Project Acronym:	GRASP
Project Type:	IP
Project Title:	Emergence of Cognitive Grasping through Introspection, Emulation and Surprise
Contract Number:	215821
Starting Date:	01-03-2008
Ending Date:	28-02-2012

Deliverable Number:	D5
Deliverable Title :	Grasping control architecture, robust grasping primitives and sensor uncertainty modeling
Type:	PU
Authors	V. Kyrki, J. Laaksonen, A. Morales and J. Felip
Contributing Partners	LUT, UJI

Contractual Date of Delivery to the EC:	28-02-2009
Actual Date of Delivery to the EC:	28-02-2009

Contents

Chapter 1

Executive summary

Deliverable D5 presents parts of developments within workpackage WP3 "Self-experience of Grasping and Multimodal Grounding" of GRASP for the first twelve months of the project.

WP3 is responsible for the instantiation of motion with the current configuration of a robot as well as symbol grounding linking the agent's sensor experience with symbols, concepts, and attributes during a manipulation attempt to allow learning. As such, it addresses the problems of how a robot i) applies uncertain world knowledge to optimise grasping on-line, ii) adapts the grasp experience using sensor feedback to cope with uncertainty and new situations, and iii) grounds the grasp experience to given symbolic grasp models.

The relationships of WP3 with other WPs can be described as follows: WP2 produces the basis for modelling of actions with primitives, thus generating the primitive sequences for WP3 while WP4 provides the position of the manipulation target. These allow WP3 to make a manipulation attempt. WP6 provides a prior guess of sensor image during a manipulation attempt. This allows WP3 to detect surprise events as incoherence between expected and measured readings. WP3 informs WP4 about the actual experience of the world such that the world description can be updated. In addition, WP3 grounds the information to the symbols used in WP4. WP3 informs WP5 that something surprising happened, together with the description of what did happen. This allows WP5 to learn from the surprise. WP7 integrates the subsystems from all other WPs, such that WP3 provides the on-line manipulation controller subsystem.

According to the Technical Annex of the project, D5 presents activities connected to Tasks 3.1, 3.2, and 3.3. The objectives of these tasks are defined as

- **[Task 3.1] - Control Architecture.** Initially, a hierarchical control architecture will be defined and developed such that it allows relating the concepts of the grasping ontology defined in WP2 to the immediate control. After the architecture has been defined, this task will continue with the definition and development of the general control architecture components, mainly a Cartesian controller and high-level supervisory and visual controllers.
- **[Task 3.2] - Multimodal Grounding.** The task aims for the definition and development of a grounding mechanism connecting action primitives and attributes with uncertain sensor information, including modelling of the uncertainties involved. Initially, the modelling of uncertainties of the three sensor types (visual, tactile, proprioceptive) is studied considering the context of the attributes of the grasping ontology. Later, the task will continue by studying the temporal grounding problem as a state estimation problem with uncertain information, as the concepts and therefore the symbol set are defined by the grasping ontology.
- **[Task 3.3] - Robust action primitives.** The task aims for the definition and evaluation of adaptive and robust control approaches for individual action primitives. The main focus will be on studying the possible grasp primitives for different hand kinematics (parallel jaw, three-fingered, five fingered) and to identify robust parameterisable primitives through evaluation. Parameterisation of the primitives allows self-experience to be used for improving the performance during future attempts.

Chapter 2 describes results of work done related to Task 3.1, "Control architecture". To understand the

decisions made on the design of the control architecture, existing manipulator control architectures are first reviewed in Sec. 2.1. During the twelve first months of the GRASP project, a control architecture has been designed. Section 2.2 describes both the design of the control architecture and the current progress of the actual implementation of the control architecture, and how the control architecture compares to the reviewed architectures from the point of view of the GRASP project to enable intelligent and adaptive robotic manipulation with hardware independency.

Chapter 3 describes results related to Task 3.3, "Robust action primitives". The chapter introduces the design of a grasping primitive that uses sensor-based feedback to deal robustly with uncertainty in the conditions of grasping in Sec. 3.3. In order to validate and test the developed primitive, a simple grasp planning system has been implemented, which principles are described in section 3.4. Early experimental results are also presented. The work is still at an early stage having started in September 2008.

Chapter 4 describes early work related to Task 3.2, "Multimodal grounding". This task will begin properly during the second year of the project, but the work has started with a literature review of sensor uncertainty modeling for grasping and manipulation. Section 4.1 outlines the role of uncertainty in control of manipulation, especially in grasping. Then, Sec. 4.2 reviews different types of models applicable for modeling sensor uncertainty. Finally, in Sec. 4.3, the phenomenon is examined from the point of view of different sensors applicable in grasping tasks.

Chapter 2

Control Architectures

Control architectures have well defined levels of hierarchy in literature [KS08]. The architectures range from planning several actions to low-level control. Planning resides at the highest level in the control architecture hierarchy and it is responsible for planning the actions, *e.g.* pick-and-place actions. The action level, or executive level, is one step below the planning level. The executive level is responsible for taking the action and divide it to smaller subactions, or motion primitives, which can be executed by logically simple low-level controllers. These low-level controllers are the most bottom level in the architecture hierarchy which is called the behavioral level.

The architecture discussed in Section 2.2, is focused on the two lower levels in the control architecture hierarchy, the executive and behavioral levels. The aim of the architecture is to provide means to input an action and split the action into suitable low-level controllers that can be executed to form the given action. Thus, the controller's goal as considered in this section is to perform a single motor action. In the system also other types of actions, such as perceptual actions, can exist as described in D4, but those do not relate to the on-line control and are not discussed here.

Sections 2.2.3 and 2.2.4 will present the current progress of the control architecture planned for use in the GRASP project. To understand the decisions made on the design of the control architecture, existing manipulator control architectures are reviewed in Section 2.1 to give an overview of the existing architectures in general.

The section 2.2 describes both the design of the control architecture and the current progress of the actual implementation of the control architecture, and how the control architecture compares to the reviewed architectures from the point of view of the GRASP project which is to provide intelligent and adaptive service robotics manipulator platform which is independent of the hardware.

2.1 Manipulator Control Architectures

As discussed above, control architectures can be divided into multiple levels, the same applies when the control architecture is used in conjunction with a manipulator, in this case a robotic arm and a hand. The planning level decides the sequence of actions, *e.g* picking up a set of objects, the executive level then divides the action into appropriate motion primitives, such as approach, grasp and transport, and the behavioral level actually implements these motion primitives. Figure 2.1 shows how the hierarchy relates to the concepts of task, action and primitive.

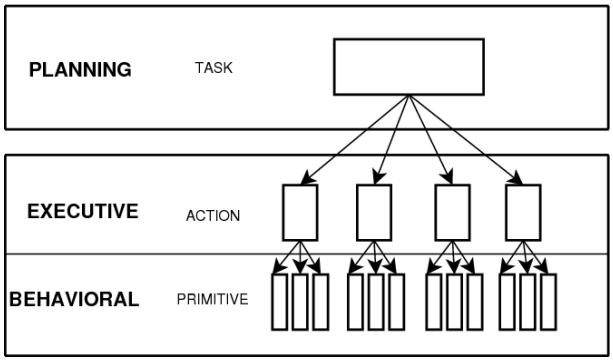


Figure 2.1: Hierarchy of control

Although a number of manipulator architectures have been presented previously, most of them do not focus on hardware independency, which is a one of the key factors in GRASP. However the concept of motion primitives is widespread and used successfully in many of the reviewed manipulation architectures.

Milighetti *et al.* [MKF+05] presented an architecture which uses the concept of primitive skills, that combine to form a skill, which in turn form a complete task. Each primitive skill is selected by heuristic selection out of many possible primitive skills, based on the sensor signals. A neural network is used to detect the change between the skills. A complete task can then be seen as a chain of primitive skills. Each primitive skill is based on a separate controller.

Haidacher *et al.* [HBF+03] demonstrated an architecture for the DLR Hand II. The architecture is based on different levels of complexity, which handle different aspects of the control. Again, the concept of hierarchical decomposition is central, but the architecture is limited to a single hand and the adaptiveness of the architecture has to be implemented at the highest level as the lower levels are statically defined.

Han *et al.* [HLT+00] present a control architecture for multi-fingered manipulation. As previously, the architecture is based on different levels that handle control from planning to actual joint control. The problem with the architecture is the lack of adaptation as the architecture shows that only predetermined architectural components, such as the low level controllers, are available to use. In addition, the architecture does not consider the robotic arm, only the hand.

Hybrid discrete-continuous control architectures for manipulation, such as [PEC99] and [SB99], differentiate the control phases according to the state of the manipulator. This is achieved by using discrete events to classify the manipulation configuration and using continuous states to control the dynamic behavior in different configurations. This type of architecture is suitable for both low-level control [SB99] and for complete control architecture [PEC99]. Petersson *et al.* [PEC99] demonstrate a control architecture (Mobile Manipulation Control Architecture, MMCA) for a mobile manipulator based on behaviors. The actual manipulator behavior is modelled as a sequence of configurable primitive actions. These primitives can be chained together using an hybrid automata to form an action. Another mobile manipulator architecture (Wheeled Mobile Manipulator, WMM) by Chang and Fu [CF06] is also based on a hybrid automata, which has discrete states and each state has its own continuous control. Problem with the WMM architecture is that the automata performing the action can not be changed, that is, the structure of the hybrid automata is static, although the states and transitions can be configured in a limited way. Compared to the MMCA, the WMM is limited in capability, as the MMCA is capable of supporting an arbitrary automata. Also MMCA is designed to be hardware agnostic, that is, MMCA can support many hardware platforms.

Aramaki *et al.* [ASK02] developed an architecture for humanoid robot. The architecture is based on

human action framework of conscious and unconsious tasks. In addition to these tasks, also a control task is defined as the lowest level of control. This architecture corresponds to the general definition of hierarchical control as described in Chapter 2. The conscious task is the planning level and the unconscious level is the executive level. The unconscious tasks are based on a state machine which controls the control tasks at the lowest level.

Prats *et al.* [PdPS06] present a control architecture based on the concept of task frame formalism [Mas81]. The architecture is divided into three parts: perceptions, actions and abilities. Perceptions are the sensors of the robot platform which produce information in the architecture. The actions are comparable to the behavioral level in the general control architecture hierarchy as they are the actual controllers controlling the hardware. Abilities form the planning and executive layer of the architecture. Abilities are formed as an automata, much like in [PEC99] and [SB99]. However, the automata can run multiple actions in parallel. Each node of the ability automata can be a primitive or another ability. This recursion can create complex actions from simpler actions which facilitates learning of abilities.

In general it seems that a control architecture based on some kind of a state machine or automata, is the most popular one for controlling the executive and behavioral level in the architecture hierarchy. The state machine structure allows to have well defined transitions and the states of the state machine can be separate from each other giving more freedom to define the state's continuous control. The structure provided by the state machine is ideal when considering the human manipulation, especially grasping, as it has been observed that human grasp is performed in multiple phases [Cas05], each with its own goal. Using the state machine, we can map the required phases to multiple states, each state having its own continuous control and transitions to next states.

2.2 Control architecture for GRASP

2.2.1 High Level Design of the Control Architecture

The high level control architecture design is based on the requirements set by the GRASP project. One of the most important requirements were hardware independency, as the architecture must work with minimal changes on multiple hardware or simulated platforms. Second important requirement was that the control should be based on a sequence of adaptive motion primitives, to mimic the human manipulation as described in Section 2.1. Also by sequencing the action, the action is easier to analyze and adapt for learning purposes. Comparing the requirements to the reviewed architectures, the most promising is the MMCA [PEC99], but it is focused on mobile manipulation and the architecture has not been used to demonstrate any manipulation tasks which makes the MMCA unsuitable as the control architecture for GRASP. For this reason, a new design for the control architecture was conceived for GRASP.

The whole control architecture is based on a high level design which illustrates the complete picture of the control architecture by defining the input and output interfaces and the inner logic of the architecture. By defining these elements it is possible to produce a concrete version of the control architecture in multiple ways using different software components. The high level design of the control architecture is presented in Figure 2.2.

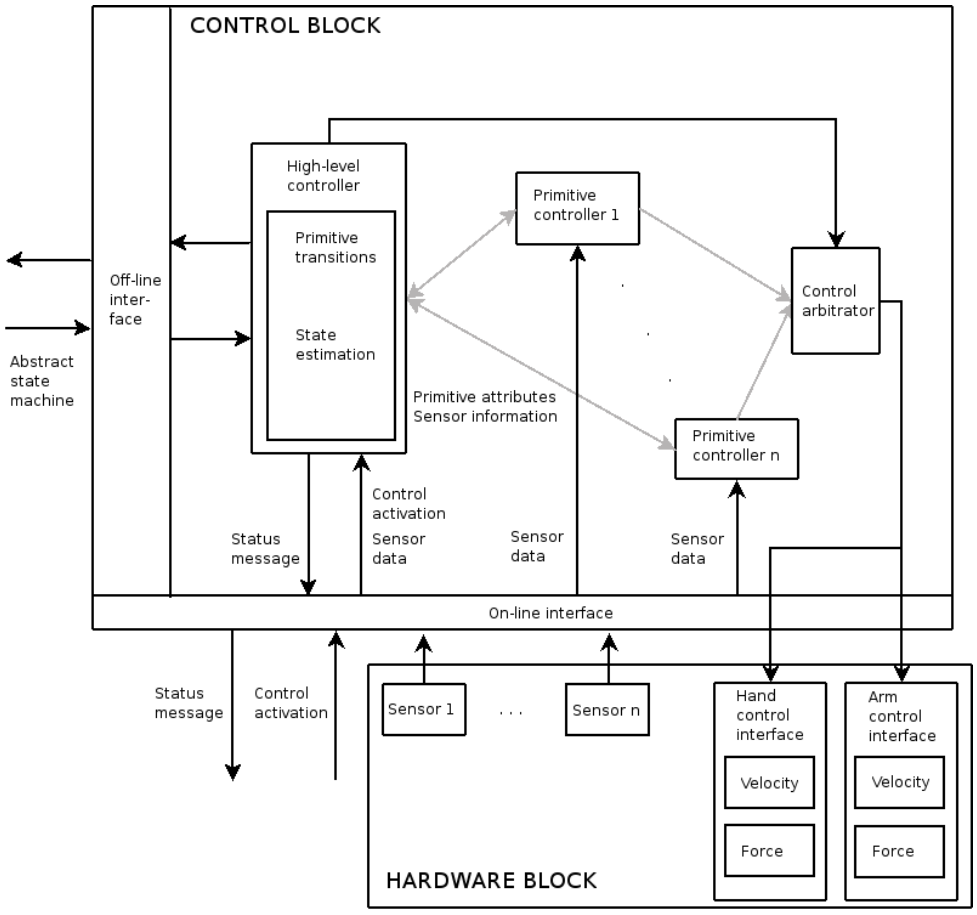


Figure 2.2: Control architecture design

The overall design has been divided into two blocks, the control block and the hardware block. The purpose of this division was to encapsulate the hardware dependent components away from the actual control to the hardware block, so that the control architecture can be implemented on multiple hardware or simulation platforms with minimal work. This encapsulation is the result of the hardware independency

requirement placed on the control architecture. The sensor and hardware control interfaces have been placed in the hardware block as they are completely dependent of the actual hardware.

The control block has two interfaces, the on-line and off-line interfaces. These interfaces are separated by their purpose. The off-line interface handles information which is not required to be processed in real-time. This information includes configuration information such as the abstract state machine, discussed in Section 2.2.2. Off-line information can also be transmitted to components outside the control block through this interface. The on-line interface provides real-time interface to the control process when the control is active. The hardware block communicates through this interface so that the sensor information can be accessed in real-time and that the control signals can be sent to the hardware at suitable intervals. The control process can be activated or deactivated through this interface and the on-line interface can be used to query the status of the control process from the high level controller.

The actual control block contains multiple components. The most important of these is the high level controller, which can be thought as the main component in the architecture. The high level controller is responsible for the internal state of the control process, that is, which primitive controllers are executed and at what time, and it also controls the control arbitrator. The high level controller can also process the information given from sensors through the on-line interface for state estimation for example.

Primitive controllers are actual controllers in the sense that they generate the control signals for the manipulator. The primitive controllers are somewhat dependent of the hardware, especially if some form of control method is required that needs information about the actual hardware capabilities. Multiple primitive controllers can be run at the same time.

Control arbitrator handles the output from the primitive controllers. The control arbitrator is used, if more than one primitive controller is running at the same time, to produce a single control signal for the hardware. The high level controller can give, for example, weights that adjust the relative strenght of the control signals when arbitrating them.

The primitive controllers and the control arbitrator are designed to be flexible so that the actual control scheme can be chosen freely. This design decision allows the control to adjust to different situations, and more importantly the different phases of manipulation can be implemented with different controllers.

2.2.2 Abstract and Concrete State Machine

The first step towards implementing the high level design of the control architecture, was to decide how the primitive controllers and the transitions between them could be represented. A hybrid discrete-continuous control architecture in the form of a state machine was decided on after reviewing the literature on control architectures. To adhere to the general requirements of the control architecture, the state machine must be divided into two parts, which are the abstract state machine and the concrete state machine. The abstract state machine is hardware independent and describes what motions are required to complete an action. The concerete state machine is the hardware dependent or embodiment specific version of the abstract state machine and the concrete state machine includes the actual controller or controllers that will produce the desired motions.

The abstract state machine is defined through XML (eXtensible Markup Language). XML provided a standard way to define a hierarchy, which in this case represents a state machine. Also, by choosing XML, a number of tools were available to read and write the state machine representation.

The structure of the abstract state machine is shown in Table 2.1. The state machine is divided into a hierarchy with three levels. The first level is the state machine level which contains the object, environment, state and transition definitions. The second level contains the individual properties for each of the definitions. The third level is the definition of the trajectory, or via points, for each state. The hierarchy is divided into columns in the table, for example the state machine definition encompasses all other definitions. Table 2.1 also presents all the possible properties in the abstract state machine for object, environment, state and transition definitions. A short description for each of these properties is written in italic font. An example of the abstract state machine is presented further in the section.

In addition to the individial properties, transition and state have their own attributes. The state definition has *name* and *type* attributes. The type attribute can have the following values:

- **success:** The success end state of the state machine.

Table 2.1: Abstract state machine structure

statemachine			
object	**environment**	**state**	**transition**
pose	obstacle	movement	success
object pose	*3D position*	*free or guarded motion*	*controller target reached*
weight		**trajectory**	grasp_lost
object weight		pose	*grasp is lost*
inertia		*via point pose*	grasp_stable
inertia matrix		point	*grasp is stable*
shape		*via point*	finger_contact
3D model			*a finger contact*
friction		hand_shape	finger_contact_lost
friction coefficient		*shape of the hand*	*a finger contact is lost*
class			timeout
class of the object			*a defined time limit*
			collision
			collision with environment
			hardware_failure
			a failure in hardware

- **failure:** The failure end state of the state machine.
- **move:** Moving the manipulator without an object.
- **transport:** Moving the manipulator with an object.
- **grasp:** Grasp the object.
- **release:** Release the object.

The type attribute will have a great impact on the concrete state machine as the type will define what kind of primitive controllers will be used at each state. The transition has also two attributes, *origin* and *destination*. These attributes inform where the transition is placed in the state machine, by defining the names of the origin and destination states. An example of a simple abstract state machine with just three states:

```
<statemachine>
  <object>
    <pose>1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1</pose>
    <weight>0.5</weight>
    <inertia>0 1 0 1 0 0 0 0 1</inertia>
    <shape>cup.mdl</shape>
    <friction>0.2</friction>
    <class>cylinder</class>
  </object>
  <state name="moving" type="move">
    <movement>free</movement>
    <hand_shape>open</hand_shape>
    <trajectory>
      <position>0 0 4</position>
      <position>0 5 3</position>
    </trajectory>
  </state>
  <state name="success_end" type="success">
  </state>
  <state name="failure_end" type="failure">
  </state>
```

```
  <transition origin="moving" destination="success_end">
    <success/>
  </transition>
  <transition origin="moving" destination="failure_end">
    <hardware_failure/>
  </transition>
  <transition origin="moving" destination="failure_end">
    <timeout>10</timeout>
  </transition>
</statemachine>
```

The example XML state machine describes a simple movement by the manipulator. As can be seen from the example, there can be multiple transitions between two states, and each transition can hold one or more transition conditions. Noteworthy mention is that the *hand_shape* property accepts only predetermined values, which are currently:

- **open:** Fully open the manipulator's hand.
- **closed:** Fully close the manipulator's hand.
- **extend:** Extends all finger of the manipulator's hand.
- **extend_finger:** Extends one finger of the manipulator's hand.
- **power_grasp_preshape:** Preshape for power grasp.
- **power_grasp:** Power grasp.
- **pinch_grasp_preshape:** Preshape for pinch grasp.
- **pinch_grasp:** Pinch grasp.
- **cylindrical_grasp_preshape:** Preshape for cylindrical grasp.
- **cylindrical_grasp:** Cylindrical grasp.
- **spherical_grasp_preshape:** Spherical grasp preshape.
- **spherical_grasp:** Spherical grasp.
- **hook_grasp_preshape:** Hook grasp preshape.
- **hook_grasp:** Hook grasp.

These values are used to determine the hand shape during each state. However, some hardware can only implement a subset of the hand shapes, for example, a parallel jaw has a very limited set of grasps. Chapter 3 describes the grasp types in detail.

The abstract state machine is able to represent most manipulation tasks, most importantly grasping, without any knowledge of the underlaying hardware which is the main objective of the abstract state machine. Because of this, the abstract state machine can not be used directly in controlling the hardware. Instead of the abstract state machine, a concrete or embodiment specific state machine must be used. The concrete state machine describes the action for a specific hardware platform completely and "perfectly", in the sense that concrete state machine can be used from the beginning of an action to the end of the action to control the manipulator at all times.

The concrete state machine must conform to the abstract state machine, so that the structure of the state machine stays unchanged. However, the abstract definitions, *e.g.* "grasp_stable", have to be translated for each hardware platform. This translation process is one of the key factors in providing the hardware independency of the control architecture.

The translation process will create a concrete state machine, which can be executed in target platform. The translation will produce the primitive controllers, as discussed in Section 2.2.1, and the transitions between the states that hold the primitive controllers. A translation of the example XML abstract state

machine to a concrete state machine is shown in Figure 2.3. The circles represent the states, and the arrows represent the transitions. As can be seen from the figure, the translation can add new transition conditions, for example, the hardware failure is translated to failure in either the manipulator's hand or the arm. Also the primitive controllers have been defined as joint controllers for the arm and the hand, as the example XML state machine did not constrain the trajectory.

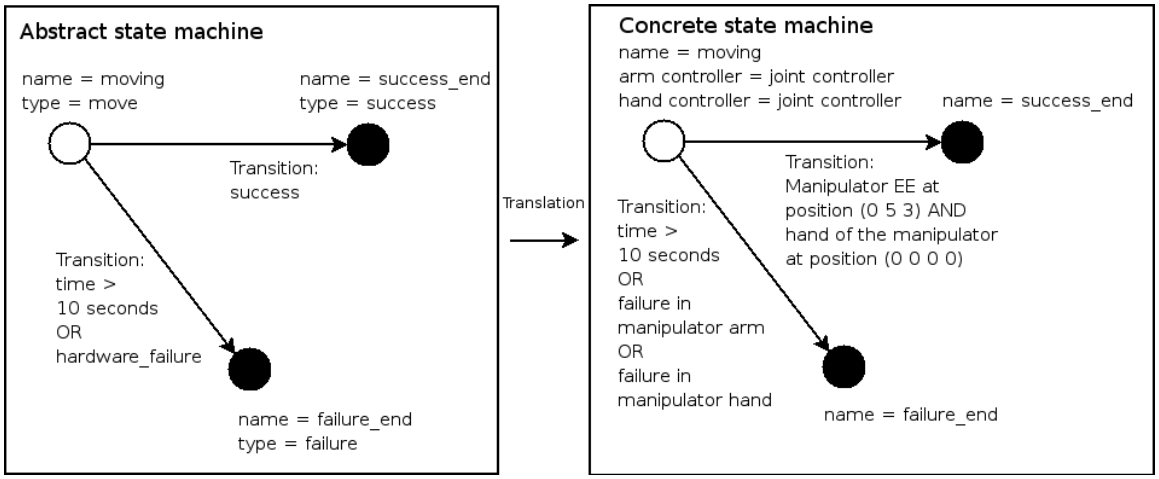


Figure 2.3: Translation of the example abstract state machine

2.2.3 OpenRAVE and Hardware Integration

OpenRAVE [DK08] is a planning architecture for autonomous robots. OpenRAVE provides simulator functionality and a possibility to use hardware components through the OpenRAVE system. OpenRAVE is a plugin based architecture which means that adding more functionality, such as new sensors, to the underlaying architecture is possible. OpenRAVE is also designed to be accessed by scripting languages, which can be used to read the states of the components or to modify the components during execution.

The control architecture is fully integrated to the OpenRAVE architecture. Components of the OpenRAVE architecture relevant to the control architecture are shown in Figure 2.4 as an unified modeling language (UML) [BRJ98] class diagram. Currently the control architecture relies on two OpenRAVE components, the controller plugin and sensor plugin, these are shown as SensorBase and ControllerBase classes in the class diagram. The controller plugin contains the whole control architecture implementation. The controller plugin can access the sensor plugins which are used to represent the hardware sensors. HighLevelController class represents the implemented controller architecture, which is discussed in Section 2.2.4. The Server class is used to gain access to the various components in the architecture and this access will be used as the off-line interface described in the high level design.

According to the high level design, real-time access to sensors and the manipulator hardware must exist. While the OpenRAVE sensor plugin can be used for the sensor hardware, OpenRAVE has no separate plugin for controlling hardware manipulators. Instead, the controller plugin must implement the connection to manipulator. For this purpose, a C++-interface was defined, which can be implemented for each hardware or simulated manipulator. The interface has a default method, velocity control, for controlling the manipulator arm. In addition to the velocity control, custom control methods, such as force control, can be implemented for both the arm and the hand of the manipulator. The interface also allows to activate or disable the hardware control. It is also possible to define separate arm and hand in the interface.

2.2.4 Implementation of the Control Architecture

The control architecture is implemented using C++-language which is common to all hardware platforms used in the project. Also, the OpenRAVE architecture is implemented in C++-language. The implemented architecture is shown in Figure 2.5 as an UML class diagram, showing only the relevant classes and the information that is communicated between the objects.

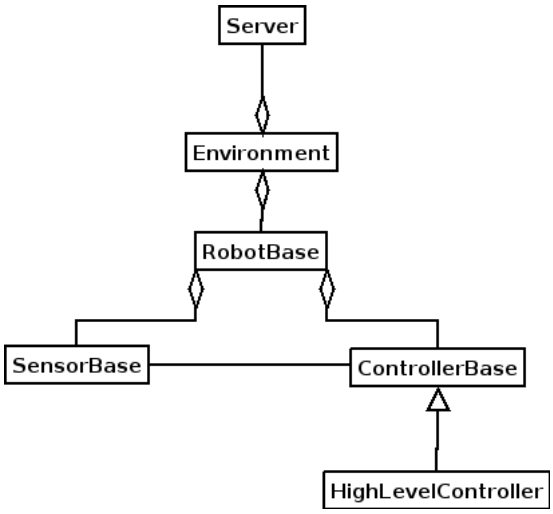


Figure 2.4: OpenRAVE structure

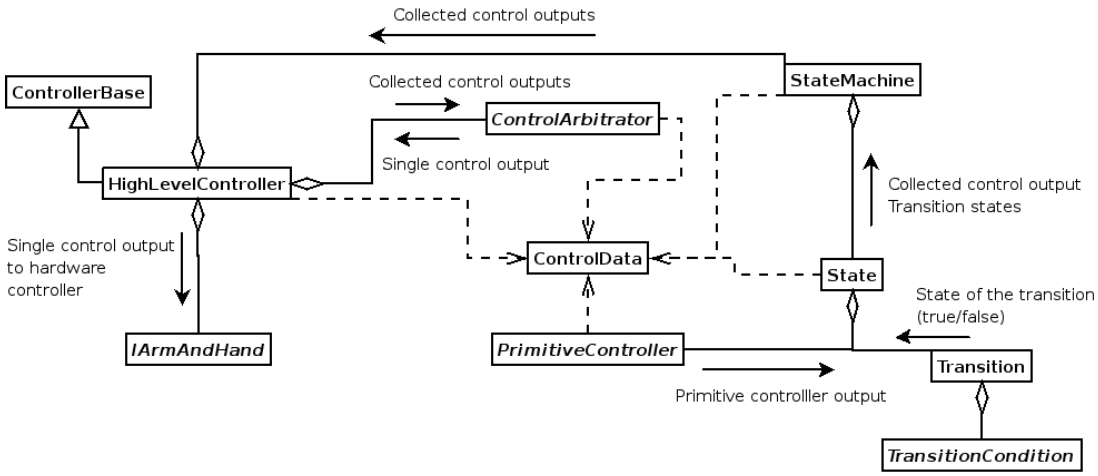


Figure 2.5: The design of the implemented controller architecture

Figure 2.5 contains the main classes of the control architecture and shows how the classes are related. As discussed in Section 2.2.3, the control architecture is integrated into OpenRAVE. The main class in the system is the HighLevelController class which implements the high level controller from the high level design. The HighLevelController class acts as a mediator between the classes to communicate the control output from the primitive controllers to the control arbitrator and from the arbitrator to the hardware interface, shown in Figure 2.5 as IArmAndHand class.

As the high level controller also had the responsibility to keep track of the state of the control process, the HighLevelController has the StateMachine class, which realizes the concrete state machine described in Section 2.2.2. The StateMachine class implements a similar hybrid automata with discrete transitions and continuous control as in some of the reviewed architectures [CF06] [PEC99].

Each state of the state machine implements the continuous control through the PrimitiveController class assigned to the State class. The PrimitiveController class implements the primitive controller of the high level design. The PrimitiveController class is an interface class which can be implemented to provide desired control method. Each state can have more than one primitive controller.

The transitions between states are handled by the Transition and TransitionCondition classes. The state can have more than one transitions, and each transition can can contain multiple transition conditions. The transitions work in the same manner as in the abstract state machine, informing the state if the transition is true or false depending on the transition conditions. TransitionCondition class is also an interface class, leaving the functionality free for implementation.

Each primitive controller outputs its result in the form of ControlData object. The ControlData class has information whether the controller controls the arm or the hand of the manipulator, which joints or degrees of freedoms (DOF) the controller controls, and which type of control is applied to each DOF, and finally the actual control output to the hardware for each DOF. This kind of structure allows, for example, single controller to control one DOF of the manipulator using some control method, while another controller can control the rest of the DOFs using some other control method. Each controller's output is given to the control arbitrator. Control arbitrator's task is to take all of the control data and give out one ControlData-object which can be communicated to the manipulator hardware interface.

One of the key elements of the architecture is also the possibility to use parameters or attributes for the primitive controllers and transition conditions which is not considered in many of the architectures reviewed in Section 2.1. The interface for both the primitive controllers and transitions conditions include a default way of modifying the parameters. This functionality can be used, for example, to adapt the primitives during the manipulation tasks. It is also possible to add new states to the state machine during the execution of the state machine which can be useful if a probabilistic approach, such as Markov chains, is used.

To investigate the design, a proof-of-concept version was first implemented in Matlab and interfaced with the GraspIt! grasping simulator. This version has been used in WP2 to do Bayesian reasoning as descibed in Deliverable 4.

2.2.5 Future Work

Future work on the control architecture will be focused on the abstract state machine and how to translate the state machine into concrete state machine. Currently, this is not possible to do automatically, although it possible to create the concrete state machine manually. Also a set of primitive controllers and transition conditions will be developed so that manipulation tasks will be possible to perform.

The problem of translating abstract definitions to a set of hardware specific entities, such as controllers and transitions, is not well defined in literature. Petersson *et al.* [PEC99] specifically mentions that the developed mobile manipulation control architecture is able to function in multiple hardware platforms, but the process of translating the given action abstraction to multiple platforms is not described nor demonstrated.

Chapter 3

Robust grasping primitives

3.1 Introduction

The architecture presented in previous chapter implements the two lower levels, executive and behavioral, of the hierarchy of levels (see Sec. 2.1 and Fig. 2.1). Their practical components are actions and primitives, respectively. In practical terms a primitive is composed of a single controller that realizes a indivisible and specific movement or interaction. In the upper executive level, actions represent plans that combine several primitives in order to realize more complex actions. In our case these plans are represented through finite state machines, and are composed of several primitives, transitions and conditions. The proposed architecture provides a framework to synchronise a pool of control primitives and to generalise their implementation with no regard of the underlying hardware setup.

In this chapter we focus in the design of a set of such controller primitives. In particular we focus on the realm of grasp execution. Grasping occurs when a robot hand makes a first contact with an object and immobilises it securely with the purpose of lifting and transporting it (or any other purpose). Grasping primitives are only a part of the primitives necessary to complete even the simplest manipulation action. On a complete system approaching, preshaping, lifting, transporting, and releasing primitives would be necessary to perform a pick-and-place action.

A successful grasp allows a secure completion of the whole manipulation action. This success can be achieved relatively easily if all the constraints of the grasping task are known in advance, however if there exist uncertainties in these constraints the problem is much more difficult.

This chapter describes a set of grasping primitives that uses sensor-based feedback to deal robustly with uncertainty in the conditions of grasping.

3.1.1 Grasping in the presence of uncertainty

Traditionally, most works on robot grasp planning, analysis and control have assumed to know in advance the layout of the workspace, to have a model of the object to manipulate and of the robot hand. In these conditions the problem of grasping becomes an analytical planning problem, and many theoretical and computational solutions have been proposed for the different stages of grasping [BK00].

The above assumptions are reasonable in industrial and controlled environments. However, in unstructured scenarios this is not the case, and often, theoretical solutions are not directly applicable. The main sources of uncertainty come from the attempt of manipulating objects for which models are previously unknown, or whose pose and physical characteristics are variable and not known in advance.

A common approach to overcome these difficulties has been the use of sensors to acquire information about the environment and hence reduce uncertainty. Within the field of grasp planning and execution, the use of sensors focus on three main stages; first, on object model acquisition, allowing a most traditional grasp planning after a model is built; second, on the approaching phase; and third, on the control loop of the grasp execution phase, with the purpose of obtaining a stable grasp.

Vision has been the most widely used modality in the two first stages. Many approaches to reconstruct

the shape of the object from visual input have been developed [JMLH05, WJL+05]. The reconstruction approach has been used successful with certain limitations with 2D objects [DB98, MSdF06], but no completely satisfactory solutions have been provided for the 3D case. Vision is often employed also when approaching the target objects, by using techniques of visual servoing and active vision[CH08].

However, when the moment comes to touch the object, vision leaves its leading role, and other type of sensory modalities are applied, mostly contact based sensors. They are employed mainly as feedback for the grasp control execution loop [PFG02] and in object exploration strategies [TM00], and both of these are often bundled together. From these works an interesting conclusion can be drawn: robust and stable grasps can be obtained even though a detailed model of the object is not available.

Another important problem with traditional grasp planning approaches is that grasps are described as sets of contact points on the object surface where forces/torques are exerted. But, as stated above, most of these works assume to have a perfect model of the object or to be able to obtain them from sensor data (e.g. vision, 3D laser scans). The difficulty arises from the fact that the sensor-based reconstruction of the object can be dramatically different from the real shape. Under these conditions it does not make sense to talk about pre-computed accurate contact locations any more.

An additional drawback of analytical approaches is that often the geometry and the kinematic constraints of the robot hand are not taken into account. As a result many general solutions are not applicable because they do not fit the particular characteristics of the robot hand.

The two former problems considered above lead us to conclude that grasp planning based on a set of contact points is not appropriate for manipulation tasks with a certain degree of uncertainty.

The rest of the chapter introduces the design of a grasping primitive able to robustly grasp objects whose shape and location is unknown in advance. No exact shape models of objects are assumed, but only a "common sense" knowledge of object classes is applied. The chapter focuses on those parts related with the goals of GRASP, that is, the characterisation and definition of grasping primitives (Section 3.2), and the implementation of particular primitive (Section 3.3.3). In order to validate and test the developed primitive, a simple grasp planning system has been implemented. The principles of the grasp planning are described in section 3.3. In addition, a recently submitted paper extending the approach presented in this chapter is included as Appendix A.

3.2 Grasping Primitives

Here we propose an approach, that describes grasps in a qualitative and knowledge-based fashion. Grasps are represented as primitives that define the control strategy and the sensory feedback to use in the execution.

A grasping primitive determines several key aspects of the grasp execution. First, it defines the hand preshape, that is, the posture of the hand when approaching the target. Second, it describes the control strategy to be used for executing the primitive. This also includes which sensor information is used an how it is interpreted. It also determines the metrics that evaluate the degree of accomplishment of the primitive.

Our definition of a grasp primitive presents a two sided perspective. From a practical point of view a grasp primitive is a single controller that performs a specific task on a particular embodiment. From an abstract point of view primitives are the simplest pieces of a vocabulary to elaborate plans or "actions" (see Sec. 2.1). Hence, they are well suited to be the basic pieces of a reasoning and learning procedure.

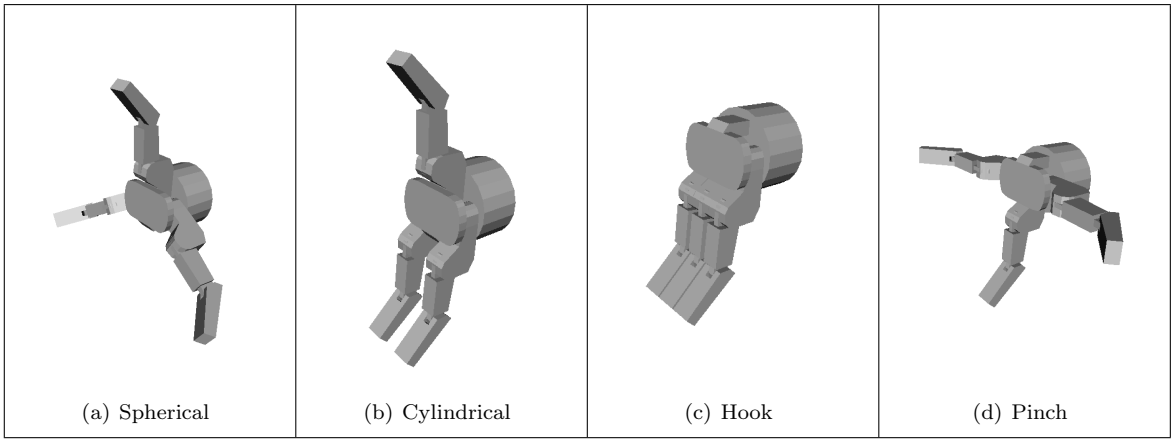
Here, we must refer to the hardware-independence issue. A grasp primitive, and in general, any controller primitive is likely to be hardware dependent. On the other side actions and state machines defined in Section 2.2 are hardware independent. Then, the controller primitives are the frontier or interface between specific hardware and general levels. For this reason, Sec. 2.2 defines both abstract primitives, which are hardware independent, and concrete primitives, which depend on the embodiment. This chapter focuses only on concrete, hardware dependent primitives.

The set of grasping primitives to be developed depends on the capabilities and particular features of the robot hand, and the different tasks (pulling objects, opening/closing doors, etc) to be performed by this hand. In any case a detailed study of the hand constraints, objects, and tasks is necessary. An example of a study for an anthropomorphic five-fingered hand is presented in [MAA+06].

In the case of the Barrett Hand, Miller et al. [MKCA03] presented a study on the possible hand preshapes that can be obtained with it. We use them to illustrate a subset of hand primitives for the Barrett Hand (see Fig.: 3.1). Our taxonomy of concrete grasping primitives for the Barrett Hand is the following:

- **Cylindrical grasp:** All fingers close around a cylindrical object. The thumb finger opposes the other two.
- **Spherical grasp:** All fingers close around a ball-shaped object.
- **Pinch grasp:** The grasp is characterised by the opposition of the two mobile fingers The thumb does not participate. This is appropriate to grasp small objects.
- **Hook grasp:** In this grasp the hand opposes the gravity. All fingers, form a hook that would enclose a cylindrical shaped object. The palm might exert force opposing the fingers.

In the first stage of the work we have only fully implemented the cylindrical preshape primitive.



(a) Spherical (b) Cylindrical (c) Hook (d) Pinch

Figure 3.1: Barrett Hand preshapes for the different grasp types.

3.2.1 Primitive parameters

In order to execute a grasp primitive in a particular scenario, several parameters must be determined (see Fig. 3.2):

- **Distance and approaching direction:** once the hand is positioned in the vicinity of the object it reaches toward it following the estimated direction for the necessary distance. The **approaching line** is the path followed by the robot hand when it approaches the object.
- **Hand orientation:** the hand can rotate around the approaching direction. The rotation angle is a relevant parameter to define the primitive initial configuration.
- **Object size:** the estimated size and proportion of the object affects the practical execution of the primitive, and is included as an input parameter to the execution controllers.
- **Force limits:** Depending on the estimated weight and solidness of the object, maximum and minimum force limits can be established for the controller.

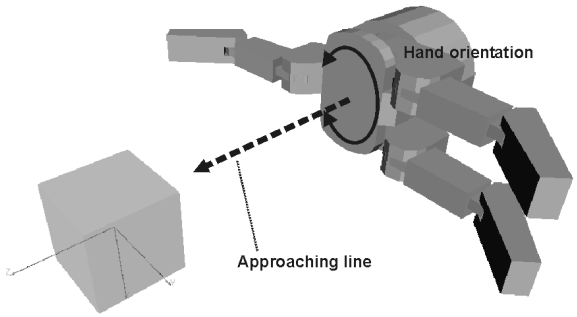


Figure 3.2: Schema with some components of a manipulation primitive

The generation of the values of these parameters and, in a higher level, the election of a particular grasp primitive and the generation of plans depends on specific planning modules, which principles and design are out of the scope of this report. A simple example of how this can be addressed is presented in section 3.4.

3.3 Implementation of a grasp primitive: the cylindrical case

The implementation of a primitive consists in the design of the control and execution strategies to be applied for the execution using the particular parameters of the primitive. In this section we describe the specific strategies applied for the implementation of a particular case: the cylindrical grasp.

But before describing the primitive algorithms it is important to describe the hardware that this primitives controls. It was stated before that grasp primitives are dependent on the structure and characteristics of the robot hand, and on the sensors available.

The robot hand is a three-fingered Barrett Hand and a JR3 force/torque and acceleration sensor mounted at the wrist, between the hand and the end-effector. The hand (Fig 3.4, on the right) has been improved by adding arrays of pressure sensors of the fingertips, which initiates roughly the tactile sensing. More details of these system are given later in Section 3.4.1.

3.3.1 Stages of the execution

In the first stage of the execution, the robot arm reaches toward the object and moves down until the fingertips are at level with the estimated object centroid. The hand is set in the preshape configuration and orientation. The second stage is the hand closing, the tactile sensors are used to determine when the fingertips first contact the object. Once this happens, the finger movement stops. This second stage finishes when all fingertips have contacted the object. In case that a finger misses the object, it stops as a security measure when it reaches a previously defined extension threshold. During the third and last stage the grasp is assessed to verify whether it is stable to lift the object. A grasp stability criterion is used to measure this aspect. If the grasp is not stable, correction movements are performed until a stable grasp is produced. The procedure described for these two last stages is an example of what we called *reactive controller*.

3.3.2 Grasp stability criterion

For assessing the reliability of a grasp in 2D, we previously defined several stability criteria [CMFP05]. For the purpose of this work we selected and adapted to the 3D case the *finger extension criterion*. The finger extension is defined as the distance of the contact fingertip to the center of the hand. In Fig. 3.3 the finger extensions of the two parallel fingers e_1 and e_2 and that of the thumb e_3 are depicted. Following the assumption that fingers with the same extensions are supposed to act more uniformly on the object, finger extensions are compared. The quality value Q according to this criterion is computed as the sum of the square differences between the three finger extensions:

$$Q = (e_1 - e_2)^2 + (e_2 - e_3)^2 + (e_3 - e_1)^2 \tag{3.1}$$

The criterion is set so that lower values are assigned to more stable grasps. In the situation where all three extensions are equal the best value (i.e. 0) is achieved. The worst value is obtained when some finger are at their maximum or minimum possible extensions. The experiments we performed showed that a grasp can be considered as stable in our setup when $Q < 0.01$.

3.3.3 Grasp execution

When the fingers close on the object, they stop at the moment of contacting the object surface. At that point, the above criterion is computed and the executed grasp assessed. A divergence between the estimated and the real values of distance, pose and size of the object can result in an unstable planned grasp. In these cases, the finger extension criterion will provide a high value, characteristic of an unstable configuration (see Fig. 3.6). A proper response for adapting the hand pose to the new situation has to be carried out, through a number of suitable correction movements. The implemented movements are:

1. **Orientation correction:** it is performed when the slant of the object is different from expected. This situation is identified using the extensions of the two parallel fingers e_1 and e_2 and the inter-finger distance i:

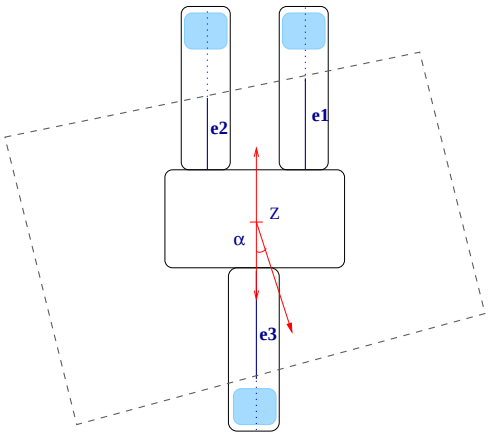


Figure 3.3: Graphical interpretation of the finger extensions, e_1, e_2, and e_3, and correction movements (α and Z).

$$\alpha = arctan(e_2 - e_1)/i \tag{3.2}$$

The computed correction angle α will bring the hand in the situation in which the parallel fingers have the same extension, and the hand is rotated accordingly (see fig. 3.3). The threshold we chose for executing the correction movement is $\alpha = 0.01rad$, and smaller deviations are not considered.

2. **Position correction:** this is performed when the position of the object is different from the estimation. In this case, either the thumb or the parallel fingers will contact the object much earlier, and thus show a much smaller extension. When one of the finger extensions is smaller than an assumed threshold value (54mm in our case), the correction is calculated and the arm is moved accordingly.

   The required displacement for position correction is computed with the following expression, which uses again the notation of Fig. 3.3.:

$$Z = \frac{1}{2}(\frac{e_1 + e_2}{2} - e_3) \tag{3.3}$$

After the correction movements have been applied, grasp stability is checked again. If the grasp is still considered unstable the correction process is repeated, otherwise the grasp is executed and the object lifted.

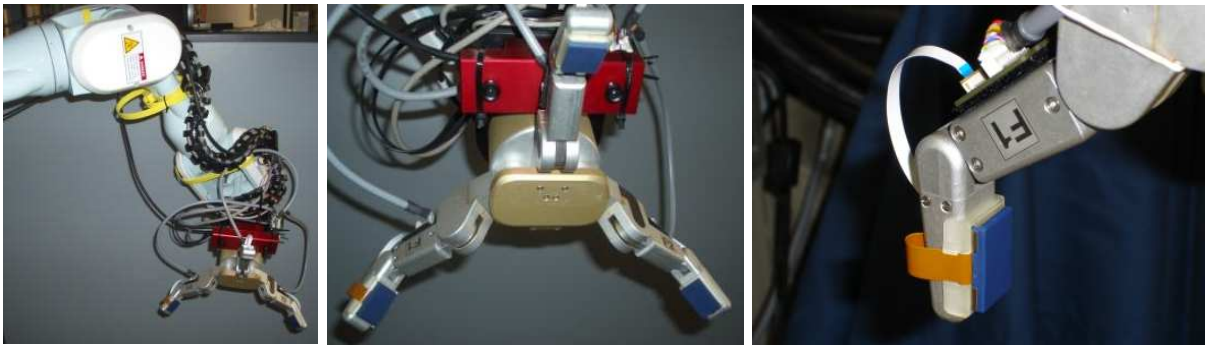


Figure 3.4: From left to right: robot arm, hand and stereoscopic camera; and a detail of the tactile sensors on the fingers.

3.4 A complete grasping system

A simple but complete grasping system that has been implemented with the purpose of testing the controller primitives. It does not make use of the GRASP architecture yet, but we plan to install the architecture in our experimental demonstrator in the next few months.

The design of this system consists of two main components. The first component is a vision system that estimates pose, location and size of a target object. It must be noticed that within GRASP consortium alternative visual approaches are being developed to provide these information.

The second component is the grasp execution that uses the grasp primitive controller described in section 3.3 to complete the grasping task.

3.4.1 Experimental setup

This grasping system has been implemented on the UJI service robot. This is a prototype of a mobile manipulator designed to assist in every day tasks [PSdP+07].

Within this application we have considered the use of tactile sensors jointly with vision and force, aiming at improving the manipulation skills of the robot.

The robotic setup, depicted on the left of Fig. 3.4, consists of a Mitsubishi PA-10 7 d.o.f. arm mounted on an ActivMedia PowerBot mobile robot. The manipulator is endowed with a three-fingered Barrett Hand and a JR3 force/torque and acceleration sensor mounted at the wrist, between the hand and the end-effector. This sensor not only provides a six dimensions force/torque data, but also measures the compensated linear and angular accelerations, which are six dimensions more.

The hand (Fig. 3.4, on the right) has been improved by adding on the fingertips arrays of pressure sensors designed and implemented by Weiss Robotics. The sensors consist of three 8 x 5 cell matrices, that cover the inner parts of the distal phalanxes of each of the three fingers (see Fig 3.4). Each cell is a square of 2.3 mm side. The sensor is based on resistivity. It is able to detect a complete two dimensional pressure profile by the use of a homogeneous sensor material which is connected to an adequate electrode matrix [WW04].

The application presented in this paper assumes that objects to be grasped are lying on a flat surface inside the workspace of the manipulator. In order to reduce the complexity of the visual processing, the robot world is a black environment in which simple, clear shapes are placed on a table at variable positions and orientations (see Fig. 3.5). The range of possible positions are those that allow to view the object and also keep it at a reaching distance for the hand. There is no previous assumption about the size, position and weight of the object. Nevertheless, the system knows that the object can be of a reduced number of different class (for the moment, only box-like and roughly cylindrical shapes are considered), and uses this information to perform an ad-hoc estimate of its size and position (see next section).

Figure 3.5: Workspace with possible target objects.

3.4.2 Visual analysis and planning

Visual analysis produces an estimation of the pose, distance and size of the object, as well as an identification of its shape type. In our particular case this is achieved through the integration of binocular (stereoptic) and monocular (perspective) visual cues in algorithms based on a computational model of primate visual mechanisms [CdP07, CGP08]. Initially the camera, attached to the wrist of the manipulator, is placed laterally with respect to the workspace, in order to have a perspective view of the target object, and then several images from different views are captured and used.

These results are then used to plan the initial parameters of grasp primitive controller (see Sec. 3.2.1). The target object is approached from top, and then, depending of the object estimated shape, only one grasp will be planned. In the case of box shaped objects, a cylindrical preshape along the shortest side of the box is produced; the same happens for cylindrical shapes. The estimated distance and size of the object are used to set the initial hand position at a convenient distance above the object. The orientation of the hand corresponds to the estimated slant of the object. Size and centroid of the object are also available, and affect both the transport and the preshape component.

At this point the grasp controller is started.

3.4.3 Early experimental results

For what concerns grasp execution and the work of the tactile system, we tested our robotic system in two different conditions, i.e., without or with small environmental changes. The first condition, corresponding to a normal working situation, usually ends with a successful grasp without performing any correction movement. In fact, in almost all cases the input provided by the visual system is good enough to allow the execution of the grasp without the need of correcting hand position or orientation.

While testing the system performance in the second condition, we were introducing on purpose some changes in the object position and/or orientation, to check if the system was able to deal with unexpected and suddenly changing situations. The changes were made after the visual system analysis had been finished so that the real pose of the object was much different from the estimated one, like in the example of Fig. 3.6. In this situations the robot may not be able to grasp the object without the support of the tactile feedback. Using the information about the finger extension and the hand contact with the surface of the object, the orientation of the hand, as well as its position, are corrected in a closed-loop manner. When the difference between the real and estimated object pose is big, more than one correction movement might be required. As a limit, the correction of the grasp can not be performed when the displacement of the object is so large that, after its reaching movement, the hand does not have any contact with the object surface.

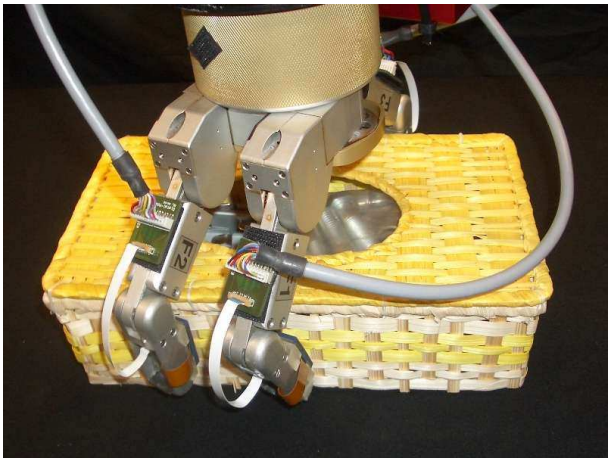


Figure 3.6: Example of grasp in which, after the approaching phase, the hand-object contact is unstable. A correction movement is necessary

3.5 Future work

The work presented in this report regarding the development of grasp primitives is still in an early development stage. Only one grasp primitive has been completed up to the moment. The next immediate goal is to extend the set of implemented primitives to the cylindrical and specially the hook, and pushing/pulling primitives. This would provide a wider and flexible set of options to the controller architecture.

But, a more important task is to embed the developed grasp primitives within the the GRASP control architecture. Several steps have been taken in this direction. The actuator and sensors interfaces of our hardware setup has been implemented using the architecture specifications. But the most important task to be realized in the next few most is to prepare our experimental setup as a demonstrator of the control architecture. This implies to install and run the control architecture, to embed the grasp primitives, and to develop the complementary primitives which are necessary to realize a full pick-and-place action.

Chapter 4

Approaches to modeling sensor uncertainty in grasping

The aim of this chapter is to look into ways how sensor uncertainty can be modeled in grasping, and how this modeling can be used in order to improve control of grasping. As such, there are few publications addressing the issue directly and therefore the topic is approached from several points of view. As a limitation to the focus, we will concentrate in sensors that are used as a feedback during the control, and thus the discussion of modeling static (constant) uncertainties such as calibration errors is only briefly mentioned.

First, the role of uncertainty in control of manipulation, especially in grasping, is outlined, with the traditional approaches emphasized. Then, different types of models applicable for modeling sensor uncertainty are considered. Finally, the phenomenon is examined from the point of view of different sensors applicable in grasping tasks, including proprioceptive, tactile, force/torque, and visual sensors. Most of the work in this chapter is a collection of literature, but the analysis of visual control uncertainty contains some new results developed within the GRASP project.

4.1 Role of uncertainty in robotic grasping

There are several sources of uncertainty in robotic manipulation, including 1) uncertainty in embodiment knowledge, 2) uncertainty in environment (world) knowledge, and 3) uncertainty in sensor measurements. A typical example of uncertainty in embodiment knowledge are robot calibration errors. These errors should be mainly taken into account by performing accurate calibration of the robot, while the remaining small errors can be coped with through the use of different kinds of sensor feedback. The uncertainty of environment knowledge relates to measurements of the environment attributes made outside the control loop, for example, uncertainties in locations of workpieces or uncertainties in the friction coefficients of surfaces. Finally, we use the term sensor uncertainty for uncertainties of measurements which are used in the closed-loop control of the robot. These sensors can be proprioceptive, tactile, force/torque or visual.

The robot grasping research has traditionally concentrated into analysing the stability of grasps through form and force closure analysis. Then, with the help of understanding the form and force closure, the work has concentrated on grasp planning. The grasp planning is performed off-line and thus does not relate directly to the sensor uncertainty. However, we will briefly overview literature related to uncertainty modeling in grasp planning. Already in 1988, Brost proposed a 2-D grasp planning strategy for which if the maximum uncertainties for object and gripper orientations and friction coefficient are known, it is possible to make guaranteed grasp plans, i.e., all grasping moves are guaranteed to succeed even in the case of worst-case modeled uncertainties [Bro85]. The effect of uncertainties in friction and contact position to the closure properties of grasps has been analyzed in [ZQ05].

As we stated earlier, the consideration of uncertainty with respect to on-line closed loop control has received little attention. In a more general context outside grasping, Bruyninckx [Bru95] considers on-line identification of uncertainties in instantaneous geometric parameters, i.e., the position of contact

points, the direction of contact normal, and the local curvature parameters in force controlled compliant motion.

Son and Howe use tactile sensing for stiffness control in the presence of uncertainty [SH96]. The orientation uncertainty of an object grasped by two fingers in a two-dimensional case is considered. More precisely, the orientation of a grasped object is estimated through tactile sensing. However, the paper concentrates on developing a stiffness controller responding to errors in grasped object location and does not address the modeling of the uncertainties.

4.2 Uncertainty models

There are two main approaches for modeling uncertainty in robotics, probabilistic approaches which describe the uncertainty as a probability distribution, and interval models, which describe the bounds for uncertainty.

4.2.1 Probabilistic models

Here, we briefly review the main probabilistic models used in robotics. For a more complete discussion, the reader is asked to refer for example to [TBF05], which discusses probabilistic robotics especially from the mobile robotics perspective of navigation, localization, and mapping.

Probabilistic models of uncertainty model both the world knowledge and sensors with probability density functions. Thus, let $\mathbf{x}_t$ be the uncertain state of the world at time t. Then, the evolution of the state is described with the (probabilistic) system model $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, describing the time evolution of the state given the previous state and the system input $\mathbf{u}$. Perception of the system is then modeled by an observation model $p(\mathbf{y}_t|\mathbf{x}_t)$ relating the observations to the state.

Possible models to the distributions above include simple parametric density models, mixture distributions, and sequential Monte Carlo models, better known in robotics as particle filtering.

The most common family of models is parametric Gaussian models, in which the uncertainty in state, system evolution, and measurement is modeled with a normal (Gaussian) distribution. With linear system and observation models this leads to Kalman filtering as an optimal solution for state estimation. With non-linear system models, extended Kalman filtering (EKF) or unscented Kalman filtering (UKF) can be used to linearize the model. Iterated extended Kalman filter (IEKF) can be used to decrease the error with a non-linear measurement model. However, the true posterior distributions with non-linear models are not anymore Gaussians, and thus the model does not correspond to the true distribution, and EKF and UKF are suboptimal approaches.

Mixture distributions, such as mixtures of Gaussians can also be used to model the uncertainties. Approaches similar to EKF and UKF can be then used for state estimation, sometimes called Gaussian sum filtering [KD03], or multiple model estimators [BSLK01]. With mixture models, more complex uncertainty distributions can be modelled, as a mixture model can also be used to approximate any probability density function. However, the use of mixture models typically introduces a problem with the computational complexity of state estimation, as typically the number of mixture components required grows exponentially over time [BSLK01]. Because this is not tractable, the number of components is kept fixed, by using a fixed number of mixture components to approximate the posterior state.

Particle filtering has become a popular method in robotics for modeling complex probabilistic phenomena, as it allows the modeling of arbitrary probability density functions [TBF05]. However, the particle filtering is also prone to be not tractable for computational reasons as complex distributions require a great number of particles. Even with non-linear system and measurement models, particle filtering is optimal in mean, that is, the mean of several particle ensembles approaches the true density function. Thus, particle filtering performs better with non-linear system and measurement models compared to linearization approaches based on Kalman filtering, if the number of particles is sufficient. For example, non-linear sensor models with several different failure modes have been used in mobile robot navigation for sensors such as sonars (e.g. [TBF05]) and wireless network signal strength [LK08].

4.2.2 Interval models

Interval analysis is an approach where uncertainties are modeled as allowed ranges (or boundaries) instead of statistical distributions [JKDW01]. Thus, estimates can sought within restricted uncertainty bounds. While the use of interval analysis in intelligent robotics is currently rare compared to statistical methods, it provides a useful alternative approach with many possible applications from kinematics to mobile robot navigation [JKDW01].

An advantage of interval analysis compared to statistical approaches is the possibility to perform bounded error parameter estimation [1]. Thus, interval analysis can be thought to take into account the worst-case behavior of a system.

4.3 Sensor types in grasping

When discussing the uncertainty of real-world sensors, it is important to remember to consider the concepts of precision and accuracy separately. Accuracy, that is the measurement bias, can be improved through calibration, but only up to the precision. Thus calibration errors affect the accuracy significantly. In the following we will concentrate on errors caused by factors other than calibration.

4.3.1 Proprioception

By proprioception we mean here the sensors used for sensing the joint angles of a robot. In mobile robotics there is a significant amount of work on modeling the uncertainty of proprioception, see for example [TBF05]. However, most of these are based on either modelling the errors with a Gaussian distribution on a higher-level concept such as angular velocity, or modelling the low-level sensor uncertainty (e.g., wheel encoder error) with a Gaussian distribution.

Another line of study relevant to proprioception errors is the study of uncertain kinematics (e.g., [CHKA03, DDZ00]). While typically the modeling is based on uncertainty in the kinematic parameters (calibration) instead of the proprioceptive measurements, similar analysis could be performed to consider the sensor errors.

4.3.2 Tactile sensors

One challenge in modeling tactile sensors is the number of different tactile sensor types [LN99], including magnetic [Now91], miniaturized force/torque sensors [LMH95], resistive sensor matrices [SWMF01, JWR97, KWW03] and PVDF (polyvinylidene fluoride) sensors [JWR97]. Each type has different characteristics, and the uncertainty analysis of the sensors is seldom presented. An exception to this is [OTKI06], presenting an experimental analysis of uncertainty for a tactile sensor consisting of twelve three-axis force sensors.

Currently, the partners in the GRASP project are using tactile sensors from Weiss Robotics, technically similar to [KWW03]. Resistance changes are measured and mapped to local pressure. The mapping is strongly non-linear, and is linearized using analog electronics [KWW03]. A model describing the dependence of resistance versus load is presented in [WW05]. Due to the non-linearity of the dependence, the sensor uncertainty distribution is very likely to be skewed, such that an assumption of Gaussian errors will not be very accurate. Considering the GRASP project, it is suggested that the sensor uncertainty would be measured experimentally such that in addition to the bias and the variance, the whole distribution of errors is considered.

4.3.3 Force/torque sensors

After joint encoders, force/torque sensors are the most common of the sensor types, and therefore sensors are readily available from several vendors, such as ATI and JR3. For industrial sensors, the resolution is

[1] Interval analysis can also be thought equivalent to statistical estimation where uncertainties are modelled as uniform distributions.

often specified, but the absolute accuracy is unknown. The modeling of measurement uncertainties needs then to be performed experimentally, as the internal operation of the sensor is not known.

Dynamic behavior of force/torque sensors has been analyzed in order to understand the behavior and speed up the dynamic response in [LC98, XLZ07]. The approach could be used also in considering the uncertainty estimation.

4.3.4 Vision

The direct use of visual feedback for closed loop control, visual servoing, has great promises for integrating vision tightly in the control of a robotic system. Some of the possibilities include decreasing the adverse effects of calibration and measurement errors as well as uncertainties in the world model by closed loop control, and decreasing latency, especially useful for acting in dynamic settings with moving targets. In grasping, visual servoing is most often used to track and grasp moving objects (e.g., [ATYM93, NN00]), or increase robustness (e.g., [HDE98, KC03]). To grasp objects with high velocity special high-speed vision hardware has been proposed [NNII99].

Many of the key research problems with visual servoing are related to the performance of visual servoing methods in the presence of measurement and system modeling errors. For example, the effect of camera calibration errors has been studied in [Esp93]. Also, the convergence properties of the control part of the systems have been evaluated extensively, for example in [Cha98, MC02].

It has recently been demonstrated that the effect of measurement errors on the open loop trajectory in visual servoing can be estimated using linear propagation of errors [KKC06]. In a soon to be published paper, the earlier work is extended to image-based visual servoing and it is shown how the analysis of open loop uncertainty can be extended to the case of closed loop control for both position-based and image-based visual servoing [Kyr09].

References

[ASK02] S. Aramaki, H. Shirouzu, and K. Kurashige. Control program structure of humanoid robot. *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, 3:1796–1800 vol.3, Nov. 2002.

[ATYM93] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2), 1993.

[BK00] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *IEEE International Conference on Robotics and Automation*, April 2000.

[BRJ98] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.

[Bro85] Randy C. Brost. Planning robot grasping motions in the presence of uncertainty. Technical Report CMU-RI-TR-85-12, Computer Science Department and Robotics Institute, Carnegie Mellon University, 1985.

[Bru95] Herman Bruyninckx. *Kinematic Models for Robot Compliant Motion with Identification of Uncertainties*. PhD thesis, Katholieke Universiteit Leuven, Department of Mechanical Engineering, 1995.

[BSLK01] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Appications to Tracking and Navigation*. Wiley Interscience, 2001.

[Cas05] U. Castiello. The neuroscience of grasping. *Nature Neuroscience*, 6:726–736, sep 2005.

[CdP07] E. Chinellato and A. P. del Pobil. Integration of stereoscopic and perspective cues for slant estimation in natural and artificial systems. In Jose Mira and Jose R. Alvarez, editors, *Nature Inspired Problem-Solving Methods in Knowledge Engineering*, page 399408. Springer, 2007.

[CF06] C-F. Chang and L-C Fu. A hybrid system design of a mobile manipulator. In *Proc. IEEE ICRA'06*, pages 406–411, May 2006.

[CGP08] E. Chinellato, B. J. Grzyb, and A. P. Del Pobil. Brain mechanisms for robotic object pose estimation. In *Intl. Joint Conf. on Neural Networks*, Hong Kong, 2008.

[CH08] François Chaumette and Seth Hutchinson. Visual servoing and visual tracking. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 563–583. Springer-Verlag, Berlin, Germany, first edition, 2008.

[Cha98] François Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The Confluence of Vision and Control*, number 237 in Lecture Notes in Control and Information Sciences, pages 66–78. Springer-Verlag, 1998.

[CHKA03] C. C. Cheah, M. Hirano, S. Kawamura, and S. Arimoto. Approximate jacobian control for robots with uncertain kinematics and dynamics. *IEEE Transactions on Robotics and Automation*, 19(4), 2003.

[CMFP05] E. Chinellato, A. Morales, R. B. Fisher, and A. P. Del Pobil. Visual quality measures for characterizing planar robot grasps. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 35(1):30–41, 2005.

[DB98] C. Davidson and A. Blake. Caging Planar Objects with a Three-Finger One-Parameter Gripper. In *IEEE International Conference on Robotics and Automation*, pages 2722–2727, Leuven, Belgium, May 1998.

[DDZ00] W. E. Dixon, D. M. Dawson, and E. Zergeroglu. Tracking and regulation control of a mobile robot system with kinematic disturbances: A variable structure-like approach. *Journal of Dynamic Systems, Measurement and Control*, 122(4), 2000.

[DK08] R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University, July 2008.

[Esp93] Bernard Espiau. Effect of camera calibration errors on visual servoing in robo tics. In *3rd International Symposium on Experimental Robotics*, pages 182–192, Kyoto, Japan, October 1993.

[FM09] Javier Felip and Antonio Morales. Robust sensor-baser grasp primitive for a three-figner grasp. In *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, October 2009. Submitted. Review notification by June 2009.

[HBF+03] S. Haidacher, J. Butterfass, M. Fischer, M. Grebenstein, K. Joehl, K. Kunze, M. Nickl, N. Seitz, and G. Hirzinger. DLR hand II: Hard- and software architecture for information processing. In *Proc. IEEE ICRA'03*, pages 684–689, September 2003.

[HDE98] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. *IEEE Transactions on Robotics and Automation*, 14(4), 1998.

[HLT+00] L. Han, Z. Li, J. C. Trinkle, Z. Qin, and S. Jiang. The planning and control of robot dextrous manipulation. In *Proc. IEEE ICRA'00*, pages 263–269, September 2000.

[JKDW01] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. *Applied Interval Analysis*. Springer, 2001.

[JMLH05] Han-Young Jang, Hadi Moradi, Sukhan Lee, and JungHyun Han. A visibility-based accessibility analysis of the grasp points for real-time manipulation. In *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, Edmonton, Canada, August 2005.

[JWR97] J. Jockusch, J. Walter, and H. Ritter. A tactile sen-sor system for a three-fingered robot manipulator. In *IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico, Apr 1997.

[KC03] D. Kragic and H. I. Christensen. Robust visual servoing. *International Journal of Robotics Research*, 22(10–11), 2003.

[KD03] Jayesh H. Kotecha and Petar M. Djuric. Gaussian sum particle filtering. *IEEE Transactions on Signal Processing*, 51(10), Oct 2003.

[KKC06] Ville Kyrki, Danica Kragic, and Henrik Christensen. Measurement errors in visual servoing. *Robotics and Autonomous Systems*, 54(10):815–827, 2006.

[KS08] D. Kortenkamp and R. Simmons. Robotic systems architecture and programming. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Germany, first edition, 2008.

[KWW03] Oliver Kerpa, Karsten Weiss, and Heinz Wörn. Development of a tactile sensor system for a humanoid robot. In *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, 2003.

[Kyr09] Ville Kyrki. Control uncertainty in image-based visual servoing. In *IEEE International Conference in Robotics and Automation*, 2009. Accepted to be published.

[LC98] Y. F. Li and X. B. Chen. On the dynamic behavior of a force/torque sensor for robots. *IEEE Transactions on Instrumentation and Measurement*, 47(1):304–308, 1998.

[LK08] Janne Laaksonen and Ville Kyrki. Localization in ambiguous environments using multiple weak cue s. *Intelligent Service Robotics*, 1:281–288, 2008.

[LMH95] H. Liu, P. Meusel, and G. Hirzinger. A tactile sensing system for the dlr three-finger robot hand. In *International Symposium on Measurement and Control in Robotics*, 1995.

[LN99] M. H. Lee and H. R. Nicholls. Tactile sensing for mechatronics: a state of the art survey. *Mechatronics*, 1999.

[MAA+06] A. Morales, P. Azad, T. Asfour, D. Kraft, S. Knoop, R. Dillmann, A. Kargov, Ch. Pylatiuk, and S. Schulz. An anthropomorphic grasping approach for a humanoid robot. In *International Symposium on Robotics*, Munich, Germany, May 2006. On CD.

[Mas81] Matthew T. Mason. Compliance and force control for computer controlled manipulators. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(6):418–432, June 1981.

[MC02] Ezio Malis and François Chaumette. Theoretical improvements in the stability analysis of a new cl ass of model-free visual servoing methods. *IEEE Transactions on Robotics and Automation*, 18(2):176–186, April 2002.

[MKCA03] A.T. Miller, S. Knoop, H.I. Christensen, and P.K Allen. Automatic grasp planning using shape primitives. In *IEEE International Conference on Robotics and Automation*, Taipe, Taiwan, Septmeber 2003.

[MKF+05] G. Milighetti, H. B. Kuntze, C. W. Frey, B. Diestel-Feddersen, and J. Balzer. On a primitive skill-based supervisory robot control architecture. In *Proc. IEEE ICRA'05*, pages 141–147, July 2005.

[MSdF06] A. Morales, P.J. Sanz, A.P. del Pobil, and A.H. Fagg. Vision-based three-finger grasp synthesis constrained by hand geometry. *Robotics and Autonomus Systems*, 54(6):496–512, June 2006.

[NN00] H. Nomura and T. Naito. Integrated visual servoing system to grasp industrial parts movingon conveyer by controlling 6dof arm. In *IEEE International Conference on Systems, Man, and Cybernetics*, Nashville, TN, USA, Oct 2000.

[NNII99] A. Namiki, Y. Nakabo, I. Ishii, and M. Ishikawa. High speed grasping using visual and force feedbac. In *IEEE International Conference on Robotics and Automation*, Detroit, MI, USA, May 1999.

[Now91] W. C. Nowlin. Experimental results on bayesian algorithms for interpreting compliant tactile sensing data. In *IEEE International Conference on Robotics and Automation*, Sacramento, California, Apr 1991.

[OTKI06] Hiroko Oshima, Nobutaka Tsujiuchi, Takayuki Koizumi, and Akihito Ito. Verification of tactile sensor for manipulator. In *IEEE International Conference on Robotics and Biomimetics*, 2006.

[PdPS06] Mario Prats, Angel P. del Pobil, and Pedro J. Sanz. A control architecture for compliant execution of manipulation tasks. In *Proc. IEEE/RSJ IROS'06*, pages 4472–4477, October 2006.

[PEC99] L. Petersson, M. Egerstedtt, and H.I. Christensen. A hybrid control architecture for mobile manipulation. In *Proc. IEEE/RSJ IROS'99*, pages 1285–1291, 1999.

[PFG02] R. Platt Jr., A. H. Fagg, and R. Gruppen. Nullspace composition of control laws for grasping. In *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, pages 1717–1723, Lausanne, Switzerland, 2002.

[PSdP+07] M. Prats, P. J. Sanz, A. P. del Pobil, E. Martínez, and R. Marín. Towards multipurpose autonomous manipulation with the uji service robot. *Robotica Journal*, 2007. Accepted.

[SB99] Th. Schlegl and M. Buss. A discrete-continuous control architecture for dextrous manipulation. In *Proc. IEEE SMC'99*, volume 2, pages 860–865, 1999.

[SH96] Jae S. Son and Robert D. Howe. Tactile sensing and stiffness control with multifingered hands. In *IEEE International Conference on Robotics and Automation*, pages 3228–3233, Minneapolis, Minnesota, USA, April 1996.

[SWMF01] H. Sugiuchi, S. Watanabe, T. Morino, and R. Fukuno. A control system for multi-fingered dual robotic hand with distributed touch sensor. In *Proceedings of the 32nd International Symposium on Robotics*, Apr 2001.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, 2005.

[TM00] M. Teichmann and B. Mishra. Reactive robotics I: Reactive grasping with a modified gripper and multi-fingererd hands. *International Journal of Robotics Research*, 19(7):697–708, 2000.

[WJL+05] B. Wang, L. Jiang, J.W. Li, H.G. Cai, and H. Liu. Grasping unknown objects based on 3d model reconstruction. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Monterrey, California, July 2005.

[WW04] K. Weiss and H. Wörn. Tactile sensor system for ana anthropomorphic robot hand. In *IEEE International Conference on Manipulation and Grasping*, Genua, Italy, July 2004.

[WW05] K. Weiss and H. Wörn. The working principle of resistive tactile sensor cells. In *IEEE International Conference on Mechatronics and Automation*, 2005.

[XLZ07] Ke-Jun Xu, Cheng Li, and Zhi-Neng Zhu. Dynamic modeling and compensation of robot six-axis wrist force/torque sensor. *IEEE Transactions on Instrumentation and Measurement*, 56(5):2094–2100, 2007.

[ZQ05] Yu Zheng and Wen-Han Qian. Coping with the grasping uncertainties in force-closure analysis. *International Journal of Robotics Research*, 24(4):311–327, 2005.

Appendix A

Submitted scientific article

A submitted scientific article is included to complete the contents of the deliverable. It extends the contents of Chapter 3 by implementing a more robust primitive to execute power grasps.

Javier Felip and Antonio Morales. Robust sensor-baser grasp primitive for a three-figner grasp. In *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, October 2009. Submitted. Review notification by June 2009

Robust sensor-based grasp primitive for a three-finger robot hand

Javier Felip and Antonio Morales

***Abstract*— This paper addresses the problem of robot grasping in conditions of uncertainty. We propose a grasp controller that deals robustly with this uncertainty using feedback from different contact-based sensors. This controller assumes a description of grasp consisting of an instance of primitive that only determines the initial configuration of the hand and the control law to be used.**

We exhaustively validate the controller by carrying out a large number of tests with different degrees of inaccuracy in the pose of the target objects and by comparing it with results of a naive grasp controller.

I. INTRODUCTION

Management of uncertainty is one of the big problem to address when developing applications for unstructured scenarios. In the case of robot grasping, uncertainty can arise from several sources: the shape and physical properties of the target objects are completely or partially unknown, the pose of the object can not be determined accurately, the configuration of the robot (i.e.: position of mobile robot) is not accurate enough, and many others.

Analytical solutions to the grasp planning problem has been provided for structured scenarios [1]. However these solutions often depend on the assumption that the contact locations obtained as solutions are reachable by actuators with enough precision. For common robots scenarios this is not realistic even in the case that the shape of the object is perfectly known. Several attempts to design analytical grasp planning algorithms that take into account a certain degree of inacuracy has been made. In an early work Brost [2] proposed a grasp planning algorithm that was able to compute stable grasps if knowing the error limits of gripper orientation and friction coefficient. More recently Zheng and Qian [3] analysed the impact of variations in the friction coefficient and in the contact location on the force-closure condition. Both works focus exclusively on the 2D case.

A common approach to reduce uncertainty is the use of sensor information in the planning and execution phases of grasping. Vision has been used to obtain the shape of unknown target objects [4], [5], or to determine the location and pose of them [6]. In both cases, visual input is used to plan feasible grasps. Visual feedback is also used when the arm tries to reach the object. Murphy et al. uses visual techniques to correct the orientation of four-finger hand while approaching an object to allow better contact locations [7]. Namiki et al. uses a fast control schema in combination with tactile feedback to cage an object[8]. Infrared sensors has been also used to correct the approaching orientation of the gripper [9]. In innovative design Hsiao et al. use IR sensors to estimate the normal direction of closer object surfaces to search a suitable contact location[10].

J. Felip and A. Morales are with Robotic Intelligence Laboratory at the Department of Computer Science and Engineering, Universitat Jaume I, 12006 Castellón, Spain {jfelip,morales}@uji.es

Fig. 1. PA-10 7 d.o.f with Barrett Hand and a JR3 force/torque and acceleration sensor. The hand has Weiss Robotics pressure sensors on its fingertips.

Once the object is contacted with the robot gripper tactile and force sensors can be applied. Contact force measurement it is used to estimate the quality of the grasps [11], [12] or the shape of the object [13] with the purpose of reach better contact locations through a sequence of grasping/regrasping actions. Contact information can also be used to program complex dexterous manipulation operations like finger repositioning while holding the object[14]. Several works have combined the use of several sensors to complete the whole process of grasp planning and execution [15], [16].

Robustness in grasp execution is not only achieved by designing sensor-based controllers but also by combining several controllers with different optimisation goals. These combinations has been based on hierarchical schemes based on reflex programming [17], [12] or complementary controllers [11], [10].

A. Grasp primitives

In this paper we follow a sensor-based approach that is based on an alternative paradigm of describing grasps. Most of the above papers assume that grasps are described as set of contact points on the object surface. In fact, most of the problems arise as a consequence of the impossibility of reaching those points. Our paper is developed under a different assumption. Grasps are describes as instances of basic primitives [16]. A grasp primitive is a specific controller designed to perform a particular indivisible action, in our case a grasp. In practical terms it is defined by a initial hand preshape, a sensor-based controller, and a

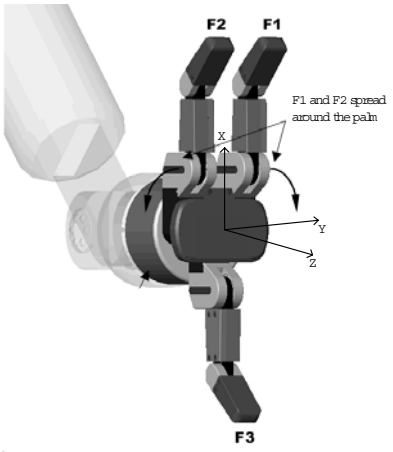


Fig. 2. Detail of the Barrett hand.

set of ending conditions. Its behaviour can be determined by several parameters like initial position and orientation, maximum force allowed, and others. An *instance* of a grasp primitive are the set of values of the parameters. Hence, a grasp is an instance of a primitive that determines the initial configuration of the robot hand and the control policy that is going to be applied to execute the grasp.

This definition of grasp primitive presents two perspectives. From a practical point of view a grasp primitive is a single controller that performs a specific task on a particular embodiment. From an abstract point of view, primitives are the simplest pieces of a vocabulary to elaborate plans. Hence, they are well suited to be the basic piece of a reasoning and learning procedures.

The concept of grasp primitive is not new and has been used in many other robot-related works. Actually the term "motor primitive" is borrowed from neuroscience literature [18], and has also been widely used in robot learning [19], [20]. Nagatani and Yuta implemented and combined several action primitives to perform a complex behaviour: a mobile robot behaviour capable of opening and going through a door [21]. Aarno et al. implement visual analysis to program "Elementary Grasping Actions (EGA)", a kind of grasp primitives, for a parallel gripper [5]. Finally Finite State Machines has been proposed to combine primitive actions in the execution of complete manipulation tasks [22].

Paper outline

This papers presents the implementation and validation of a robust grasp primitive for a three-finger Barrett hand that uses sensor feedback form force/torques sensors and tactile sensors. Section II describes the design principles of the grasp primitive. This primitive is tested through a exhaustive series of experiments (Sec. III), which results are described and discussed in sections IV and V.

II. Methodology

A. System description and Assumptions

We implemented our primitive for a robotic setup consisting of a Mitsubishi PA-10 7 d.o.f. (Degrees of freedom)

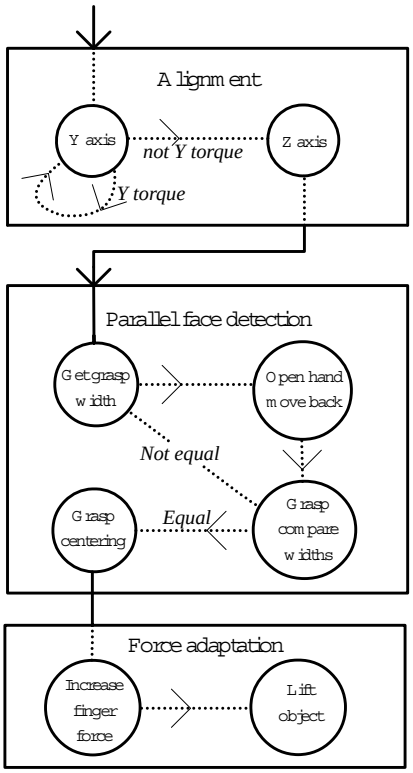


Fig. 3. Algorithm execution diagram.

mounted on an Active Media PowerBot mobile robot. The manipulator is endowed with a three-fingered Barrett Hand and a JR3 force/torque and acceleration sensor mounted at the wrist, between the hand and the end-effector. The hand has been improved by adding on the fingertips arrays of pressure sensors designed and implemented by Weiss Robotics.

The Barrett hand is a 4 d.o.f. three-fingered hand. Each finger has one degree of freedom thus phalanxes are not independent. F1 and F2 can rotate around the palm and move next to F3 (Thumb) or oppose to it, this d.o.f. is called adduction. The reference frame of the hand and the adduction d.o.f. are depicted in Fig. 2. Each finger of the hand has built-in strain sensor. The JR3 is a 12 d.o.f. sensor that measures force, torque and acceleration in each direction of the space.

Out experimetnal workspace consists of horizontal surface where the targets objects are lying. The objects that can be manipulated are those that can fit inside the hand, at least 25mm height and 70mm long, the minimum width is 10mm. Big objects that can be held by the hand should have a maximum width of 200mm. The objects used for the development of the controller are box-like and cylinder-like (see Figs. 6, 7 and 8).

The input of the primitive controller is the starting position and orientation of the hand and the maximun finger force. In optimal conditions the hand will perfectly oriented in the direction of the object, and rotated perpendicularly with respect the main axis of the object bounding box. These input parameters are provided from an external module based on

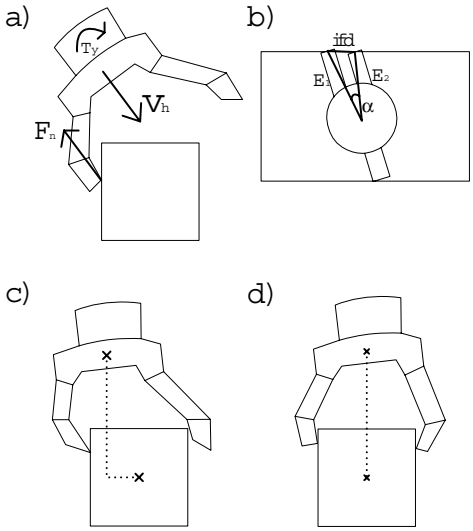


Fig. 4. Correction of alignment errors. a) The hand touches the object. The contact is not perpendicular and a normal force (Fn) appears at a distance from the center creating the torque Ty. b) The hand Z error is calculated using Index and Middle finger extensions. c) The grasp center is displaced. d) Y translation error correction, result of grasp center correction.

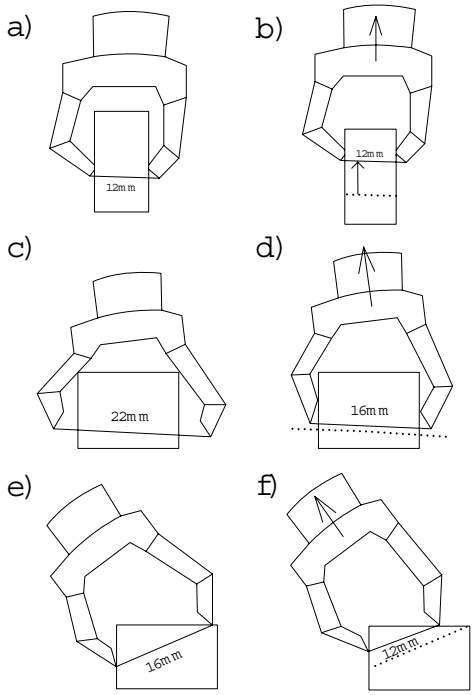


Fig. 5. Determining parallelism of grasped faces: a) First contact. b) Grasp width is constant, the faces grasped are parallel. c) Inner phanlanxes contact the object. d) Grasp width is not the same, the faces are not parallel.

visual information.

The designed controller is able to deal with errors in determination of the parameters. The estimitation is decomposed in translation error and rotation error (see Fig. 9). The former is defined by the cartesian distance on each frame axis between the center of the object and its estimation. The latter is calculated from the difference between each estimated object axis and its true orientation.

This controller tries to grasp the object aproaching from above following and orientation close to the vertical one. In order to allow lateral approaching directions several changes in the desing of the contorller should be necessary. Basically an estimation of the distance to the object should be knoen in advance. During the grasp execution the robot can inadvertly move the object but the controller is designed to take into account most of these cases.

This algorithm starts with a cylindrical preshape. That is F1 and F2 compltely oppose thumb finger F3. In some cases the hand preshape is switched to a spherical configuration where the fingers are arranged in an equilateral triangle.

B. Algorithm

The controller tries to obtain grasp stability on the basis of the following criteria (ordered by relevance):

- Hand-object alignment
- Parallelism of grasped faces
- Maximization of contact surface
- Finger position symmetry
- Finger force symmetry

Hand and object are aligned when all their axis are parallel and the projection of the Z axis of the hand intersects the object bounding box on its center. The alignment avoids torque forces from appearing when lifting the object.

The starting position and orientation of the hand is a critical parameter for the algorithm. This pose sets the approach vector to the object which is along the positive Z axis of the hand (see Fig. 2). The aim of the algorithm is to perform a stable grasp of the object allowing a considerable error in starting position. Thus the importance of the object feature estimation decreases and the possibility of grasping an object is less dependent from the accuracy of the estimations.

The execution of the grasp is divided into three main phases (see Fig. 3).

1) Alignment: This phase tries to align hand and object using force and tactile feedback. First of all the hand moves forward until the force/torque sensor on the wrist detects contact with the object. If this contact causes a torque force around Y axis it means that the object and the hand are not aligned (see Fig. 4.a). To correct this error the hand moves back and rotates an angle of two degrees, then continues touching the object until the torque disappears or it changes the sign. If the torque changes the sign, the hand rotates one degree in the opposite direction and the Y alignment ends.

At this point the hand closes. The difference in the extension of the F1 and F2 fingers, determine the rotation around Z axis (see Fig. 4.b) needed to align with the object. Both must have the same extension to have an stable grasp. This correction is not applied if the spherical pregrasp shape is set.

2) Parallel face detection: In the second stage the controller tries to determine if the grasped surfaces are parallel and stable, the differente width of two consecutive grasps is used to determine the grasp stability: The width of the

Fig. 6. Cylinder-like objects. Properties (radius x height, weight) from left to right and top to bottom: Cylinder1(110x80, light) Cylinder2(65x215, light) Cylinder3(105x75, heavy) Cylinder4(115x50, light)

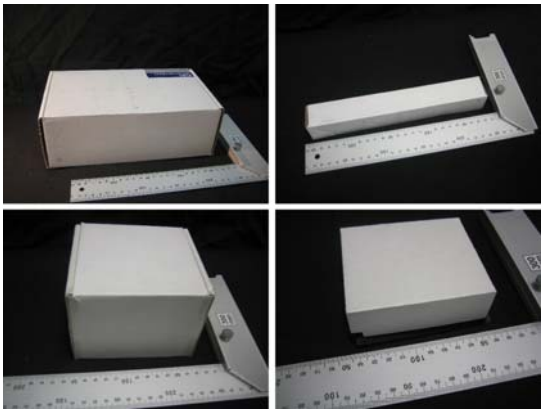


Fig. 7. Box-like objects. Properties (base x height, weight) from left to right and top to bottom: Box1(270x53x95, light) Box2(236x35x35, light) Box3(127x116x92, heavy) Box4(100x87x45, light)

current grasp is measured (see Fig. 5.a and c), then the hand opens a little and moves 5mm backwards. After that the hand closes and the width of the grasp is measured again. If there is a big difference between the two samples it means that the grasped faces are not parallel (see Fig. 5.d) and the process starts again. This phase of the algorithm repeats until the difference is close to zero or the object is lost. If the object is lost a reflex that will try to recover it is triggered, this reaction is explained at the end of this section.

When the grasp is stable, the fingers move the object aligning it with the palm center and keeping the extension of the fingers (see Fig. 4.c and d) in order to improve contact surface, finger position symmetry, and finger force symmetry.

3) Force adaptation: Using the fingertip integrated force sensors of the Barrett hand, the force of each fingertip is increased until it reaches the predefined limit. Then, the hand lifts the object and evaluates if the grasp has benn successful.

C. Security reflexes

During the execution of the primitive, the algorithm is attentive for some important events in order to inform the user, adapt to the environment conditions and perform a successful grasp. The first event is the loss of the object. This happens when the three finger close completely without making contact with the object. In this case the hand opens and moves a little bit down then closes again trying to recover object contact.

Another event is the adduction of the fingers, when contacting the object the adduction degree is set free. Depending on the shape of the object the opposite fingers can adduct. If this happens it is assumed that an spherical preshape is more appropriate for the the object.

The last event is the miss of the object by only one finger. The reaction is to open the hand and to move laterally 5cm (inter finger distance).

D. Additional parameters

Other parameters as distance, size, weight and shape could be used to improve the accuracy and execution of the grasp. The distance could be used avoid blind first contact with the object; the size could be used to set the starting opening of the fingers; the weight to determine the force to be applied by the fingers; and the shape to set the pregrasp shape reducing the time consumed by the pregasp shape detection and switching.

III. Validation

A test bench has been designed in order to validate the grasping controller. This test bench consists of a set of objects and a set of starting positions to be tried with each object.

To have a comparison reference for outr controller, we have designed an alternative naive grasp controller without corrections. This controller needs 4 input parameters: starting position, distance to the object, pregrasp size and finger force. The fingers moves to the pregrasp size and the hand moves forward along its Z axis the distance specified. The hand closes and lifts the object. If the object is lifted and does not fall for 10 seconds, the execution is successful.

A. Objects and test bench

The objects selected are classified according to their shape (cylinders in Fig. 6, boxes in Fig. 7 and others in Fig. 8), their size (thin, normal, thick) or their weight (light, heavy). All the objects are solid. Following the shape classification, we have selected thin, normal and thick objects for each shape in order to test as many different combinations of object features as possible. The optimal conditions have been tested in all the objects. We have selected a subset of 2

Fig. 8. Other objects. Properties (base x height, weight) from left to right: Other1(180x90x90, heavy) Other2(90x90x163, light)

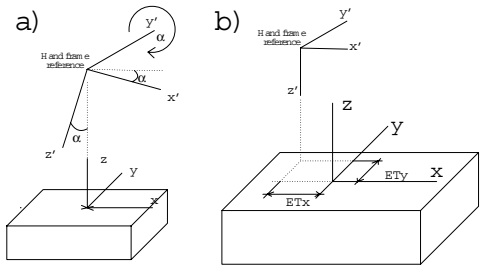


Fig. 9. Rotation and translation error, the object frame reference is on the center of the object. a) Example of alpha degrees Y rotation error. b) Example of X and Y translation error

box-like objects and 2 cylinder-like objects to test rotation and translation error conditions. This selected objects are the biggest and the smallest from each category.

Translation error is the deviation from the center of the object to the center of the hand and it is measured in mm. Rotation error is the deviation between hand and object main axis and is measured in degrees (see Fig. 9).

This test bench evaluates robustness against translation and rotation errors. The behaviour of each algorithm has been also evaluated in optimal conditions which can present a rotation error of 5 degrees and 10% of translation error.

To evaluate the effect of rotation errors the following conditions have been taken into consideration:

- 15 degrees on X, Y, Z, XY, XZ, YZ and XYZ.
- 20 degrees on X, Y, Z

The combined rotation error is applied first in X next in Y and later in Z. A rotation error of 15 degrees in XYZ is a rotation of 15 degree on X, then 15 degree on Y and finally 15 degree on Z. The results of the rotation tests are shown in Table III for the robust controller and in Table IV for the simple controller. The cylinder-like objects are invariant to rotation in Z axis. The Z rotation error is not applicable to cylinder-like objects.

The amount of translation error is relative to the size of the object because usually this two variables (size and error) are related. To evaluate the effect of translation errors the following conditions have been taken into consideration:

- 20% on X, Y, Z, XY, XZ, YZ and XYZ
- 40% on X, Y, Z

IV. Results

The global results are presented in Table I and Table II, the first column shows the results for the optimal case, the second and third columns present the summary of the rotation and translation error. The last column shows the averaged results for each object.

Details about experiments with error conditions can be found in Table III and Table IV for rotation error and in Table V and Table VI for translation error.

V. Discussion

Summary tables I and II show clearly the better performance obtained by our robust controller in comparison

	Optimal	Rotation	Translation	Total
Box 1	100%	100%	54%	85%
Box 2	100%	73%	92%	88%
Cylinder 2	100%	58%	80%	79%
Cylinder 4	100%	100%	95%	98%

TABLE I

Robust algorithm global results

	Optimal	Rotation	Translation	Total
Box 1	100%	50%	40%	63%
Box 2	100%	52%	100%	84%
Cylinder 2	100%	92%	88%	93%
Cylinder 4	100%	96%	70%	89%

TABLE II

Simple algorithm global results

Error	Box 1	Box 2	Cylinder 2	Cylinder 4
15 X Axis	2/2(100%)	2/2(100%)	2/2(100%)	2/2(100%)
20 X Axis	2/2(100%)	2/2(100%)	2/2(100%)	2/2(100%)
15 Y Axis	3/3(100%)	2/2(100%)	1/2(50%)	3/3(100%)
20 Y Axis	3/3(100%)	2/2(100%)	1/4(25%)	2/2(100%)
15 Z Axis	3/3(100%)	2/2(100%)	N/A	N/A
20 Z Axis	3/3(100%)	2/2(100%)	N/A	N/A
15 XY Axis	2/2(100%)	1/2(50%)	2/2(100%)	2/2(100%)
15 XZ Axis	2/2(100%)	1/2(50%)	N/A	N/A
15 YZ Axis	2/2(100%)	1/2(50%)	N/A	N/A
15 XYZ Axis	2/2(100%)	1/4(25%)	N/A	N/A

TABLE III

Results with rotation error for the robust algorithm

Error	Box 1	Box 2	Cylinder 2	Cylinder 4
15 X Axis	5/5(100%)	5/5(100%)	5/5(100%)	5/5(100%)
20 X Axis	5/5(100%)	5/5(100%)	5/5(100%)	5/5(100%)
15 Y Axis	4/5(80%)	2/5(20%)	3/5(20%)	4/5(80%)
20 Y Axis	2/5(20%)	0/5(0%)	5/5(100%)	5/5(100%)
15 Z Axis	4/5(80%)	5/5(100%)	N/A	N/A
20 Z Axis	4/5(80%)	5/5(100%)	N/A	N/A
15 XY Axis	0/5(0%)	0/5(0%)	5/5(100%)	5/5(100%)
15 XZ Axis	0/5(0%)	4/5(80%)	N/A	N/A
15 YZ Axis	1/5(20%)	0/5(0%)	N/A	N/A
15 XYZ Axis	0/5(0%)	0/5(0%)	N/A	N/A

TABLE IV

Results with rotation error for the simple algorithm

Error	Box 1	Box 2	Cylinder 2	Cylinder 4
20% X Axis	2/2(100%)	2/2(100%)	2/2(100%)	2/2(100%)
40% X Axis	1/2(50%)	2/2(100%)	2/2(100%)	2/2(100%)
20% Y Axis	1/2(50%)	2/2(100%)	1/2(50%)	2/2(100%)
40% Y Axis	0/2(0%)	2/2(100%)	0/2(0%)	1/2(50%)
20% Z Axis	2/2(100%)	2/2(100%)	2/2(100%)	2/2(100%)
40% Z Axis	2/2(100%)	2/2(100%)	2/2(100%)	2/2(100%)
20% XY Axis	1/4(25%)	3/4(75%)	2/2(100%)	2/2(100%)
20% XZ Axis	2/2(100%)	1/2(50%)	2/2(100%)	2/2(100%)
20% YZ Axis	1/2(50%)	1/2(50%)	1/2(50%)	2/2(100%)
20% XYZ Axis	1/4(25%)	3/4(75%)	2/2(100%)	2/2(100%)

TABLE V

Results with translation error for the robust algorithm

Error	Box 1	Box 2	Cylinder 2	Cylinder 4
20% X Axis	0/4(0%)	4/4(100%)	4/4(100%)	4/4(100%)
40% X Axis	0/4(0%)	4/4(100%)	3/4(75%)	4/4(100%)
20% Y Axis	4/4(100%)	4/4(100%)	4/4(100%)	4/4(100%)
40% Y Axis	0/4(0%)	4/4(100%)	4/4(100%)	0/4(0%)
20% Z Axis	3/4(75%)	4/4(100%)	4/4(100%)	2/4(50%)
40% Z Axis	2/4(50%)	4/4(100%)	2/4(50%)	2/4(50%)
20% XY Axis	0/4(0%)	4/4(100%)	4/4(100%)	4/4(100%)
20% XZ Axis	2/4(50%)	4/4(100%)	4/4(100%)	2/4(50%)
20% YZ Axis	4/4(100%)	4/4(100%)	2/4(50%)	4/4(100%)
20% XYZ Axis	1/4(25%)	4/4(100%)	4/4(100%)	2/4(50%)

TABLE VI

RESULTS WITH TRANSLATION ERROR FOR THE SIMPLE ALGORITHM

with the naive one. It not only successes in a 100% of the experiments in optimal conditions but also outperforms the naive one when rotational and translational errors are introduced.

The only exemption to this rule is the case of Cylinder 2 (object on the top-left corner on fig 6). This object is too light and when touched while lying on a surface moves easily. We observed that the successive contacts that our controller produce causes that the object variates its position making not possible to grasp it.

This case shows one of the drawbacks of our approach. Our controller is touching the objects several times before finally closing the finger to catch them. In case of light or unstable objects this can be a problem. This difficulty could be surpassed by the use of more sensitive sensors or proximity sensors in a similar fashion as [10].

One of the advantages of our approach is the little previous information it needs about the object. No exact model of the object is necessary, and the only input is the maximum force to be applied by the fingers. It is supposed that the hand is appropriately oriented and preshaped. More information, like estimated size or the distance to them, would be definitively help to improve the controller robustness and the time necessary to complete a grasp.

Currently all the grasp tried approach from above. That is, the objects are lying on a surface and the hand approaches vertically. This simplifies our controller since the movements of the objects are limited. Improvements are necessary if grasps from a side are going to be executed, since the stability of the objects could be compromised if they are touch. In this case an estimation of the distance to the object would be necessary.

At the moment the average time to execute a grasp is about 40 seconds, though this time depends on the object and the initial position error. It could be reduced providing more information about the location and characteristics of the objects.

Finally, an attached video shows pose correction phases, event adaptation and grasp force increasing. It is also shown that the stability of the grasps performed by the robust algorithm are better than the ones performed by the simple controller.

VI. CONCLUSION

In previous section three are indications of how to improve the grasp primitive controller implemented. The most immediate future is to develop complementary primitives that allow the execution of a complete pick-and-place task, i.e.: approaching and preshaping, lifting, transportation and landing primitives. The development of these primitives would provide a vocabulary of basic skills that will allow planning, and learning in future stages.

To conclude, we remark that we have developed a robust sensor-based grasp primitive that need little information to execute its task and that is able the correct and adapt to variations and inaccuracies in the expected conditions of the scenario.

ACKNOWLEDGMENT

This paper describes research carried out at the Robotic Intelligence Laboratory of Universitat Jaume I. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project), and by Fundació Caixa-Castelló (P1-1A2006-11).

REFERENCES

[1] A. Bicchi, "Hand for dexterous manipulation and robust grasping: A difficult road towards simplicity," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 652–662, 2000.

[2] R. C. Brost, "Planning robot grasping motions in the presence of uncertainty," Tech. Rep. CMU-RI-TR-85-12, Computer Science Department and Robotics Institute, Carnegie Mellon University, 1985.

[3] Y. Zheng and W.-H. Qian, "Coping with the grasping uncertainties in force-closure analysis," *International Journal of Robotics Research*, vol. 24, no. 4, pp. 311–327, 2005.

[4] A. Morales, P. Sanz, A. del Pobil, and A. Fagg, "Vision-based three-finger grasp synthesis constrained by hand geometry," *Robotics and Autonomus Systems*, vol. 54, pp. 496–512, June 2006.

[5] D. Aarno, J. Sommerfeld, D. Kragic, N. Pugeault, S. Kalkan, F. Wörgötter, D. Kraft, and N. Krüger, "Early reactive grasping with second order 3D feature relations," in *IEEE Conference on Robotics and Automation (submitted*, 2007.

[6] P. Azad, T. Asfour, and R. Dillmann, "Combining appearance-based and model-based methods for real-time object recognition and 6d-localization," in *International Conference on Intelligent Robots and Systems (IROS)*, (Beijing, China), 2006.

[7] T. Murphy, D. Lyons, and A. Hendriks, "Stable grasping with a multi-fingered robot hand: A behavior-based approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, (Yokohama, Japan), pp. 867–874, July 1993.

[8] A. Namiki and M. Ishikawa, "Optimal grasping using visual and tactile feedback," in *IEEE/SICE/RSJ Intl. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, (Whashington, DC), pp. 589–596, Dec. 1996.

[9] M. Teichmann and B. Mishra, "Reactive robotics I: Reactive grasping with a modified gripper and multi-fingererd hands," *International Journal of Robotics Research*, vol. 19, no. 7, pp. 697–708, 2000.

[10] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng, "Reactive grasping using optical proximity sensors," in *IEEE International Conference on Robotics and Automation*, (Kobe, Japan), May 2009. To appear.

[11] R. Platt Jr., A. H. Fagg, and R. Gruppen, "Nullspace composition of control laws for grasping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Lausanne, Switzerland), pp. 1717–1723, 2002.

[12] T. Mouri, H. Kawasaki, and S. Ito, "Unknown object grasping strategy imitating human grasping reflex for anthropomorphic robot hand," *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, vol. 1, no. 1, pp. 1–11, 2007.

[13] P. Allen and K. Roberts, "Haptic object recognition using a multi-fingered dextrous hand," *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pp. 342–347 vol.1, May 1989.

[14] M. Huber and R. Grupen, "Robust finger gaits from closed-loop controllers," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1578–1584 vol.2, 2002.

[15] P. Allen, A. T. Miller, P. Oh, and B. Leibowitz, "Using tactile and visual sensing with a robotic hand," in *IEEE International Conference on Robotics and Automation*, (Albuquerque, New Mexico), pp. 677–681, Apr. 1997.

[16] B. J. Grzyb, E. Chinellato, A. Morales, , and A. P. del Pobil, "Robust grasping of 3D objects with stereo vision and tactile feedback," in *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, (Coimbra, Portugal), pp. 851 – 858, 2008.

[17] K. Imazeki and T. Maeno, "Hierarchical control method for manipulating/grasping tasks using multi-fingered robot hand," *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4, pp. 3686–3691 vol.3, Oct. 2003.

[18] F. A. Mussa-Ivaldi, S. F. Giszter, and E. Bizzi, "Linear combinations of primitives in vertebrate motor control," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 91, no. 16, pp. 7534–7538, 1994.

[19] M. Ferch and J. Zhang, "Learning cooperative grasping with the graph representation of a state-action space," *Robotics and Autonomous Systems*, vol. 38, pp. 183–195, 2002.

[20] R. Amit and M. J. Mataric, "Parametric primitives for motor representation and control," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2002*, pp. 863–868, 2002.

[21] K. Nagatani and S. Yuta, "Door-opening behavior of an autonomous mobile manipulator by sequence of action primitives," *Journal of Robotic Systems*, vol. 13, no. 11, pp. 709–721, 1996.

[22] D. Kragic, S. Crinier, D. Brunn, and H. Christensen, "Vision and tactile sensing for real world tasks," *IEEE International Conference on Robotics and Automation*, pp. 1545–1550, September 2003.