Berger E, Van der Loos H, Salisbury J (2008) Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In: IEEE International Conference on Robotics and Automation, Pasadena, California, pp 2165–2170