# 26. Motion for Manipulation Tasks

Oliver Brock, James Kuffner, Jing Xiao

This chapter serves as an introduction to Part D by giving an overview of motion generation and control strategies in the context of robotic manipulation tasks. Automatic control ranging from the abstract, high-level task specification down to fine-grained feedback at the task interface are considered. Some of the important issues include modeling of the interfaces between the robot and the environment at the different time scales of motion and incorporating sensing and feedback. Manipulation planning is introduced as an extension to the basic motion planning problem, which can be modeled as a hybrid system of continuous configuration spaces arising from the act of grasping and moving parts in the environment. The important example of assembly motion is discussed through the analysis of contact states and compliant motion control. Finally, methods aimed at integrating global planning with state feedback control are summarized.

## 26.1 Overview

Part D of this handbook is concerned with the interfaces that connect robots to their environment. We differentiate three such interfaces, depicted in Fig. 26.1. The first interface, between the robot and the computer, is primarily concerned with the automatic generation of motion for performing a task. The second interface is concerned with the physical interaction between the robot and the environment. This chapter relates both interfaces in the context of manipulation tasks. It is focused on automatic specification, planning, and execution of motion of the robot manipulator or the manipulated object to meet a task goal under the constraints of the physical environment. Chapters 27–29 address other important issues pertaining to the second interface, such as the physical characteristics of contacts between the robot and the environment, grasping (i. e., the interaction between the robot and the object for manipulation), and the interaction between cooperative manipulators. The third interface, described in Chaps. 30–33, lies between humans and robots. Key issues that need to be

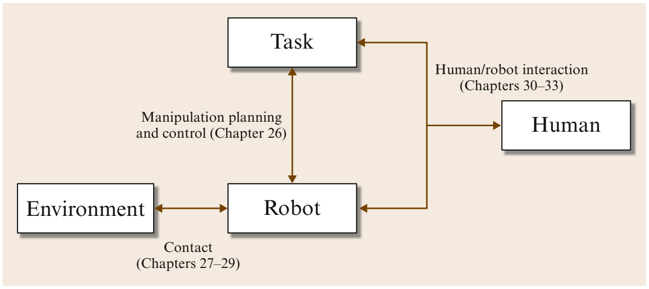
Task
Human/robot interaction (Chapters 30–33)
Human
Manipulation planning and control (Chapter 26)
Environment
Robot
Contact (Chapters 27–29)

**Fig. 26.1** Overview of Part D

addressed involve displaying appropriate sensor information to humans, or permitting humans to interactively specify the task or motion a robot is supposed to perform.

Manipulation refers to the process of moving or rearranging objects in the environment [26.1]. To perform a *manipulation task*, a robot establishes physical contact with objects in the environment and subsequently moves these objects by exerting forces and moments. The object to be manipulated can be large or small, and may serve various purposes in relation to the manipulation task (e.g., flipping a switch, opening a door, polishing a surface). In the context of automated assembly and industrial manipulation, the object to be manipulated is often referred to as a *part*. The *end-effector* generally refers to the link of the *manipulator* that makes contact and applies forces to the *part*. The *end-effector* is so named as it is often the end link in a serial kinematic chain. However, complex manipulation tasks may require the simultaneous exertion of multiple forces and moments at different contact points as well as the execution of sequences of force applications to several objects.

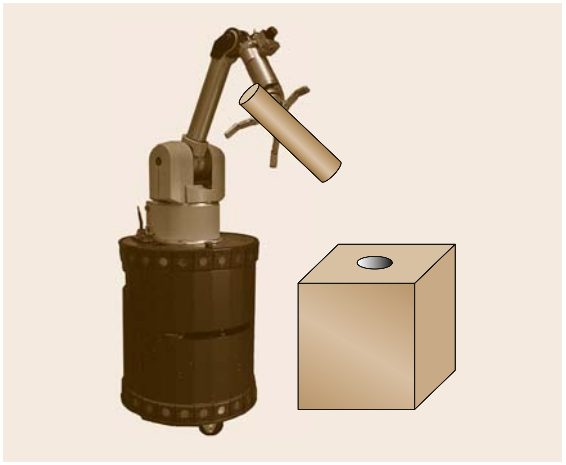

**Fig. 26.2** A mobile manipulator inserting a peg into a hole

To understand the challenge of generating the motion required for the execution of a manipulation task, we will consider the classic example of inserting a peg into a hole. The mobile manipulator shown in Fig. 26.2 is commanded to perform the peg insertion *manipulation task*, whose general problem structure arises in a number of contexts such as compliant assembly. For simplicity, we will only consider the *transfer motion* (see Sect. 26.3) and assume that the robot has already established a stable grasp (see Chap. 28) and is holding the peg. If the clearance between the peg and the hole is very small, the peg may not be able to enter the hole easily, and often some contact occurs. In order to guide the peg successfully into the hole, the robot has to deal with the possible contact states between the peg and the hole (Sect. 26.4) and select an appropriate sequence of control strategies that will result in successful completion of the task.

The peg-and-hole example illustrates that successful robotic manipulation involves dealing with models of object geometry and contact in the presence of uncertainty. There are various kinds of uncertainty, including modeling, actuation, and sensing uncertainty. To plan motion strategies in the presence of uncertainty poses difficult computational challenges. Historically, motion planning for manipulation has been divided into *gross motion planning* and *fine motion planning*. The former involves reasoning about the manipulator motion on the macro scale and considers its overall global movement strategy, while the latter considers how to deal with uncertainty to accomplish a task requiring high precision robustly. The primary reason for the historical division stems from attempts to simplify the various aspects of the problem into something computationally tractable. Assuming that the uncertainty in actuator position, object models, and environment obstacle shapes can be bounded, gross motion planning can be formulated primarily in terms of geometry (Sect. 26.3). As the robot or the manipulated object makes contact with the environment, such as in the example of the insertion task, fine motion planning techniques are employed to take into account contact geometry, forces, friction, and uncertainty effectively (see *Mason*'s book [26.1] and [26.2, 3] for an overview and historical perspective on fine motion strategies for manipulation tasks).

This peg-and-hole insertion example also illustrates that motion in the context of manipulation tasks is subject to *constraints*. These constraints have to be maintained to perform the task successfully. The specific constraints depend on the type of manipulation task, but they may include contact constraints, position, and force constraints for the point at which the manipulator makes contact with the environment, kinematic and dynamic constraints imposed by the mechanism and its actuation capabilities, posture constraints that specify behavior to be performed in addition to the manipulation task, reactive obstacle avoidance in an unpredictably changing environment, and global motion constraints to ensure that a specific goal location is attained. These motion constraints are imposed by the task, by the kinematics of the mechanism performing the task, by the actuation capabilities of the mechanism, and by the environment.

To ensure that constraints are satisfied in spite of uncertainty, sensory information (see Part C – Sensing and Perception) is considered in the context of feedback control loops (Chaps. 6 and 35). Depending on the type of constraint, this feedback has to be considered at various time scales. For example, the exertion of a constant force on an object in the environment requires high-frequency feedback at rates up to 1000 Hz. At the other end of the time scale, we consider changes to the global connectivity of the environment. These changes, resulting from opening doors or moving obstacles, for example, occur relatively slowly or infrequently and feedback about these changes only has to be considered a few times per second. Figure 26.3 graphically illustrates a task-dependent ordering of motion constraints encountered in manipulation tasks and their associated feedback requirements.

This chapter is concerned with algorithms that generate motion for manipulation tasks. Previous chapters already discussed planning algorithms (Chap. 5) and control methods (Chap. 6). While in these two former chapters the focus was on specific algorithmic techniques for robot motion, the current chapter is focused on a specific application, namely robotic manipulation. This application dictates motion constraints and their feedback requirements that will have to be satisfied during the motion for a manipulation task. In contrast, the methods discussed in the two aforementioned chapters each only address a subset of the motion constraints, as indicated in Fig. 26.3.

This chapter first discusses task-level control in Sect. 26.2. Task-level control describes a set of techniques to control a robotic mechanism in terms of its contact points with the environment. These contact points are controlled to perform motion that accomplishes a specific manipulation task. This means that, instead of directly controlling the mechanism, task-relevant points on the mechanism, so-called *operational points*, are controlled. This indirect control provides an intuitive means to specify desired motion for manipulation tasks.

Section 26.3 gives an overview of the configuration space formalization for manipulation planning, and how it can be cast as a classical motion planning problem (Chap. 5). Through this formalism, we can gain an understanding and intuition regarding how the geometry and relative positions of the manipulator, the manipulated objects, and the obstacles in the workspace determine both the geometry and topological structure of the manipulation configuration space. As we shall see, there are often an infinite number of possible ways to accomplish a manipulation task. Thus, the challenge in manipulation planning is to develop methods that deal effectively with the combinatorics of the search over all possible motions.

Section 26.4 addresses planning motion for assembly tasks, for which the peg-in-hole insertion is a typical example. The focus will be on motion constrained by contact, called *compliant motion*. There are two broad classes of assembly strategies involving compliant motion; one class is through special mechanism or control, called *passive compliance*, and the other is through active reasoning of contact states, called *active compliant motion*. Fine motion planning is discussed within the class of active compliant motion.

Methods from task-level control and manipulation planning address aspects of motion in a complementary fashion. While control methods are able to satisfy the feedback requirements of manipulation tasks, they often do not account for the overall global progress in accomplishing the manipulation task. Manipulation planners, on the other hand, reason about the global nature of the task at the high level, but are often computation-

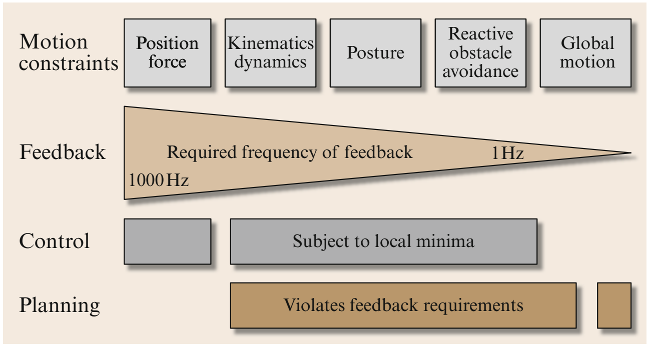

**Fig. 26.3** Examples of motion constraints encountered in manipulation tasks and their feedback requirements

ally too expensive to deal with uncertainty or satisfy the high-frequency feedback requirements of manipulation tasks. A robust and skillful manipulator must satisfy all feedback requirements while at the same time ensuring the accomplishment of the manipulation task. Section 26.5 discusses various techniques that offer a unified view of planning and control for manipulation tasks.

## 26.2 Task-Level Control

To perform a manipulation task, a robot has to establish contact with the environment. Through contact points the robot is able to apply forces and moments to objects. By controlling the position and velocity of the contact points as well as the forces acting at them, the robot causes the desired motion of objects, thereby performing the manipulation task. The programming of such a task is most conveniently accomplished by directly specifying positions, velocities, and forces at the contact points, rather than by specifying the joint positions and velocities required to achieve them.

Consider the manipulation task of serving a cup of water shown in Fig. 26.2. This task can easily be specified by providing a trajectory for the cup's motion. To determine a joint space trajectory that achieves the same motion, however, would be much more complex. One would have to rely on inverse kinematics (Sect. 1.9), which can be computationally challenging. More importantly, the task constraints imposed on the cup's motion do not uniquely specify the cup's trajectory. For example, while the cup is not allowed to tilt, it can be delivered to the goal location in any vertical orientation. Such task constraints can easily be specified in terms of the object motion – at the task level – but would be very difficult to characterize in terms of joint trajectories. Operational space control is therefore a natural choice for performing manipulation tasks. We will see in Sect. 26.2.3 that the operational space framework provides important additional advantages over joint space control in the context of redundant manipulators.

The advantages associated with task-level control come at the cost of depending on the manipulator Jacobian. For singular configurations of the manipulator (see Chap. 3), the manipulator Jacobian is not well defined. In these configurations, task-level control of a manipulator becomes unstable. Special care has to be applied to prevent the manipulator from entering these configurations, or to resort to specialized controllers in the proximity of these configurations. Most commonly, the manipulator is controlled so as to avoid singular configurations.

### 26.2.1 Operational Space Control

Task-level control – also called operational space control [26.4,5] (Sect. 6.2) – specifies the behavior of a robot in terms of operational points rather than joint positions and velocities. An operational point is an arbitrary point on the robot that is required to perform a particular motion or to exert a specified force to accomplish a manipulation task. In a serial-chain manipulator arm the operational point is most commonly chosen to coincide with the end-effector. More complex, branching mechanisms, such as humanoid robots, but also redundant serial-chain mechanisms, can have several operational points. To specify positions, velocities, accelerations, forces, and moments at the operational point, it is convenient to define a coordinate system with an origin that coincides with the operation point. This coordinate frame is called the operational frame. The orientation of the frame should be chosen in accordance with the task. For now we will ignore the orientation, however, and only consider tasks that require positioning of the operational point. In Sect. 26.2.2, we will consider more general tasks, combining position and force control.

Let us consider an operational point $\boldsymbol{x}$ on a robot. The relationship between the joint velocities $\dot{\boldsymbol{q}}$ and the velocity of the operational point, $\dot{\boldsymbol{x}}$, is given by the Jacobian matrix of the mechanism (Sect. 1.8) at the operational point $\boldsymbol{x}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_x(\boldsymbol{q})\dot{\boldsymbol{q}} \ . \tag{26.1}$$

Note that the Jacobian $\boldsymbol{J}$ depends on the placement of the operational point $\boldsymbol{x}$, and on the current configuration $\boldsymbol{q}$ of the robot. For simplicity, we will omit this dependency from our notation and just write $\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}$. From (26.1) we can derive an expression for the instantaneous torques $\boldsymbol{\tau}$ acting at the joints when the forces and moments described by $\boldsymbol{F}$ is applied at the operational point $\boldsymbol{x}$:

$$\boldsymbol{\tau} = \boldsymbol{J}^\top(\boldsymbol{q})\boldsymbol{F}_x \ . \tag{26.2}$$

The vector $\boldsymbol{F}$ captures the task to be performed at operational point $\boldsymbol{x}$. If the task specifies the complete position and orientation of the operational frame, $\boldsymbol{F}$ will be given by $\boldsymbol{F} = (f_x, f_y, f_z, u_x, u_y, u_z)$, where $f$ and $u$ designate forces along and moments about the respective axes. (For easy of presentation, we will refer to $\boldsymbol{F}$ as a force vector, even if it describes forces and moments.) The execution of such a task requires a manipulator with at least six degrees of freedom. If a task does not specify one of the components of $\boldsymbol{F}$, this component can be omitted and the corresponding column of the Jacobian matrix is dropped (or a manipulator with fewer than six degrees of freedom can be used to accomplish the task). For example, dropping $u_z$ would indicate that the orientation about the $z$-axis is not specified by the task. The motion behavior in the dimensions of the task space that have been dropped is unspecified and the robot will float in these dimensions [26.4]. In Sect. 26.2.3 we will discuss how additional behavior can be performed in these unspecified dimensions of the task space.

Similarly to joint space control, we have to account for manipulator dynamics to achieve acceptable performance in operational space control. Using (26.1), we can project the joint space dynamics into operational space to obtain

$$\boldsymbol{F_x} = \boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{\eta}(\boldsymbol{q}) , \tag{26.3}$$

where $\boldsymbol{\Lambda}$ is the operational space inertial matrix, $\boldsymbol{\Gamma}$ captures centrifugal and Coriolis forces in operational space, and $\boldsymbol{\eta}$ compensates for gravity forces. This equation pertains to a particular operational point. Intuitively speaking, the operational space inertia matrix $\boldsymbol{\Lambda}$ captures the resistance of the operational point to acceleration along and about different axes. More details about operational space control and its relationship to joint space control are provided in Sect. 6.2 (motion control, joint space versus operational space control).

## 26.2.2 Combined Force and Position Control

Let us consider the example task of controlling the motion of a peg inside a hole. We assume the peg is already inserted into the hole and rigidly connected to the robot's end-effector. We attach the operational frame to a point on the peg so that its $z$-axis is aligned with the desired direction of motion of the peg inside the hole. To perform this task, the robot has to control the position of the peg along the $z$-axis of the operational frame and it also has to control contact forces between the peg and the hole to avoid jamming. Thus, the task of moving a peg inside a hole requires that position and force control be seamlessly integrated.

To address contact forces within the operational space framework, we rewrite (26.3) as

$$\boldsymbol{F_x} = \boldsymbol{F}_{\mathrm{c}} + \boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{F}_{\mathrm{m}} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{\eta}(\boldsymbol{q}) , \tag{26.4}$$

where $\boldsymbol{F}_{\mathrm{c}}$ represents the contact forces acting at the end-effector [26.4]. We can replace $\ddot{\boldsymbol{x}}$ with $\boldsymbol{F}_{\mathrm{m}}$ because $\ddot{\boldsymbol{x}}$ is acting on a dynamically decoupled system, i. e., a system that acts like a unit point mass. We can now control forces and motion in the operational frame by selecting the following control structure:

$$\boldsymbol{F_x} = \boldsymbol{F}_{\mathrm{m}} + \boldsymbol{F}_{\mathrm{c}} , \tag{26.5}$$

where

$$\boldsymbol{F}_{\mathrm{m}} = \boldsymbol{\Lambda}(\boldsymbol{q})\,\Omega\,\boldsymbol{F}'_{\mathrm{m}} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{\eta}(\boldsymbol{q}) , \tag{26.6}$$

$$\boldsymbol{F}_{\mathrm{c}} = \boldsymbol{\Lambda}(\boldsymbol{q})\,\bar{\Omega}\,\boldsymbol{F}'_{\mathrm{c}} , \tag{26.7}$$

where $\Omega$ and $\bar{\Omega}$ represent complementary task specification matrices [26.4] that determine along which directions the end-effector is position controlled and along which it is force controlled. By selecting $\Omega$ appropriately, the combination of position and force control can be tailored to the task. In its simplest form, $\Omega$ and $\bar{\Omega} = I - \Omega$ are diagonal matrices. The $i$-th diagonal entry of $\Omega$ is 1 if the $i$-th operational coordinate of the end-effector is position controlled and 0 if it is force controlled. In this simplest case, positions and forces are controlled along axes of the same coordinate frame. The concept of task specification matrix can be extended to include differently oriented coordinate frames for position and force control [26.4].

Once the force $\boldsymbol{F_x}$ has been computed as given by (26.5), the corresponding joint torques used to control the robot are computed based on (26.1).

## 26.2.3 Operational Space Control of Redundant Mechanisms

The full expressiveness of task-level control comes to bear in the context of redundant manipulators. A manipulator is considered redundant with respect to the task it is performing if has more degrees of freedom than required by the task. For example, the task of holding a cup of water only specifies two degrees of freedom, namely the two rotations about the axes spanning the horizontal plane – this task has two degrees of freedom. The mobile manipulator shown in Fig. 26.2 has ten degrees of freedom, leaving eight redundant degrees of freedom with respect to the task.

The operational space framework for task-level control of redundant manipulators decomposes the overall motion behavior into two components. The first component is given by the task, specified in terms of forces and moments, $\boldsymbol{F}_{\text{task}}$, acting at an operational point. This vector $\boldsymbol{F}$ is translated into a joint torque based on (26.2): $\boldsymbol{\tau} = \boldsymbol{J}^{\top}\boldsymbol{F}_{\text{task}}$. For a redundant manipulator, however, the torque vector $\boldsymbol{\tau}$ is not uniquely specified and we can select from a set of task-consistent torque vectors. The operational space framework performs this selection by considering a secondary task, the so-called posture behavior, making up the second component of the overall motion behavior. Posture behavior can be specified by an arbitrary torque vector $\boldsymbol{\tau}_{\text{posture}}$. To ensure that this additional torque does not affect the task behavior ($\boldsymbol{F}_{\text{task}}$), the torque $\boldsymbol{\tau}_{\text{posture}}$ is projected into the null space $\boldsymbol{N}$ [26.6] of the task Jacobian $\boldsymbol{J}$. The null space of the Jacobian is the space orthogonal to the one spanned by $\boldsymbol{J}$ and will be of rank $N_J - k$, where $N_J$ is the number of degrees of freedom of the manipulator and $k = \text{rank}(\boldsymbol{J})$. The torque $\boldsymbol{N}^{\top}\boldsymbol{\tau}_{\text{posture}}$ resulting from the null space projection is task consistent, i. e., it is guaranteed not to affect on the behavior of the operational point. Since the vector $\boldsymbol{N}^{\top}\boldsymbol{\tau}_{\text{posture}}$ may not lie entirely within the null space of $\boldsymbol{J}$, however, the execution of the posture behavior cannot be guaranteed.

The operational space task ($\boldsymbol{J}^{\top}\boldsymbol{F}_{\text{task}}$) and the posture behavior ($\boldsymbol{N}^{\top}\boldsymbol{\tau}_{\text{posture}}$) are combined to obtain the general expression for the torque-level decomposition of the overall motion behavior:

$$\boldsymbol{\tau} = \boldsymbol{J}^{\top}\boldsymbol{F}_{\text{task}} + \boldsymbol{N}^{\top}\boldsymbol{\tau}_{\text{posture}} \,. \tag{26.8}$$

The null-space projection $\boldsymbol{N}$ associated with the task Jacobian $\boldsymbol{J}$ can be obtained by

$$\boldsymbol{N} = \boldsymbol{I} - \bar{\boldsymbol{J}}\boldsymbol{J} \,, \tag{26.9}$$

where $\boldsymbol{I}$ is the identity matrix, and $\bar{\boldsymbol{J}}$ is the dynamically consistent generalized inverse of $\boldsymbol{J}$, given by

$$\bar{\boldsymbol{J}} = \boldsymbol{H}^{-1}\boldsymbol{J}^{\top}\boldsymbol{\Lambda} \,, \tag{26.10}$$

where $\boldsymbol{H}$ is the joint space inertia matrix and $\boldsymbol{\Lambda}$ is the operational space inertia matrix for the operational point for which $\boldsymbol{J}$ is defined. The use of this specific inverse to compute the null-space projection results in the selection of the task-consistent torque vector that minimizes kinetic energy.

### 26.2.4 Combining Mobility and Manipulation

The operational space framework does not distinguish between degrees of freedom used for mobility and those used for manipulation. The end-effector-centric perspective of this approach considers all degrees of freedom to be in service to position and move the end-effector. An explicit coordination of manipulation and mobility for task-level control is not necessary.

A computationally efficient approach for coordinating manipulation and mobility was introduced in [26.7]. A similar, schema-based coordination of manipulation and mobility in the presence of dynamic obstacles was demonstrated in [26.8]. This approach is based on (26.2) and projects the forces that are derived from various schemas into the configuration space of the robot to determine its motion. Another coordination approach for manipulation and mobility considers a given end-effector path and generates a path of the base that maintains manipulability criteria [26.9]. This underlying representation of mobile manipulators can also be used to generate obstacle avoidance behavior of the base for a given end-effector path [26.10].

The coordination of a nonholonomic mobility platform and a cart-pushing task is addressed in [26.11]. In [26.12], an analysis of the task space for a mobile base with two manipulators is presented. Such a system with two manipulator arms can be viewed as a branching kinematic chain. Section 26.6 discusses extensions to the operational space framework to such branching mechanisms.

### 26.2.5 Combining Multiple Task Behaviors

The concept of decomposing a redundant robot's overall motion into task and posture behavior can be generalized to an arbitrary number of behaviors [26.13, 14]. Assume a set of $n$ tasks $T_i$, where $T_i$ has higher priority than $T_j$ if $i < j$. Every task $T_i$ is associated with a force vector $\boldsymbol{F}_i$ and the corresponding joint torque $\boldsymbol{\tau}_i$. These tasks can be performed simultaneously by projecting the joint torque associated with each task into the combined null space of all tasks with higher priority, just as task and posture behavior were combined in the previous section. The null-space projection ensures that the execution of task $i$ does not affect the execution of tasks with higher priority, given by the set $\text{prec}(i)$.

Given a task $i$, the joint torque consistent with the set of higher priority tasks $\text{prec}(i)$ can be obtained by projecting $\boldsymbol{\tau}_i$ into the combined null space of $\text{prec}(i)$,

$$\boldsymbol{\tau}_{i|\text{prec}(i)} = \boldsymbol{N}^{\top}_{\text{prec}(i)}\boldsymbol{\tau}_i \,, \tag{26.11}$$

where the combined null space $\boldsymbol{N}^{\top}_{\text{prec}(i)}$ is computed as

$$\boldsymbol{N}_{\text{prec}(n)} = \boldsymbol{N}_{n-1}\boldsymbol{N}_{n-2}\cdots\boldsymbol{N}_1 \,. \tag{26.12}$$

We say that $\boldsymbol{\tau}_{i|\text{prec}(i)}$ is the torque for task $i$ given the preceding tasks prec($i$). Applying this torque to the robot will not affect the execution of higher-priority tasks. The original torque vector $\boldsymbol{\tau}_i$ can be computed based on (26.2).

Given the ability to compute a task torque that is consistent with preceding tasks we are able to combine an arbitrary number of tasks into a single motion behavior:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_1 + \boldsymbol{\tau}_{2|\text{prec}(2)} + \boldsymbol{\tau}_{3|\text{prec}(3)} + \cdots \boldsymbol{\tau}_{n|\text{prec}(n)} , \tag{26.13}$$

where $\boldsymbol{\tau}$ is the torque used to control the robot.

Note that $\boldsymbol{\tau}_1$ is not projected into any null space, since it is the task with the highest priority. Task 1 is therefore performed in the full space defined by the robot's kinematics. If the robot has $N$ degrees of freedom, this space has $N$ dimensions. As the number of tasks performed on the robot increases, the null-space projections $\boldsymbol{N}_{\text{prec}(i)}$ will project the $N$-dimensional torque vectors associated with lower-priority tasks into a subspace of decreasing dimension. Eventually, the null space will be reduced to the trivial case, when all torque vectors are projected onto the null vector. This prevents lower-priority tasks from being executed. The dimensionality of the null space of tasks prec($i$) can be computed by counting the nonzero singular values [26.6] of $\boldsymbol{N}_{\text{prec}(i)}$, permitting a programmatic assessment of task feasibility.

If the execution of a particular task has to be guaranteed, it should be associated with task 1 in (26.13). Therefore, the highest-priority task is generally used for hard constraints that under no circumstances should be violated. These constraints may include contact constraints, joint limits, and balancing in the case of humanoid robots, for example (Chap. 56).

The projection of (26.11) maps the torque $\boldsymbol{\tau}_i$ of task $i$ into the null space of all preceding tasks. This projection also scales the vector, which may significantly impair the execution of task $i$. This effect can be reduced by scaling the desired accelerations $\ddot{\boldsymbol{x}}$ at the operational point with the task-consistent inertia matrix $\boldsymbol{\Lambda}_{i|\text{prec}(i)}$, given by

$$\boldsymbol{\Lambda}_{i|\text{prec}(i)} = \left(\boldsymbol{J}_{i|\text{prec}(i)}\boldsymbol{H}^{-1}\boldsymbol{J}^{\top}_{i|\text{prec}(i)}\right)^{-1} , \tag{26.14}$$

where $\boldsymbol{H}$ is the joint space inertia matrix and

$$\boldsymbol{J}_{i|\text{prec}(i)} = \boldsymbol{J}_i\boldsymbol{N}_{\text{prec}(i)} \tag{26.15}$$

is the task Jacobian consistent with all preceding tasks. This Jacobian projects operational space forces into joint torques that do not cause any acceleration at the operational points of tasks in prec($i$). The task-consistent inertia matrix $\boldsymbol{\Lambda}_{i|\text{prec}(i)}$ then captures the inertia perceived at the operational point when its motion is limited to the subspace consistent with all preceding tasks.

Using the task-consistent inertia matrix, we can obtain the task-consistent operational space dynamics by projecting the joint space dynamics into the space defined by the dynamically consistent inverse of $\boldsymbol{J}_{i|\text{prec}(i)}$. Inverting (26.2) and replacing $\boldsymbol{\tau}$ based on (6.1) we obtain

$$\bar{\boldsymbol{J}}^{\top}_{i|\text{prec}(i)}\left(\boldsymbol{H}(\boldsymbol{q}) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{\tau}_g(\boldsymbol{q}) = \boldsymbol{\tau}_{k|\text{prec}(i)}\right) , \tag{26.16}$$

which yields the task-consistent equivalent to (26.3):

$$\boldsymbol{F}_{i|\text{prec}(i)} = \boldsymbol{\Lambda}_{i|\text{prec}(i)}\ddot{\boldsymbol{x}} + \boldsymbol{\Gamma}_{i|\text{prec}(i)}\dot{\boldsymbol{x}} + \boldsymbol{\eta}_{i|\text{prec}(i)} , \tag{26.17}$$

where $\boldsymbol{\Lambda}_{i|\text{prec}(i)}$ is the task-consistent operational space inertial matrix, $\boldsymbol{\Gamma}_{i|\text{prec}(i)}$ captures centrifugal and Coriolis forces in operational space, and $\boldsymbol{\eta}_{i|\text{prec}(i)}$ compensates for gravity forces. Equation (26.17) can be used to control a task at a particular operational point in a task-consistent manner, i. e., without causing acceleration at the operational points associated with the tasks in prec($i$). Note that in practice the terms $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and $\boldsymbol{\tau}_g(\boldsymbol{q})$ can be computed in the joint space; they only need to be accounted for once in the case of multiple tasks.

Referring back to Fig. 26.3, we can analyze the effectiveness of the operational space framework. The control of operational points directly addresses force and position constraints on motion in manipulation tasks. Additional behaviors with lower priority than the task can be used to ensure the maintenance of kinematic and dynamic constraints, implement posture behavior, or perform reactive obstacle avoidance based on artificial potential fields [26.15]. The control of all of these aspects of manipulation tasks is computationally efficient. As a consequence, the operational space framework is able to maintain these motion constraints while satisfying feedback frequency requirements. However, the operational space framework does not include the consideration of global motion constraints, such as those imposed by the connectivity of the work space. Therefore, the operational space framework is susceptible to local minima, similarly to the artificial potential field method. Such local minima can occur for all of the aforementioned motion constraints and would prevent the robot from accomplishing its task. To overcome this problem, planning methods have to be applied (see Chap. 5 and Sect. 26.3).

The control basis approach [26.16] provides an alternative, compact formalism for combining multiple behaviors. Each individual behavior is described by a controller, represented by a potential function $\phi \in \Phi$, sensors $\sigma \in \Sigma$, and effectors $\tau \in \Upsilon$. A particular controller $\phi$ can be parameterized with sensors and effectors. A particular instance of a behavior can be written as $\phi|_{\tau}^{\sigma}$. These controllers can be combined using null-space projections, similar to (26.13). In the control basis framework, (26.13) would be written more compactly as

$$\phi_n \triangleleft \cdots \triangleleft \phi_3 \triangleleft \phi_2 \triangleleft \phi_1 \ , \tag{26.18}$$

where we abbreviate $\phi_i|_{\tau_i}^{\sigma_i}$ with $\phi_i$. The relation $\phi_i \triangleleft \phi_j$ means that $\phi_i$ is projected into the null space of $\phi_j$. We say that $\phi_i$ is executed *subject to* $\phi_j$. If more than two behaviors are combined, as shown in (26.18), each controller $\phi_i$ is projected into the combined null space of all superior tasks.

The control basis represents a general framework for combining controllers and is not limited to the notion of task-level control. In addition to the aforementioned formalism for representing combinations of controllers, the control basis also encompasses discrete structures that define transitions between combinations of controllers. Such transitions can be specified by the user or learned with techniques from reinforcement learning [26.16]. The control basis can compose multiple task-level controllers to implement manipulation tasks that cannot be described in terms of a single task-level controller.

## 26.3 Manipulation Planning

In order to appreciate the computational challenges inherent to gross motion planning for manipulation, it is useful to adopt a mathematical formalism for analysis. The chapter on planning (Chap. 5) introduced the *configuration space* ($\mathcal{C}$*-space*), which represents the space of allowable transformations of the robot. In general, the set of all allowable transformations forms an $n$-dimensional *manifold*, in which $n$ is the total number of degrees of freedom (DOF) of the mechanical system. For a robotic manipulator, the configuration space is defined by its kinematic structure (see Chap. 1). One common class of manipulators is *serial-chain* manipulators, which consist of a series of revolute or prismatic joints connected to form a linear kinematic chain. The discussion of manipulation planning in this section focuses on configuration spaces arising from single serial-chain manipulators. However, the mathematical formulation generalizes to more complex robots such as humanoids (Fig. 26.4; Chap. 56), whose upper body consists of two cooperating manipulator arms attached to a movable base (see Chap. 29). The model considered here greatly simplifies the problems of grasping, stability, friction, mechanics, and uncertainties and instead focuses on the geometric aspects. In this section, we will consider a simple example of a manipulation task that involves a classic pick-and-place operation. Section 26.4 further considers grasping and contact states in the context of assembly operations, and Sect. 26.5 gives an overview of techniques to unify feedback control and planning for more complex manipulation tasks.

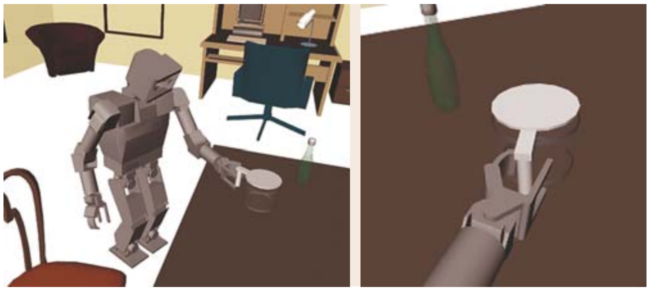
**Fig. 26.4** Simulation of the H6 humanoid robot manipulating an object

### 26.3.1 Configuration Space Formalism

The basic motion planning problem is defined as finding a path in the configuration space connecting a start and a goal configuration while avoiding static obstacles in the environment (see Chap. 5). Manipulation planning involves additional complexity because manipulation tasks fundamentally require contact with objects in the environment. When objects are grasped, regrasped, and moved, the structure and topology of the free configuration space can change drastically. However, manipulation planning shares the same basic theoretical foundations and computational complexity as classical path planning, such as PSPACE-hardness (see Chap. 5).

To be more precise, we will utilize a formalization for manipulation planning that is primarily due to *Alami* et al. [26.17], and adapted in *LaValle*'s book [26.18]. Our goal is to gain understanding and intuition regarding the search space for manipulation tasks and the different situations that can arise. Throughout this discussion, bear in mind that the geometry and relative positions of the manipulator, the manipulated objects, and the obstacles in the workspace determine both the geometry and topological structure of the free configuration space. Within this framework, manipulation planning can be intuitively placed in the context of planning for *hybrid systems* [26.19]. Hybrid systems involve a mixture of continuous variables and discrete *modes*, whose transitions are defined by switching functions (for a discussion of hybrid systems in a planning context, see Chap. 7 of *LaValle*'s book [26.18]). In the case of manipulation planning, the effect of grasping, repositioning, and releasing objects in the environment corresponds to switching between different continuous configuration spaces.

Let us consider a simple example of a simple *pick-and-place manipulation task* that requires a robot manipulator $\mathcal{A}$ to reposition a single movable rigid object (part) $\mathcal{P}$ from its current location in the workspace to some desired goal location. Note that accomplishing the task only requires that the part $\mathcal{P}$ reach the goal location, without regard to precisely *how* the manipulator does so. Because the part cannot move by itself, it must either be transported by the robot or be placed at rest in some stable intermediate configuration. There are some configurations in which the robot is able to grasp and move the part to other locations in the environment. There are also forbidden configurations in which the robot or the part collides with obstacles in the environment. The solution of a manipulation planning problem consists of a sequence of subpaths for the robot moving alone or moving while grasping the part. The primary constraints on the allowable motions are:

1. The robot manipulator must not collide with any obstacles in the environment while reaching for, grasping, or regrasping the part.
2. The robot must adequately grasp the part while transporting it (for details on grasping and metrics for stable grasps see Chap. 28).
3. The robot manipulator and movable part must not collide with any obstacles in the environment while the part is being transported.
4. If the part is not being transported by the robot, it must be placed at rest in some stable intermediate placement.
5. The part must end up at the desired goal location in the workspace.

The two modes of operation give rise to the hybrid systems formulation: either the robot moves alone, or the robot moves while grasping the part. In the literature, motions of the robot holding the object at a fixed grasp are called *transfer paths*, and motions of the robot while the part stays at a stable placement are called *transit paths* [26.17, 20, 21]. Solving a manipulation planning problem involves searching for a connected sequence of *transit* and *transfer* paths for the manipulator that will accomplish the task. For an autonomous robot, the ideal is to utilize planning algorithms that will automatically produce correct sequences of transfer and transit paths separated by grasping and ungrasping operations.

### Admissible Configurations

Building on the notation introduced in Chap. 5, we define a robot manipulator $\mathcal{A}$ with $n$ degrees of freedom operating in a Euclidean *world* or *workspace*, $\mathcal{W} = \mathbb{R}^N$, in which $N = 2$ or $N = 3$. Let $\mathcal{C}^A$ represent the $n$-dimensional $\mathcal{C}$-space of $\mathcal{A}$, and $q^A$ be a configuration. $\mathcal{C}^A$ is called the *manipulator configuration space*. Let $\mathcal{P}$ denote a movable part, which is a solid rigid body modeled with geometric primitives. We will assume that $\mathcal{P}$ is allowed to undergo rigid-body transformations. This gives rise to a *part configuration space*, $\mathcal{C}^P = SE(2)$ or $\mathcal{C}^P = SE(3)$. Let $q^P \in \mathcal{C}^P$ denote a part configuration. The area or volume in the workspace occupied by the transformed part model is denoted by $\mathcal{P}(q^P)$.

We define the combined (robot and part) configuration space $\mathcal{C}$ as the Cartesian product

$$\mathcal{C} = \mathcal{C}^A \times \mathcal{C}^P \ ,$$

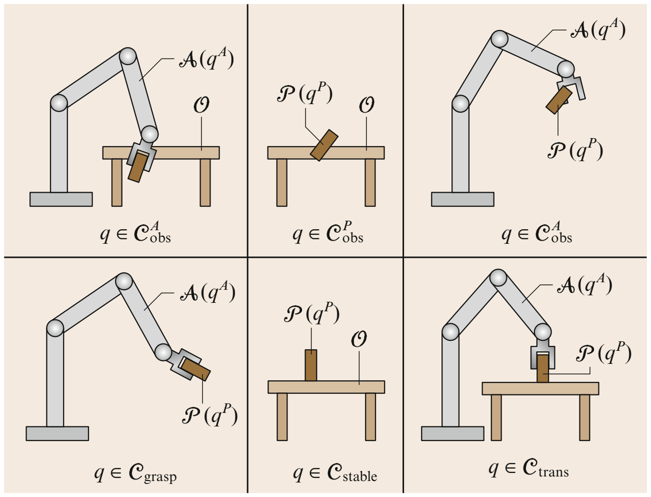
$\mathcal{A}(q^A)$
$\mathcal{O}$
$q \in \mathcal{C}^A_{\text{obs}}$
$\mathcal{P}(q^P)$
$\mathcal{O}$
$q \in \mathcal{C}^P_{\text{obs}}$
$\mathcal{A}(q^A)$
$\mathcal{P}(q^P)$
$q \in \mathcal{C}^A_{\text{obs}}$
$\mathcal{A}(q^A)$
$\mathcal{P}(q^P)$
$q \in \mathcal{C}_{\text{grasp}}$
$\mathcal{P}(q^P)$
$\mathcal{O}$
$q \in \mathcal{C}_{\text{stable}}$
$\mathcal{A}(q^A)$
$\mathcal{P}(q^P)$
$q \in \mathcal{C}_{\text{trans}}$

**Fig. 26.5** Examples of the various configuration space subsets defined for manipulation planning

in which each configuration $q \in \mathcal{C}$ is of the form $q = (q^A, q^P)$. Note that the set of admissible configurations for manipulation planning purposes is more restrictive than simply $\mathcal{C}_{\text{free}}$, the space free of collisions with obstacle regions as defined in Chap. 5. We must incorporate the constraints on the allowable motions by removing inadmissable configurations. Figure 26.5 illustrates examples of some of the important subsets of $\mathcal{C}$ for manipulation planning.

We start by removing all configurations that involve collisions with obstacles. Let the subset of configurations in which the manipulator collides with obstacles be given by

$$\mathcal{C}^A_{\text{obs}} = \{(q^A, q^P) \in \mathcal{C} | \mathcal{A}(q^A) \cap \mathcal{O} \neq \emptyset\} .$$

We also want to remove configurations for which the part collides with obstacles. However, we will allow the surface of the part to make contact with obstacle surfaces. This happens for example, when the part is resting on a shelf or table, or when a peg-shaped part is being inserted into a hole. Therefore, let

$$\mathcal{C}^P_{\text{obs}} = \{(q^A, q^P) \in \mathcal{C} | \text{int}(\mathcal{P}(q^P)) \cap \mathcal{O} \neq \emptyset\}$$

denote the open set for which $\mathcal{O}$ intersects the interior volume $\text{int}(\mathcal{P}(q^P))$ of the part for any configuration $q^P$. If the interior of the part penetrates $\mathcal{O}$, then clearly these configurations should be avoided.

We can now define $\mathcal{C} \setminus (\mathcal{C}^A_{\text{obs}} \cup \mathcal{C}^P_{\text{obs}})$, which is the set of all configurations in which both the robot and the part do not inappropriately collide with $\mathcal{O}$. Next consider the interaction between $\mathcal{A}$ and $\mathcal{P}$. The manipulator must be allowed to make contact with the surface of the part, but penetration is once again not allowed. Therefore, let all configurations in which the interior of the part overlaps with the robot geometry be defined as

$$\mathcal{C}^{PA}_{\text{obs}} = \{(q^A, q^P) \in \mathcal{C} | A(q^A) \cap \text{int}(\mathcal{P}(q^P)) \neq \emptyset\} .$$

Finally, we are able to define

$$\mathcal{C}_{\text{adm}} = \mathcal{C} \setminus (\mathcal{C}^A_{\text{obs}} \cup \mathcal{C}^P_{\text{obs}} \cup \mathcal{C}^{PA}_{\text{obs}}) ,$$

which represents the set of configurations remaining after subtracting away all unwanted configurations. We call this the set of *admissible configurations*.

### Stable and Grasped Configurations

By definition, all configurations in the set $\mathcal{C}_{\text{adm}}$ are free of penetrative collisions between the robot, the part, and obstacles in the environment. However, many of these configurations define the part as floating in space or on the verge of falling. We now consider two important subsets of $\mathcal{C}_{\text{adm}}$ that are used in manipulation planning (see Fig. 26.5). Let $\mathcal{C}^P_{\text{stable}} \subseteq \mathcal{C}^P$ denote the subset of *stable part configurations*, which are configurations at which the part can safely rest without any forces being applied by the manipulator. Examples of stable configurations include the part resting on a table or inserted into a larger assembly of other parts. The criteria for inclusion into the set of stable configurations depends on properties such as the part geometry, friction, mass distribution, and contacts with the environment. Our formalism does not consider these issues directly, but assumes that some method for evaluating the stability of a part configuration is available. Given $\mathcal{C}^P_{\text{stable}}$, we define $\mathcal{C}_{\text{stable}} \subseteq \mathcal{C}_{\text{adm}}$ to be the corresponding stable configurations of the part–robot system:

$$\mathcal{C}_{\text{stable}} = \{(q^A, q^P) \in \mathcal{C}_{\text{adm}} | q^P \in \mathcal{C}^P_{\text{stable}}\} .$$

The other important subset of $\mathcal{C}_{\text{adm}}$ is the set of all configurations in which the robot is grasping the part and is capable of manipulating it according to some defined criteria. Let $\mathcal{C}_{\text{grasp}} \subseteq \mathcal{C}_{\text{adm}}$ denote the set of *grasp configurations*. For every configuration, $(q^A, q^P) \in \mathcal{C}_{\text{grasp}}$, the manipulator touches the part, which implies that $\mathcal{A}(q^A) \cap \mathcal{P}(q^P) \neq \emptyset$. Just as before, penetration between the robot and part geometry is not allowed because $\mathcal{C}_{\text{grasp}} \subseteq \mathcal{C}_{\text{adm}}$. In general, many configurations at which $\mathcal{A}(q^A)$ contacts $\mathcal{P}(q^P)$ will not necessarily be in $\mathcal{C}_{\text{grasp}}$. The criteria for a configuration to lie in $\mathcal{C}_{\text{grasp}}$ depends on the particular characteristics of the manipulator, the part, and the contact surfaces between them. For example, a typical manipulator would not be able to pick up a part by making contact at only a single point. (For additional information, see Sect. 26.4 for contact state identification, Chap. 27, and grasping criteria, such as *force closure* models in Chaps. 28 and and 15).

For any robot and part configuration $q = (q^A, q^P) \in \mathcal{C}$ that we consider for manipulation planning we must always ensure that either $q \in \mathcal{C}_{\text{stable}}$ or $q \in \mathcal{C}_{\text{grasp}}$. We therefore define $\mathcal{C}_{\text{free}} = \mathcal{C}_{\text{stable}} \cup \mathcal{C}_{\text{grasp}}$, to reflect the subset of $\mathcal{C}_{\text{adm}}$ that is permissible for manipulation planning. In the context of hybrid systems, this gives rise to the two distinct *modes* for manipulation planning: the *transit mode* and the *transfer mode*. Recall that in the transit mode, the manipulator is not carrying the part (i. e., only the robot moves), which requires that $q \in \mathcal{C}_{\text{stable}}$. In the transfer mode, the manipulator carries the part (i. e., both the robot and the part move), which requires that $q \in \mathcal{C}_{\text{grasp}}$. Based on these conditions, the only way the mode can change is if $q \in \mathcal{C}_{\text{stable}} \cap \mathcal{C}_{\text{grasp}}$. When the manipulator is in these configurations, it has two available actions: (1) continue to grasp and move the part, or (2) release the part in its currently stable configuration. In all other configurations the mode remains unchanged. For convenience, we define $\mathcal{C}_{\text{trans}} = \mathcal{C}_{\text{stable}} \cap \mathcal{C}_{\text{grasp}}$ to denote the set of *transition configurations*, which are the places in which the mode may change and the robot may grasp or release the part.

#### Manipulation Planning Task Definition

Finally, the basic pick-and-place manipulation planning task can now be defined. An *initial part configuration*, $q^P_{\text{init}} \in \mathcal{C}^P_{\text{stable}}$, and a *goal part configuration*, $q^P_{\text{goal}} \in \mathcal{C}^P_{\text{stable}}$, are specified. Recall that this high-level task specification is only defined in terms of repositioning the part, without regard to how the manipulator actually accomplishes the task. Let $\mathcal{C}_{\text{init}} \subseteq \mathcal{C}_{\text{free}}$ be the set of all configurations in which the part configuration is $q^P_{\text{init}}$:

$$\mathcal{C}_{\text{init}} = \{(q^A, q^P) \in \mathcal{C}_{\text{free}} | q^P = q^P_{\text{init}}\} .$$

We define $\mathcal{C}_{\text{goal}} \subseteq \mathcal{C}_{\text{free}}$ similarly as the set of all configurations in which the part configuration is $q^P_{\text{goal}}$:

$$\mathcal{C}_{\text{goal}} = \{(q^A, q^P) \in \mathcal{C}_{\text{free}} | q^P = q^P_{\text{goal}}\} .$$

If the initial and final configurations of the manipulator are specified, then let $q_{\text{init}} = (q^A_{\text{init}}, q^P_{\text{init}}) \in \mathcal{C}_{\text{init}}$ and $q_{\text{goal}} = (q^A_{\text{goal}}, q^P_{\text{goal}}) \in \mathcal{C}_{\text{goal}}$. The objective of the planner is to compute a path $\tau$ such that

$$\tau : [0, 1] \mapsto \mathcal{C}_{\text{free}} ,$$
$$\tau(0) = q_{\text{init}};\ \tau(1) = q_{\text{goal}} .$$

If the initial and final positions of the manipulator are unspecified, we implicitly define the objective of the planner as computing a path $\tau : [0, 1] \mapsto \mathcal{C}_{\text{free}}$ such that $\tau(0) \in \mathcal{C}_{\text{init}}$ and $\tau(1) \in \mathcal{C}_{\text{goal}}$. In either case, a solution is an alternating sequence of transit paths and transfer paths, whose names follow from the mode. In between each transfer path, the part is placed in a stable intermediate configuration while the manipulator moves alone (transit path) in order to regrasp the part. This sequence continues until the part is ultimately placed to rest in its final goal configuration. This is depicted in Fig. 26.6.

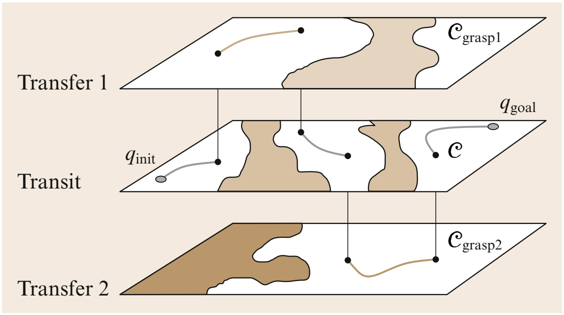

**Fig. 26.6** Manipulation planning involves searching for a sequence of transfer and transit paths in a set of hybrid continuous configuration spaces. In this example, transfer paths in different $\mathcal{C}$-spaces arise from different ways of rigidly grasping a part

## 26.3.2 Example of a Three-DOF Planar Manipulator

For manipulators with three or fewer degrees of freedom (DOF), we can construct visualizations of the $\mathcal{C}$-space in order to gain an intuition of the structure. Throughout this section, we will use the example of a three-DOF serial-chain manipulator whose end-effector operates a planar workspace ($\mathbb{R}^2$).

Figure 26.7 shows a simple redundant robot arm originally illustrated in [26.22]. The robot has three revolute joints with parallel axes and joint limits of $-\pi, \pi$ radians, providing three degrees of freedom. Since the end-effector of the robot can only reach positions in a plane, the arm is redundant in this plane and since it has only three DOF and limits imposed on the joint angles, $\mathcal{C}$ can be visualized as a cube with an edge length of $2\pi$. Generally, the $\mathcal{C}$-space manifold of a manipulator with $n$ revolute joints is homeomorphic to an $n$-dimensional *hypertorus* if no joint limits are present, and homeomorphic to an $n$-dimensional *hypercube* if joint limits are present.

Recall that obstacles in the workspace are mapped to regions in the $\mathcal{C}$-space called $\mathcal{C}$-obstacles (Chap. 5), which represent the set of all joint configurations of the robot that cause an intersection of the geometry of the robot and the obstacle

$$\mathcal{C}^A_{\text{obs}} = \{(q^A, q^P) \in \mathcal{C} | A(q^A) \cap \mathcal{O} \neq \emptyset\} .$$

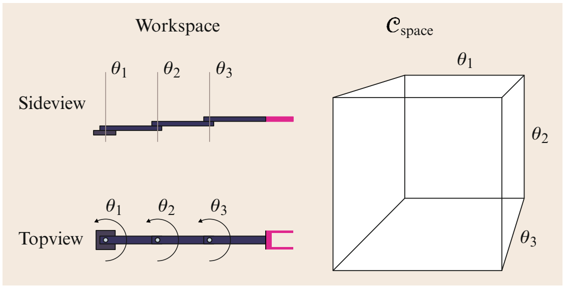

**Fig. 26.7** Planar 3 DOF manipulator

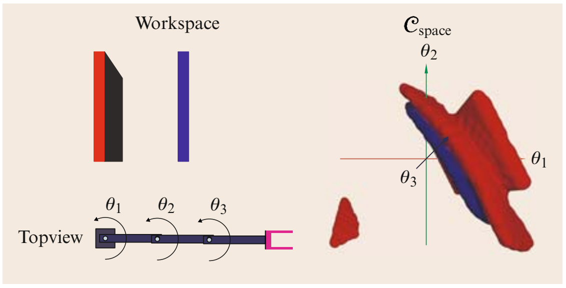
Workspace
$\mathcal{C}_{\text{space}}$
$\theta_2$
$\theta_1$
$\theta_3$
$\theta_1$ $\theta_2$ $\theta_3$
Topview

**Fig. 26.8** Planar 3 DOF manipulator in a workspace with box-shaped obstacles and associated configuration space obstacles

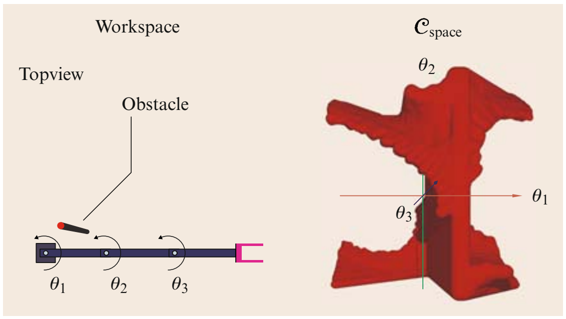
Workspace
$\mathcal{C}_{\text{space}}$
Topview
Obstacle
$\theta_2$
$\theta_1$
$\theta_3$
$\theta_1$ $\theta_2$ $\theta_3$

**Fig. 26.10** Workspace mapping to a configuration space with topologically disconnected components

In general, the structure of $\mathcal{C}^{A}_{\text{obs}}$ can be highly complex and involve multiple connected components of $\mathcal{C}$. An example visualization of a configuration space for the planar three-DOF manipulator corresponding to a workspace with two differently colored box-shaped obstacles is given in Fig. 26.8. The $\mathcal{C}$-obstacles are illustrated using the same color as the corresponding box-shaped obstacle in the workspace. The red, green, and blue coordinate axes correspond to the three joint angles $\theta_{\{1,2,3\}}$ of the manipulator, respectively.

Self-collisions induce special $\mathcal{C}$-obstacle regions that do not involve environment geometry, but only the geometric interference of the links of the robot itself. The robot shown in Fig. 26.7 is, by design, self-collision-free. Figure 26.9 shows a modified version of the manipulator, where self-collision is actually possible, and a visualization of the corresponding $\mathcal{C}$-obstacles.

$\mathcal{C}$-obstacles can cover large parts of the configuration space and even cause a complete disconnection between different components of $\mathcal{C}_{\text{free}}$. Figure 26.10 shows how the addition of only a small post as an obstacle can cause a wide range of configurations to produce collisions and disconnect regions of $\mathcal{C}_{\text{free}}$: the $\mathcal{C}$-obstacle corresponding to the post forms a *wall* of configurations parallel to the $\theta_2$–$\theta_3$-plane. This wall represents the range of values for $\theta_1$ for which the first link of the manipulator intersects with the post, splitting $\mathcal{C}_{\text{free}}$ into two disconnected components. Having disconnected components $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{C}_{\text{free}}$ means that there is no collision-free path that starts in $\mathcal{C}_1$ and ends in $\mathcal{C}_2$.

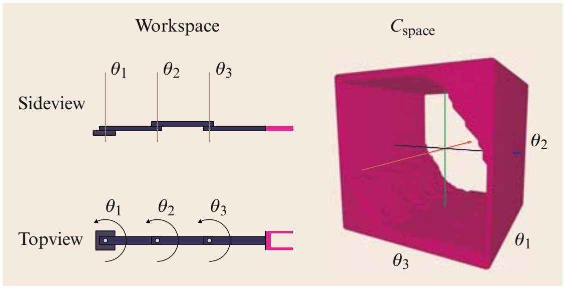

**Fig. 26.9** Non-planar 3 DOF manipulator and corresponding self-collision $\mathcal{C}$-obstacle

## 26.3.3 Inverse Kinematics Considerations

The typical formulation of a path planning problem requires a goal configuration $q_{\text{goal}}$ as input (see Chap. 5). For the case of manipulation planning, this configuration is usually computed from the desired workspace pose of the end-effector by an inverse kinematics (IK) solver (see Sect. 1.7). Apart from special cases, there currently exist no known analytical methods for solving the inverse kinematics of a general redundant mechanism (greater than six degrees of freedom). Iterative, numerical techniques (such as Jacobian-based methods or gradient-based optimization) are typically used to calculate solutions in this case. Note that the numerical computation of inverse kinematics solutions can suffer from poor performance or even nonconvergence, depending on the quality of the initial guess, the distribution of singularities in the $\mathcal{C}$-space or a combination of these effects (see Sect. 1.7).

Most IK solvers are only able to compute one configuration for a given end-effector pose. For a redundant manipulator, however, an infinite, continuous range of configurations that cause the end-effector to assume the desired pose usually exists. In the context of path planning, this means that the IK solver will select one out of an infinite number of configurations as $q_{\text{goal}}$. Note that in some cases, the selected $q_{\text{goal}}$ might not be collision-free, i. e., it is not part of $\mathcal{C}_{\text{free}}$. Such a configuration cannot be used as a goal for computing a global reaching strategy

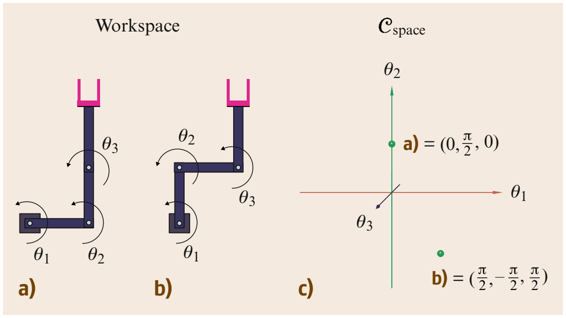
Workspace
$\mathcal{C}_{\text{space}}$
$\theta_1$
$\theta_2$
$\theta_3$
a) = $(0, \frac{\pi}{2}, 0)$
b) = $(\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2})$
a)
b)
c)

**Fig. 26.11** Two inverse kinematics solutions for the same end-effector pose of a planar 3 DOF arm

because no solution exists for the corresponding path planning query in the classical sense (Chap. 5).

Another possibility is that the selected $q_{\text{goal}}$ might be disconnected from $q_{\text{init}}$. In this case, $q_{\text{goal}}$ is unsuitable for planning despite the fact that it is a member of $\mathcal{C}_{\text{free}}$, because it is in a component of $\mathcal{C}_{\text{free}}$ that is disconnected from the component in which $q_{\text{init}}$ lies. Consider the planar manipulator shown in Fig. 26.7. Figure 26.11 shows two possible solutions for a certain end-effector pose. The corresponding configurations $q_a = (0, \frac{\pi}{2}, 0)$ and $q_b = (\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2})$ are marked by green spheres in the $\mathcal{C}$-space visualization. In this case, the IK solver might select one of the two solutions as $q_{\text{goal}}$. If the same obstacle as in Fig. 26.10 is added to the workspace, the two solutions will lie in disconnected components of $\mathcal{C}_{\text{free}}$, as shown in Fig. 26.12.

Let $q_{\text{init}} = (0, 0, 0)$ be the zero configuration, marked by the point of intersection of the $\mathcal{C}$-space axes in Fig. 26.12. Let the component of $\mathcal{C}_{\text{free}}$ in which it lies be denoted by $\mathcal{C}_1$ and the other component by $\mathcal{C}_2$. Since $q_b \in \mathcal{C}_2$ and $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$, no solution for the planning problem can exist if the IK solver selects $q_b$ as $q_{\text{goal}}$.

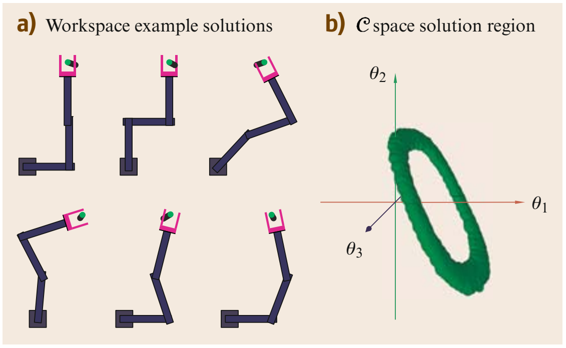
a) Workspace example solutions
b) $\mathcal{C}$ space solution region
$\theta_1$
$\theta_2$
$\theta_3$

**Fig. 26.13 (a)** Multiple arm configurations that allow the manipulator to grasp the cylinder and **(b)** $\mathcal{C}$-space visualization of the set of solution configurations

This means that every planning attempt with this $q_{\text{goal}}$ will invariably fail.

Since computing a complete representation of the connectivity of $\mathcal{C}$ is a problem of similar complexity to path planning itself, it is virtually impossible to detect a disconnection between $q_{\text{goal}}$ and $q_{\text{init}}$ before the planning attempt has failed.

Recall that the workspace goal for a classical manipulation planning problem is defined by a desired end-effector pose computed using inverse kinematics. Note however, that the purpose of a manipulation task may not require moving the end-effector to that particular location, but rather moving it to *any* feasible location which enables the desired manipulation task to be solved. Thus, it becomes important to devise computational methods and control schemes that are focused around the task, and allow some flexibility and freedom with regards to the end-effector pose and manipulator configuration. This is one of the motivations and advantages of

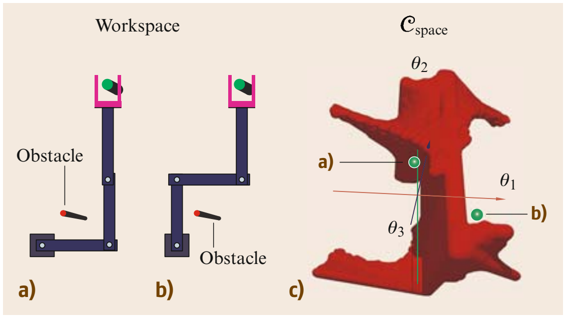

**Fig. 26.12** Two inverse kinematics solutions lying in disconnected components of $\mathcal{C}_{\text{free}}$ due to a small obstacle in the workspace

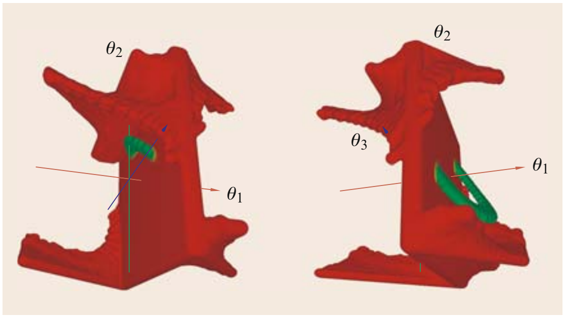

**Fig. 26.14** Two views of the set of goal configurations *(green)* for grasping a cylindrical part in a disconnected $\mathcal{C}$-space

task space or operational space control techniques (Sect. 6.2).

As an example, consider the task of grasping a cylindrical part with the planar three-DOF manipulator. Figure 26.11 shows two possible solution configurations yielding the same end-effector pose. The addition of obstacles to this workspace may cause one of the configurations to become disconnected from the zero configuration. These two configurations, however, may not be the only valid configurations that allow the manipulator to grasp the cylindrical part. Figure 26.13a shows several other configurations that allow the manipulator to grasp the part. By interpolating between these, an infinite number of different workspace poses that provide solutions to the grasping problem is generated. The inverse kinematics solutions associated with these workspace poses are contained in the continuous subset $\mathcal{C}_{\text{goal}} \subseteq \mathcal{C}$, as shown in Fig. 26.13b, which represents the actual solution space to the planning problem for this reaching task.

Adding the obstacle from Fig. 26.10 to the workspace of this manipulation task produces the configuration space depicted in Fig. 26.14. Note that the $\mathcal{C}$-obstacle corresponding to collisions between the manipulator and the cylinder itself has been disregarded in this visualization for clarity. Approximately half of the possible solutions are disconnected from the zero configuration and a quarter are in collision, which leaves only one quarter as reachable and therefore suitable candidates for $q_{\text{goal}}$ in a classical planning query.

In this section, we have covered the basics of manipulation planning, developed a mathematical formulation for the kinds of search spaces that arise, and discussed some of the important geometric and topological considerations. Section 26.6 gives an overview of additional topics and extensions with pointers to further reading.

## 26.4 Assembly Motion

*Assembly* or an *assembly task* defines the process of putting together manufactured parts to make a complete product, such as a machine. It is a major operation in the manufacturing process of any product. Assembly automation with robots aims to reduce cost and increase the quality and efficiency of the operation. In environments hazardous to humans, such as in space, having robots perform assembly tasks could save human lives. Assembly has long been not only an important but also one of the most challenging applications for robotics. There are many significant research issues related to the broad scope of assembly automation, from design for assembly to tolerance analysis, assembly sequence planning, fixture design, etc. This section is only focused on the issue of robotic motion for assembly.

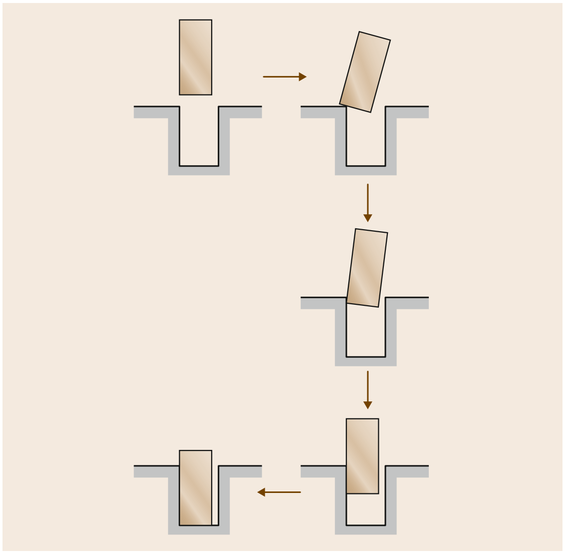
Fig. 26.15 A sequence of contact transitions for the peg-in-hole insertion task

The concerned *assembly motion* is that of a robot manipulator holding a part and moving it to reach a certain *assembled state*, i. e., a required spatial arrangement or contact against another part. The main difficulty of assembly motion is due to the requirement for high precision or low tolerance between the parts in an assembled state. As a result, the assembly motion has to overcome uncertainty to be successful. *Compliant motion* is defined as motion constrained by the contact between the held part and another part in the environment. As it reduces uncertainty through reducing the degrees of freedom of the held part, compliant motion is desirable in assembly.

Consider the peg-in-hole insertion example introduced earlier (see Fig. 26.2). If the clearance between the peg and the hole is very small, the effect of uncertainty will most likely cause a downward insertion motion of the peg to fail, i. e., the peg ends up colliding with the entrance of the hole in some way without reaching the desired assembled state. Therefore, a successful assembly motion has to move the peg out of such an unintended contact situation and lead it to reach the desired assembled state eventually. To make this transition, compliant motion is preferred. Often a sequence of contact transitions via compliant motion is necessary before the desired assembled state can be reached. Figure 26.15 shows a typical sequence of contact transitions for the peg-in-hole task.

Assembly motion strategies that incorporate compliant motion can be broadly classified into two groups: *passive compliance* and *active compliant motion*, and both groups of strategies require certain information characterizing *topological contact states* between parts.

## 26.4.1 Topological Contact States

When a part A contacts a part B, the configuration of A (or B) is a *contact configuration*. Often a set of contact configurations share the same high-level contact characteristics. For example, "a cup is on the table" is a high-level description shared by all the contact configurations of the cup when its bottom are on the table. Such a description is often what really matters in assembly motion as it characterizes a spatial arrangement that could be either an assembled state or just a *contact state* between a part and another part.

For contacting polyhedral objects, it is common to describe a *contact state* topologically as a set of *primitive contacts*, each of which is defined by a pair of contacting surface elements in terms of faces, edges, and vertices.

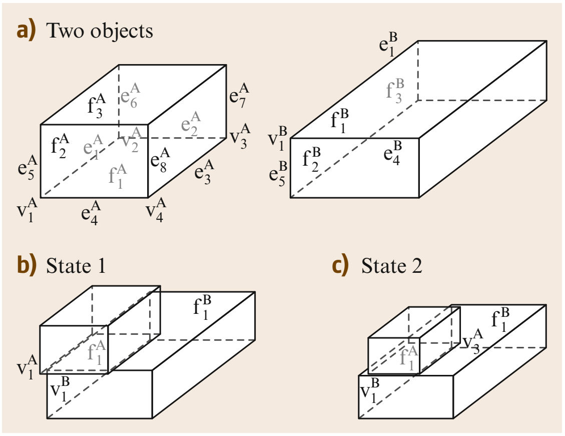

**Fig. 26.16** Indistinguishable contact states due to positional uncertainty of the top part

Different contact state representations essentially differ only in how primitive contacts are defined. One common representation [26.23] defines primitive contacts as point contacts in terms of vertex–edge contacts for two-dimensional (2-D) polygons, and vertex–face and edge–edge contacts for three-dimensional (3-D) polyhedra. Another representation [26.24] defines a primitive contact as between a pair of *topological surface elements* (i. e., faces, edges, and vertices). Here a contact primitive can characterize a contact region that is either a point, a line segment, or a planar face, unlike the point-contact notion. From the viewpoint of contact identification via sensing, however, both representations can result in states that are different by definition but indistinguishable in identification due to uncertainties. Figure 26.16 shows such an example.

The notion of a *principal contact* [26.25,26] presents a high-level primitive contact that is more robust to recognition. A *principal contact* (PC) denotes a contact between a pair of topological surface elements that are not boundary elements of other contacting topological surface elements. The boundary elements of a face are the edges and vertices bounding it, and the boundary elements of an edge are the vertices bounding it. Different PCs between two objects correspond to different degrees of freedom of the objects, which often also correspond to significant differences in the contact forces and moments. As shown in Fig. 26.16, the indistinguishable states in terms of the other contact primitives are grouped as a single contact state in terms of PCs. Thus, there are also fewer contact states in terms of PCs, leading to a more concise characterization of contact states. In fact, every contact state between two convex polyhedral objects is described by a single PC.

## 26.4.2 Passive Compliance

*Passive compliance* refers to strategies that incorporate compliant motion for error correction during the assembly motion without requiring active and explicit recognition and reasoning of contact states between parts.

### Remote Center Compliance

In the 1970s, a *remote center compliance* (RCC) device was developed to assist high-precision peg-in-hole insertion [26.27–29]. The RCC is a mechanical spring structure used as a tool attached to the end-effector of a robot manipulator to hold a round peg when it is inserted into a round hole. The RCC is designed to have high stiffness along the direction of insertion but high lateral and angular compliances $K_x$ and $K_\theta$, and it projects the center of compliance near the tip of the peg (hence the name remote center compliance) to overcome small lateral and angular errors in the peg's position and orientation in response to the contact forces applied to the peg by the hole during insertion (Fig. 26.17). The large lateral and angular compliance of the RCC also helps to avoid *wedging* and *jamming*, conditions that cause the peg to stick in two-point contacts due to contact forces in direct opposition (wedging) or ill-proportioned contact forces/moments (jamming). Because the RCC is low cost and reliable for fast insertions, it has long seen successful industrial applications.

However, the RCC has limitations. It only applies to round peg-in-hole insertions, and a particular RCC works better for a peg and hole of a particular size. Attempts have been made to expand the applicability of the RCC. Some attempts try to make certain parameters of an RCC adjustable to achieve variable compliance or projection (i. e., the position of the remote center) [26.30–32]. A spatial RCC (SRCC) [26.33] was proposed to achieve square-peg-in-hole insertions. A notable difference with this extension is the combinatorial explosion of the number of possible contact states, given the range of the position and orientation uncertainty of the held part that have to be considered in the design of the device. Unlike the case of the round peg and hole, which is essentially a 2-D problem with a handful of different contact states, there are hundreds of possible different contact states between a square peg and a square hole. From each of these possible contact states, a motion leading the held part to the goal assembled state has to be realized by the device. This makes it difficult to design an RCC device that works for parts

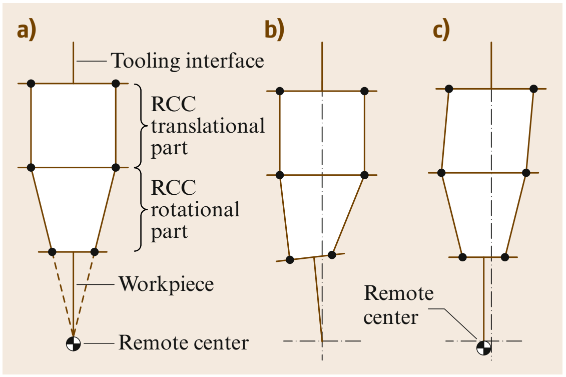

**Fig. 26.17** **(a)** Planar representation of RCC. **(b)** Rotational part of RCC allowing workpiece to rotate. **(c)** Translational part of RCC allowing workpiece to translate

of more general and complex shapes with more possible contact states between them. This is a major limiting factor that prevent more extensions of the RCC to assembly of different parts.

### Admittance Matrix

As an alternative method to the RCC, a particular form of manipulator force control, damping control, was proposed to achieve compliant motion of the held part by the manipulator and to correct small location errors of the held part during assembly. This approach eliminates the need to build a mechanical device like an RCC to achieve error correction. Among the force control laws [26.34], damping control is a common strategy, where a commanded velocity of the held part is modified based on the sensed force caused by contact between the held part and the environment. The resulted actual velocity leads to reduction and hopefully eventual correction of small position or orientation errors of the held part. Let $\boldsymbol{v}$ be a six-dimensional vector representing the actual translational and angular velocity of the held part, $\boldsymbol{v}_0$ be the six-dimensional commanded velocity, and $\boldsymbol{f}$ be a six-dimensional vector representing the sensed force and moment. A linear damping control law is described as:

$$\boldsymbol{v} = \boldsymbol{v}_0 + \boldsymbol{A}\boldsymbol{f} ,$$

where $\boldsymbol{A}$ is a $6\times 6$ matrix, called an *admittance matrix* or accommodation matrix.

The effectiveness of such a damping control law depends on the existence and finding of a proper admittance matrix $\boldsymbol{A}$. There is considerable research on the design of a single $\boldsymbol{A}$ that can make an assembly operation successful regardless of what contact states the held peg may encounter in the process [26.35–39]. This is aimed at cases where a single commanded velocity would be sufficient to achieve an assembly operation when there were no uncertainty or error, such as certain peg-in-hole insertion operations. One main approach to design $\boldsymbol{A}$ is based on explicit kinematic and static analysis of contact conditions under all possible contact states and mating requirements, which result in a set of linear inequalities as constraints on $\boldsymbol{A}$. Learning is also used [26.38] to obtain an $\boldsymbol{A}$ that minimizes the force $\boldsymbol{f}$ without causing instability. Another approach [26.39] applies perturbations to the end-effector during insertion in order to obtain richer force information.

### Learning Control for Assembly

Another category of approaches is to learn proper control for a particular assembly operation through stochastic or neural-network-based methods [26.40–44]. The essence of most of these approaches is to learn to map a reaction force upon the held object caused by contact to the next commanded velocity in order to reduce errors and to achieve an assembly operation successfully. A more recent approach [26.44] maps fused sensory data of pose and vision obtained during human demonstration of assembly tasks to compliant motion signals for successful assembly.

A different approach observes assembly tasks performed by human operators through vision [26.45] or in a virtual environment [26.46,47] and generates a motion strategy necessary for the success of the task that consists of a sequence of recognized contact state transitions and associated motion parameters.

As a sequence of commanded velocities can be generated, unlike RCC or strategies based on a single admittance matrix described above, can be applied to cases with large uncertainties. However, the learned controllers are task dependent.

All of the above assembly motion strategies do not require explicit recognition of contact states during their execution.

## 26.4.3 Active Compliant Motion

Active compliant motion is characterized by error correction based on online identification or recognition of contact states in addition to feedback of contact forces. The capability for active compliant motion allows a robot the flexibility to deal with a much broader range of assembly tasks with large uncertainties and tasks beyond assembly where compliance is required. An active compliant motion system generally requires the following components: a planner to plan compliant motion commands, an identifier to recognize contact states, state transitions, and other state information during task operation, and a controller to execute compliant motion plans based on both low-level feedback provided by sensors and high-level feedback provided by the identifier. Research on each component is described below.

### Fine Motion Planning

Fine motion planning refers to planning fine-scale motions to make an assembly task successful in spite of significant uncertainties. A general approach was proposed [26.48] based on the concept of preimages in configuration space ($\mathcal{C}$-space) [26.23] to devise motion strategies that would not fail in the presence of uncertainties. Given a goal state of the held part defined by a region of goal configurations, a *preimage* of the goal state encodes those configurations from which a commanded velocity will guarantee that the held part reaches the goal state recognizably in spite of location and velocity uncertainties (Fig. 26.18).

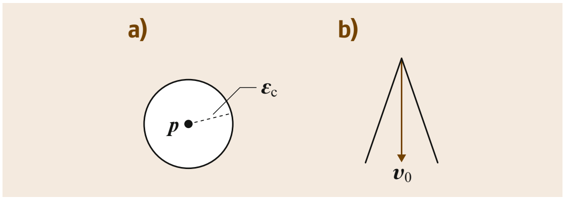

**Fig. 26.18a,b** Location and velocity uncertainties (in the $\mathcal{C}$-space). **(a)** The actual configuration of a part could be inside an uncertainty ball centered at an observed configuration. **(b)** The actual velocity of a part could be inside an uncertainty cone of a commanded velocity

Given the initial location of the held part and the goal state, the preimage approach generates a motion plan in backward chaining by finding the preimage of the goal state associated with a commanded velocity and then the preimage of the preimage, and so on, until a preimage that includes the initial configuration of the held part is found. Figure 26.19 shows an example.

The sequence of commanded velocities associated with the sequence of preimages (starting from the initial preimage that includes the initial location) forms the motion plan that guarantees the success of the task. Starting from the initial preimage, each subsequent preimage in the sequence can be viewed as a subgoal state. In this approach, compliant motions are preferred wherever possible because they typically produce larger preimages [26.48] than pure positioning motions, and subgoal states are often contact states. The approach was further extended [26.49] to include modeling uncertainties of objects.

However, the computability of the approach is a major problem. The requirement of both goal reachability

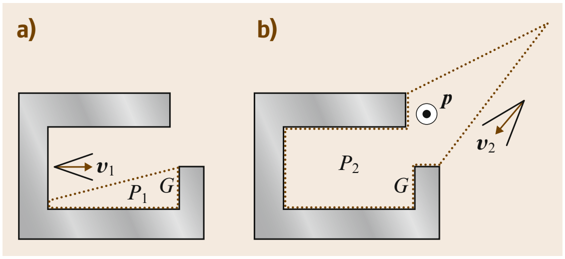

**Fig. 26.19** Backward chaining of preimages: $P_1$ is the preimage of the goal region G, and $P_2$ is the preimage of $P_1$

and recognizability under sensing uncertainties, which are intertwined issues, complicates the computation of preimages. It has been shown [26.50] that the time complexity of generating a plan can be double exponential in $nmr$, where $n$ is the number of plan steps, $m$ is the environment complexity (of the physical space), and $r$ is the dimension of the configuration space. By separating reachability and recognizability and restricting the recognizability power of the original preimage model, computability was improved [26.51, 52].

As an alternative, a two-phase approach is to simplify the fine motion planning into (1) global and offline nominal path planning assuming no uncertainty and (2) local and online replanning to deal with unintended contacts due to uncertainties. Different variations of the two-phase approach have been proposed [26.53–59]. The success of such an approach depends on successful online identification of contact states (see Sect. 26.4.3).

Several researchers also studied the representation and propagation of uncertainties and constraints that have to be satisfied for the success of a task [26.60–64].

### Compliant Motion Planning

Compliant motion planning focuses on planning motions of objects always in contact. *Hopcroft* and *Wilfong* [26.65] proved that, if two arbitrary objects in contact can be moved to another configuration where they are also in contact, then there is always a path of contact configurations (on the boundary of configuration space obstacles, or $\mathcal{C}$-obstacles) connecting the first and second contact configurations. Therefore, not only is compliant motion desirable in many cases, but it is always possible given initial and goal contact configurations.

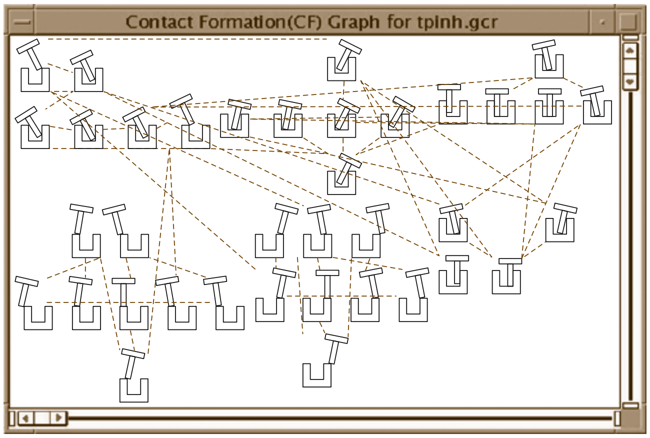

Fig. 26.20 A contact state graph between two planar parts

Because compliant motion occurs on the boundary of $\mathcal{C}$-obstacles, planning compliant motion poses special challenges not present in collision-free motion planning: it requires the exact information of contact configurations on the boundary of $\mathcal{C}$-obstacles. Unfortunately computing $\mathcal{C}$-obstacles exactly remains a formidable task to date. While there are exact descriptions of $\mathcal{C}$-obstacles for polygons [26.66, 67], which are three dimensional, there are only approximations of $\mathcal{C}$-obstacles for polyhedra [26.68, 69], which are six dimensional. If $m$ and $n$ indicate the complexity of two polyhedral objects in contact, the complexity of the corresponding $\mathcal{C}$-obstacle is $\Theta(m^6 n^6)$ [26.70].

Researchers have focused on either reducing the dimensionality and scope of the problem or getting around the problem of computing $\mathcal{C}$-obstacles altogether. Some researchers have studied compliant motion planning on $\mathcal{C}$-obstacles of reduced dimensions [26.59, 71, 72]. It is more popular to plan compliant motions based on a predetermined graph of topological contact states to avoid the problem of computing $\mathcal{C}$-obstacles [26.73–76]. In particular, one approach [26.74, 75] models an assembly task of polygonal parts as a discrete event system using Petri nets. However, contact states and transitions are often generated manually, which is awfully tedious for even assembly tasks of simple geometry [26.33] and is practically infeasible for complex tasks due to the huge number of different contact states.

Therefore, automatic generation of a contact state graph is desirable and even necessary. A method was first developed to enumerate all possible contact states and their connections between two convex polyhedral objects [26.77]. Recall that a contact state between two convex polyhedra can be described by a single principal contact (Sect. 26.4.1), and any principal contact between two topological surface elements of two convex polyhedra describes a geometrically valid contact state. These are nice properties and greatly simplify the problem of contact state graph generation. In general, however, to construct a contact state graph between two objects automatically requires the handling of two rather difficult issues:

1. how to generate *valid* contact states, i. e., how to tell if a set of principal contacts corresponds to a geometrically valid contact state, given the geometries of objects, and
2. how to link one valid contact state to another in the graph, i. e., how to find the neighboring (or adjacency) relations among the regions of contact

configurations belonging to different contact states in the contact configuration space.

A general and efficient divide-and-merge approach was introduced [26.26] for automatically generating a contact state graph between two arbitrary polyhedra. Each node in the graph denotes a contact state, described by a topological *contact formation* (CF) [26.25] as a set of principal contacts and a configuration satisfying the CF. Each edge connects the nodes of two neighboring contact states. Figure 26.20 shows an example contact state graph between two planar parts.

The approach handled the above two issues simultaneously by directly exploiting both topological and geometrical knowledge of contacts in the *physical space* of objects and by dividing the problem into simpler subproblems of generating and merging special subgraphs. Specifically, the approach takes advantage of the fact that a contact state graph can be divided into special subgraphs called the *goal-contact relaxation* (GCR) graphs, where each GCR graph is defined by a locally most constrained valid contact state, called the seed, and its less-constrained neighboring valid contact states, which is easier to generate because of several properties. The main properties include:

- given a valid contact state, $\mathrm{CS}_i$, all of its less constrained neighboring contact states can be hypothesized topologically from the principal contacts in $\mathrm{CS}_i$,
- a hypothesized less constrained neighboring contact state, $\mathrm{CS}_j$, is valid, if and only if there exists a compliant motion to relax certain constraints of $\mathrm{CS}_i$ to obtain $\mathrm{CS}_j$ which does not result in any other contact state, called neighboring relaxation,
- neighboring relaxation can often be achieved by instantaneous compliant motion.

With this approach, a contact state graph of several hundreds or thousands of nodes and links can be generated in a few seconds.

With a contact state graph, the problem of compliant motion planning can be decomposed into two simpler subproblems at two levels: (1) high level: graph search for state transitions from one node to another in a contact state graph, and (2) low level: contact motion planning within the set of contact configurations constrained by the *same* contact state (and the same contact formation), called *CF-compliant motion planning*. A general contact motion plan crossing several contact states can be considered as consisting of segments of CF-compliant motions in different contact states. One approach plans [26.78] CF-compliant motions based on random sampling of CF-compliant configurations and extending the probabilistic roadmap motion planning technique [26.79].

### Contact State Identification

Successful execution of a fine-motion or compliant motion plan depends on correct online identification of contact states during execution. Contact state identification uses both model information of contact states (including topological, geometrical and physical information) and sensory information of location and force/torque of the end-effector. The presence of sensing uncertainties make the identification problem nontrivial. Approaches for contact state identification are different in ways of dealing with uncertainties.

One class of approaches obtain the mapping between sensory data (in the presence of sensing uncertainties) and corresponding contact states through learning. Models for learning include hidden Markov models [26.80, 81], thresholds [26.82, 83], neural network structures, and fuzzy classifiers [26.84–88]. Training data are obtained either in an unsupervised way or by human task demonstration. Such approaches are task dependent: a new task or environment requires new training.

Another class of approaches are based on analytical models for contact states. A common strategy is to predetermine the set of configurations, constraints on configurations, or the set of forces/torques or force/torque constraints that are possible for each contact state and match such information against the sensed data to identify contact states online. Some approaches do not consider the effect of uncertainties [26.89,90], while others consider uncertainties modeled either by uncertainty bounds or by probability distributions [26.24,76,91–95]. An alternative strategy is based on checking the distance between contacting elements in Cartesian space: either by growing the objects with pose uncertainty and intersecting the obtained regions [26.96], or by combining the distance between the objects with several other factors such as the instantaneous approach direction [26.97].

In terms of how sensory data are used, some identification schemes are rather static because they do not consider the prior motion and state identifications before the current contact state is reached, while others use prior history or even use new motion or *active sensing* to help current identification. The latter include approaches that perform simultaneous contact state identification and parameter estimation. In such approaches, an analytical contact state model is expressed as a function of uncertain parameters, such as the pose or dimension of an object in contact. At each contact state, some of the uncertain parameters can be observed or estimated with reduced uncertainty. For example, if the held object is at a face–face contact with another object, parameters describing the normals of the contacting faces can be estimated with the help of force/torque sensing. During a task execution (i. e., the execution of a motion plan), the uncertain parameters are estimated along with the identification of contact states with increasing accuracy, which in turn makes the subsequent contact state identifications more accurate. This also improves force control and contact state transition monitoring. The increased performance, however, comes at a higher computational cost.

Simultaneous contact state identification and parameter estimation is most often done by ruling out contact state models that are inconsistent with sensed data or parameter estimation results [26.98–104]. One notable exception [26.105] performs a truly simultaneous estimation by describing the system as a hybrid joint probability distribution of contact states and parameters.

Active sensing is about deliberate use of applied force/torque or motion to better aid contact state identification and parameter estimation. Earlier research focused on simple force strategy [26.106] or movability tests [26.24, 97]. More recently, methods were introduced [26.107] to design sequences of contact state transitions and compliant motion strategies that were optimized for active sensing to determine all uncertain geometric parameters while achieving certain goal contact state.

From a different perspective, an approach was introduced to analyze contact state distinguishability and unknown/uncertain parameter identifiability [26.108] during the design phase of a contact state identifier or parameter estimator.

### Execution of Compliant Motion Plans

While compliant or force control is a much researched subject (see Chap. 8), how to execute a compliant motion plan automatically has received much less attention until recently.

A general compliant motion plan (as the output of a compliant motion or fine motion planner) usually consists of a sequence of contact states, and within each contact state, a path of contact configurations compliant to the contact state and leads to the next contact state in the sequence. Such topological and geometrical information alone is not sufficient for a compliant controller to execute the plan, especially when there are many different and complex contact states.

A hybrid position/force controller requires a specification that separates dimensions of force control from those of position/velocity control. Such specifications not only have to vary for different contact states but may have to vary from one contact configuration to another within the same contact state, i. e., a control specification is a function of time or configuration along a compliant trajectory in general. Moreover, it is far from trivial to make control specifications for complex contact states involving multiple principal contacts [26.109].

Recently an approach was introduced [26.110] to convert automatically a compliant path of contact configurations across a sequence of contact states into wrench, twist, and position control signals $\boldsymbol{w}(t)$, $\boldsymbol{t}(t)$, and $\boldsymbol{p}(t)$ for a hybrid controller to execute the plan. The approach was experimentally verified successfully.

However, there is not yet a full integration of planning, online contact state identification, online replanning (to deal with contact states that are off the preplanned path due to uncertainty), and compliant motion control.

## 26.5 Unifying Feedback Control and Planning

In previous sections we discussed task-level control methods and planning algorithms. Both task-level control and planning methods are able to address specific constraints imposed on robot motion in the context of manipulation tasks. However, as illustrated in Fig. 26.3, neither of these categories of methods is able to address all constraints completely. Control remains susceptible to local minima and thus cannot guarantee that a particular motion will lead to the desired result. Planning methods, on the other hand, overcome the problem of local minima by projecting possible motions into the future to predict if a sequence of motion will lead to success. Given certain assumptions, the resulting plan can be guaranteed to succeed. However, the computations associated with this process are generally too computationally complex to satisfy the feedback requirements of manipulation tasks. Unfortunately, this means that control and motion planning by themselves are unable to address the problem of moving robots for general manipulations tasks.

In this final section of this chapter, we will review efforts to combine the advantages of control methods and planning methods into a single approach to determine the motion of a robot. These efforts aim to create methods that avoid the susceptibility to local minima while satisfying the feedback requirements. Early attempts of integrating planning and control occurred in the early 1990s. Not all of these efforts are specifically directed at manipulation but many contain insights relevant to the topic. In this section, we will review these efforts and also discuss some of the more recent research efforts aimed at unifying motion planning and feedback control.

Motion planning (Chap. 5) and motion control (Chap. 6) have traditionally been regarded as two distinct areas of research. However, these areas have many features in common. Both areas are concerned with the motion of robotic mechanisms, and more importantly planning and control methods both determine a representation that maps robot state to robot motion. In the case of feedback control, this representation is a potential function defined for a given region of the state space. The gradient of the potential function at a specific state encodes the motion command. In contrast, motion planning determines motion plans. These motion plans also encode motions for a set of states.

Motion planning and feedback control also differ in certain aspects. A motion planner generally makes stronger assumptions than a controller about the environment, the ability to assess its state, and the changes that can occur in the environment. Motion planning also requires the ability to project the robot's state into the future, given its current state and a particular action. By using this ability together with global information about the environment, motion planners can determine motions that are not susceptible to local minima. This positive characteristic of motion planners, however, results in significantly increased computational cost (see Chap. 5). The large discrepancy in computational requirements for planning and control and the resulting divergence of computational techniques may explain the current separation of the two fields. Researchers are beginning to attempt to reverse this separation by devising a unified theory of planning and control.

## 26.5.1 Feedback Motion Planning

Feedback motion planning combines planning and feedback into a single motion strategy. A planner considers global information to compute a feedback motion plan free of local minima. Such a feedback motion plan can be interpreted as a potential function or vector field whose gradient will lead the robot to the goal state from any reachable part of the state space [26.18]. Local minima-free potential functions are also called navigation functions [26.111, 112]. Given the current state of the robot and the global navigation function, feedback control is used to determine the robot's motion. The consideration of feedback about the robot's state in the context of a global navigation function reduces the susceptibility to sensing and actuation uncertainty.

Feedback motion planning, in principle, can address the entire spectrum of motion constraints and their feedback requirements (see Fig. 26.3). Given a global navigation function that considers all motion constraints, the feedback requirements can easily be satisfied, since the feedback motion plan already specifies the desired motion command for the entire state space. Obviously, the major challenge in feedback motion planning is the computation of such a navigation function or feedback motion plan. The problem becomes particularly difficult in the context of a manipulation task, since the state space (or configuration space) changes each time the robot grasps or releases an object in the environment (Sect. 26.3). This change makes it necessary to recompute the feedback motion plan. Frequent recomputation is also necessary in dynamic environments, where the motion of obstacles can repeatedly invalidate a previously computed feedback motion plan.

In the remainder of this section, we will review a variety of methods for computing navigation functions. In general, the problem of efficiently computing feedback motion plans for manipulation tasks remains unsolved. We therefore also present methods that do not explicitly consider manipulation. These methods can be divided into three categories: (1) exact methods, (2) approximate methods based on dynamic programming, and (3) approximate methods based on composing and sequencing simpler potential functions.

The earliest exact methods for the computation of navigation functions are applicable to simple environments with obstacles of specific shapes [26.111–113]. Approximate methods based on discretized spaces (grids) overcome this limitation but possess an exponential computational complexity in the number of dimensions of the state space. These approximate navigation functions are called numerical navigation functions [26.114]. These navigation functions are often used for applications in mobile robotics. Due to the low-dimensional configuration space associated with mobile robots, they can generally solve motion planning problems robustly and efficiently.

Some physical processes, such as heat transfer or fluid flow, can be described by a specific type of differential equation, called harmonic functions. These functions possess properties that make them suitable as navigation functions [26.115–119]. Navigation functions based on harmonic functions are most commonly computed in an approximate, iterative fashion. The requirement for iterative computation increases the computational cost relative to the simpler numerical navigation functions [26.114].

More recent methods for the computation of numerical navigation functions consider differential motion constraints at a significantly reduced computational cost [26.120]. These methods rely on classical numerical dynamic programming techniques and numerical optimal control. However, in spite of their reduced computational complexity, these methods remain too computationally costly to be applied to manipulation tasks with many degrees of freedom in dynamic environments.

Navigation functions can also be computed by composing local potential functions based on global information. This is illustrated in Fig. 26.21. The goal is to compute a navigation function for the entire configuration space $\mathcal{C}$. This is accomplished by sequencing overlapping funnels. Each of the funnels represents a simple, local potential function. By following the gradient of this potential function, motion commands can be determined for a subset of the configuration space. If the funnels are sequenced correctly, the composition of local funnels can yield a global feedback plan. This feedback plan can be viewed as a hybrid system [26.121] in which the sequencing of funnels represents a discrete transition structure, whereas the individual controllers operate in a continuous domain. The composition of funnels is considered planning. The consideration of global information during the planning permits to determine a funnel composition that avoids local minima.

One of the earliest methods based on the global composition of local funnels was proposed by *Choi* et al. [26.122]. In this method, the state space of the robot is decomposed into convex regions. The connectivity of these regions is analyzed to determine global information about the state space. This information can be used to combine simple, local potential functions, one for each convex regions of state space, into a local minima-free potential function. More rigorous approaches based on this idea have been developed. These approaches can consider the dynamics of the robot and nonholonomic

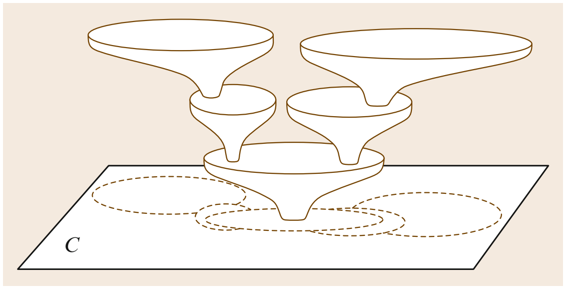

**Fig. 26.21** Composing funnels into a navigation function

motion constraints [26.123–125]. These methods have only been applied to low-dimensional state spaces and cannot easily be applied to manipulation tasks.

The random neighborhood graph [26.126] is a sampling-based method of computing a decomposition of the overall state space. As before, a navigation function can be computed by analyzing the global connectivity of the decomposition and imposing adequate local potential functions for each subdivision. A specialized method following the same principle for planar robots in polygonal environments has also been proposed [26.127]. The idea of composing local potential functions has also been applied successfully to a complex robot control task [26.128]. Finally, in [26.129] a general and efficient method of computing a smooth feedback plan over a cylindrical algebraic decomposition of configuration space has been proposed.

The computation of a navigation function over the entire configuration space quickly becomes intractable as the dimensionality of the space increases. To overcome this challenge, in particular in the context of autonomous mobile manipulation, workspace heuristics have been employed to determine a navigation function efficiently. This navigation function does not cover the entire configuration space but only those regions heuristically determined to be relevant to the motion problem [26.130]. This method of generating feedback plans is able to satisfy the various motion constraints depicted in Fig. 26.3 and their respective feedback requirements.

### 26.5.2 Augmenting Global Plans with Feedback

The manipulation planning techniques described in Sect. 26.3 are not susceptible to local minima, as they consider global state space information. Due to the computational complexity associated with the consideration of global information, these planning techniques are not able to satisfy the feedback requirements of manipulation tasks. However, the required frequency of feedback about global motion are relatively low: the global connectivity of the configuration space changes relatively infrequently (Fig. 26.3). It would therefore be possible to consider feedback for global motion at the slow rates the planner can accommodate, while considering feedback for other motion constraints at higher frequencies. To achieve this, global motion plans have to be augmented with reactive components that incrementally modify the global plan in response to feedback from the environment. As long as the global connectivity information captured by the plan remains valid, the incremental modifications can ensure that all other motion constraints, ranging from task requirements to reactive obstacles avoidance, are satisfied.

The elastic-band framework [26.131] augments global plans with reactive obstacle avoidance. A global configuration space path, determined by a planner, is covered with local potential functions, each of which is derived from the local distribution of obstacles around the path. These local potentials cause the path to deform so as to maintain a minimum distance from obstacles. Visually, the path behaves as an elastic band that is deformed by the motion of obstacles. The local potential functions, together with the global path, can be viewed as a navigation function for a local region of the configuration space. Integrated with a global planner and replanner, the elastic-band framework permits real-time obstacle avoidance that is not susceptible to local minima. However, the feedback frequency for global motion remains limited by the global motion planner. Specific task constraints have not been integrated into the elastic-band framework; consequently, its application to manipulation tasks is limited.

In its original formulation, the elastic-band framework assumed that all degrees of freedom of the robot are holonomic. An extended formulation augments motion paths for nonholonomic platforms with reactive components [26.132].

The elastic-strip framework [26.133] also augments global motion plans with reactive obstacle avoidance. In addition to reactive obstacle avoidance, however, the elastic-strip framework can accommodate task constraints. Similarly to an elastic band, an elastic strip covers a global path with local potential functions. In contrast to the elastic-band framework, these potential functions are based on task-level controllers (Sect. 26.2) and therefore allow the task-consistent modification of the global path. The elastic-strip framework is therefore well suited for the execution of manipulation plans in dynamic environments. An elastic strip will be incrementally modified to represent a constraint-consistent trajectory, as long as the global information captured by the underlying plan remains valid. The elastic-strip framework has been applied to a variety of manipulation tasks on a mobile manipulation platform.

Extending the elastic-band and elastic-strips frameworks, the elastic-roadmap framework combines reactive task-level control with efficient global motion planning [26.130]. The elastic roadmap represents a hybrid system of task-level controllers that are composed into a navigation function, thereby satisfying the motion constraints depicted in Fig. 26.3 and their respective feedback requirements.

## 26.6 Conclusions and Further Reading

In this chapter, an overview of motion generation and control strategies in the context of robotic manipulation tasks has been provided. Issues related to modeling the interfaces between the robot and the environment at the different time scales of motion and incorporating sensing and feedback were considered. Manipulation planning was introduced as an extension to the basic motion planning problem, which can be modeled as a hybrid system of continuous configuration spaces arising from the act of grasping and moving parts in the environment. The important example of assembly motion has been discussed through the analysis of contact states and compliant motion control. The operational space framework, as described in Sect. 26.2, permits the position and force control of operational points on a serial-chain manipulator. A number of extensions have been proposed in the literature, extending the framework to situations with multiple concurrent contact points, cooperative manipulation scenarios, and branching kinematic chains. For manipulation planning, extensions involving grasp and regrasp planning, multiple robots, multiple parts, and movable obstacles have been considered. In this section we will briefly discuss these extensions and refer to the appropriate literature.

### General Contact Models

The hybrid force/motion control described by *Raibert* and *Craig* [26.134] has been shown to have shortcomings [26.135]. A general contact model for dynamically decoupled force/motion control in the context of operational space control that overcomes these problems has been presented in [26.135]. This general contact model has been extended into a compliant motion control framework to enable force control for multiple contact points [26.136] in nonrigid environments, and to enable force control for multiple contact points on different links of a kinematic chain [26.137]. The operational space framework for task-level control has been applied for the real-time simulation of complex dynamic environments [26.138]. Contact points between objects in the environment are modeled as operation points, resulting in a mathematically elegant framework for resolving impulses and contact constraints for moving bodies.

#### Cooperative Manipulation Control

If multiple task-level controlled robots collaborate to manipulate an object, they form a closed kinematic chain that is connected through the object. The dynamics of the entire system can be described using the notion of an augmented object [26.139, 140], in which the dynamics of the manipulators and the object are combined to form a model of the overall system. The internal forces that occur at the grasp points during motion can be modeled using the virtual linkage framework [26.141]. A number of alternative strategies for the cooperative manipulation of multiple robots outside of the operational space framework have been proposed [26.142–148].

#### Control of Branching Mechanisms

So far, we have implicitly assumed that the task-controlled robot consists of a single kinematic chain. This assumption does not hold in the case of kinematically more complex mechanisms, such as humanoid robots (Chap. 56). These robots can consist of multiple branching kinematic chains. If we consider the torso of a humanoid robot to be the robot's base, for example, then the legs, arms, and the head represent five kinematic chains attached to this base. We call such a mechanism a branching mechanism if it does not contain any closed kinematic loops. Tasks performed by a branching mechanism may require the specification of operational points on any one of those branches; for example, while the legs perform locomotion the hands achieve a manipulation task and the head is oriented to maintain visibility of the manipulated object.

The operational space framework has been extended to task-level control of branching mechanisms [26.149]. This extension combined operational points and associated Jacobians and computes an operational space inertia matrix that combines all operational points. This matrix can be computed efficiently with an algorithm that in practice is linear in the number of operational points [26.150, 151].

#### Nonholonomic Mobile Manipulation

All previously discussed work on task-level control assumes that the degrees of freedom are holonomic. For manipulator arms this is generally a valid assumption. The most common type of mobility platform, however, is based on either differential drives, synchro-drives, or Ackerman steerings (point to a section), all of which are subject to nonholonomic constraints. The operational space framework has been extended to mobile manipulation platforms that combine a nonholonomic mobility platform with a holonomic manipulator arm [26.152–154], enabling the task-level control of a large class of mobile manipulation platforms.

#### Learning Models with Uncertainty

The efficacy of operational space control depends on the accuracy of the dynamic model of the robot. In particular when multiple behaviors are executed using null-space projections, modeling errors can have significant effects. To overcome the reliance on accurate dynamic models, reinforcement learning can be used to learn operational space controllers [26.155].

#### Grasping and Regrasp Planning

Deciding the intermediate stable configurations for a part necessary to complete a manipulation task and selecting proper grasps are both difficult problems in themselves that operate over a continuum of possible solutions. *Simeon* et al. developed a manipulation planning framework that considers continuous grasps and placements [26.156]. However, selecting from among all possible grasp configurations and deciding when to regrasp is still an open research problem. For an overview of various grasp quality metrics, see *Miller* and *Allen* [26.157, 158] and Chap. 28.

#### Multiple Parts

The manipulation planning framework nicely generalizes to multiple parts, $\mathcal{P}_1, \dots, \mathcal{P}_k$. Each part has its own $\mathcal{C}$-space, and $\mathcal{C}$ is formed by taking the Cartesian product of all the part $\mathcal{C}$-spaces with the manipulator $\mathcal{C}$-space. The set $\mathcal{C}_{\text{adm}}$ is defined in a similar way, but now part–part collisions also have to be removed, in addition to part–manipulator, manipulator–obstacle, and part–obstacle collisions. The definition of $\mathcal{C}_{\text{stable}}$ requires that all parts be in stable configurations; the parts may even be allowed to stack on top of each other. The definition of $\mathcal{C}_{\text{grasp}}$ requires that one part is grasped and all other parts are stable. There are still two modes, depending on whether the manipulator is grasping a part. Once again, transitions occur only when the robot is in $\mathcal{C}_{\text{trans}} = \mathcal{C}_{\text{stable}} \cap \mathcal{C}_{\text{grasp}}$.

### Planning for Multiple Robots

Generalizing to $k$ robots would lead to $2k$ modes, in which each mode indicates whether each robot is grasping a part. Multiple robots may even be allowed to grasp the same part, which leads to the interesting problem of planning for closed kinematic chains and cooperative motion (see Chap. 29). *Koga* addressed multi-arm manipulation planning where the same object must be grasped and moved by several manipulator arms [26.21]. Another generalization could allow a single robot to grasp more than one part simultaneously.

### Planning for Closed Kinematic Chains

The subspace of $\mathcal{C}$ that results from maintaining kinematic closure arises when multiple robots grasp the same part, or even when multiple fingers of the same hand grasp a single part (see Chap. 15). Planning in this context requires that paths remain on a lower-dimensional variety for which a parameterization is not available. Planning the motion of parallel mechanisms or other systems with loops typically requires maintaining multiple closure constraints simultaneously (Chap. 12).

### Planning with Movable Obstacles

In some cases, the robot may be allowed to reposition obstacles in the environment rather than simply avoid them. *Wilfong* first addressed motion planning in the presence of movable obstacles [26.159]. Wilfong proved that the problem is PSPACE-hard even in two-dimensional environments where the final positions of all movable objects are specified. *Erdmann* and *Lozano-Perez* considered coordinated planning for multiple moving objects [26.160]. *Alami* presented a general algorithm for the case of one robot and one movable object and formulated the space of grasping configurations into a finite number of cells [26.17]. *Chen* and *Hwang* developed a planner for a circular robot that is allowed to push objects aside as it moves [26.161]. *Stilman* and *Kuffner* considered movable obstacles in the context of navigation planning [26.162], and manipulation planning [26.163]. *Nieuwenhuisen* et al. also developed a general framework for planning with movable obstacles [26.164].

### Nonprehensile Manipulation

*Lynch* and *Mason* explored scenarios in which grasping operations are replaced by pushing operations [26.165]. The space of stable pushing directions imposes nonholonomic constraints on the motion of the robot, which opens up issues related to controllability. Related issues also arise in the context of part-feeders and manipulation for manufacturing and assembly (Chap. 42).

### Assembly Motion Extensions

The work in assembly motion described in this chapter has focused on rigid-part assembly, and the parts considered are mostly polyhedral or of simple nonpolyhedral shapes, such as round pegs and holes. More recent work on contact state analysis for curved objects can be found in [26.166]. An emerging field in assembly is micro/nanoassembly (Chap. 18). Flexible or deformable part assembly has also begun to attract attention [26.167–169]. Virtual assembly [26.170] for simulation and prototyping is another interesting area of study.

## References

26.1 M.T. Mason: *Mechanics of Robotic Manipulation* (MIT Press, Cambridge 2001)

26.2 H. Inoue: Force feedback in precise assembly tasks, Tech. Rep. **308** (Artificial Intelligence Laboratory, MIT, Cambridge 1974)

26.3 T. Lozano-Pérez, M. Mason, R.H. Taylor: Automatic synthesis of fine-motion strategies for robots, Int. J. Robot. Res. **31**(1), 3–24 (1984)

26.4 O. Khatib: A unified approach to motion and force control of robot manipulators: the operational space formulation, Int. J. Robot. Autom. **3**(1), 43–53 (1987)

26.5 O. Khatib, K. Yokoi, O. Brock, K.-S. Chang, A. Casal: Robots in human environments, Arch. Contr. Sci. **11**(3/4), 123–138 (2001)

26.6 G. Strang: *Linear Algegra and Its Applications* (Brooks Cole, New York 1988)

26.7 H. Seraji: An on-line approach to coordinated mobility and manipulation, Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Vol. 1 (Atlanta 1993) pp. 28–35

26.8 J.M. Cameron, D.C. MacKenzie, K.R. Ward, R.C. Arkin, W.J. Book: Reactive control for mobile manipulation, Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Vol. 3 (Atlanta 1993) pp. 228–235

26.9 M. Egerstedt, X. Hu: Coordinated trajectory following for mobile manipulation, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Francisco 2000)

26.10 P. Ögren, M. Egerstedt, X. Hu: Reactive mobile manipulation using dyanmic trajectory tracking, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Francicso 2000) pp. 3473–3478

26.11 J. Tan, N. Xi, Y. Wang: Integrated task planning and control for mobile manipulators, Int. J. Robot. Res. **22**(5), 337–354 (2003)

26.12 Y. Yamamoto, X. Yun: Unified analysis on mobility and manipulability of mobile manipulators, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Detroit 1999) pp. 1200–1206

26.13 L. Sentis, O. Khatib: Control of free-floating humanoid robots through task prioritization, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Barcelona 2005)

26.14 L. Sentis, O. Khatib: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives, Int. J. Human Robot. **2**(4), 505–518 (2005)

26.15 O. Khatib: Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. **5**(1), 90–98 (1986)

26.16 M. Huber, R.A. Grupen: A feedback control structure for on-line learning tasks, Robot. Auton. Syst. **22**(3-4), 303–315 (1997)

26.17 R. Alami, J.P. Laumond, T. Sim'eon: Two manipulation planning algorithms, Workshop Algorithm. Found. Robot. (1994)

26.18 S.M. LaValle: *Planning Algorithms* (Cambridge Univ. Press, Cambridge 2006), (also available at http-//msl.cs.uiuc.edu/planning/)

26.19 R. Grossman, A. Nerode, A. Ravn, H. Rischel (Eds.): *Hybrid Systems* (Springer, Berlin, Heidelberg 1993)

26.20 K.J. Gupta Ahuactzin, E. Mazer: Manipulation planning for redundant robots: a practical approach, Int. J. Robot. Res. **17**(7), 731–747 (1998)

26.21 Y. Koga: On Computing Multi-Arm Manipulation Trajectories. Ph.D. Thesis (Stanford University, Stanford 1995)

26.22 D. Bertram, J.J. Kuffner, T. Asfour, R. Dillman: A unified approach to inverse kinematics and path planning for redundant manipulators, Proc. IEEE Int. Conf. Robot. Autom. (ICRA'06) (2006) pp. 1874–1879

26.23 T. Lozano-Pérez: Spatial planning: a configuration space approach, IEEE Trans. Comput. **C-32**(2), 108–120 (1983)

26.24 R. Desai: On Fine Motion in Mechanical Assembly in Presence of Uncertainty. Ph.D. Thesis (Department of Mechanical Engineering, University of Michigan, 1989)

26.25 J. Xiao: Automatic determination of topological contacts in the presence of sensing uncertainties, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Atlanta 1993) pp. 65–70

26.26 J. Xiao, X. Ji: On automatic generation of high-level contact state space, Int. J. Robot. Res. **20**(7), 584–606 (2001), (and its first multi-media extension issue eIJRR)

26.27 S.N. Simunovic: Force information in assembly processes, Proc. 5th Int. Symp. Ind. Robots (1975) pp. 415–431

26.28 S.H. Drake: Using Compliance in Lieu of Sensory Feedback for Automatic Assembly. Ph.D. Thesis (Department of Mechanical Engineering, Massachusetts Institute of Technology 1989)

26.29 D.E. Whitney: Quasi-static assembly of compliantly supported rigid parts, ASME J. Dyn. Syst. Meas. Contr. **104**, 65–77 (1982)

26.30 R.L. Hollis: A six-degree-of-freedom magnetically levitated variable compliance fine-motion wrist: design, modeling, and control, IEEE Trans. Robot. Autom. **7**(3), 320–332 (1991)

26.31 S. Joo, F. Miyazaki: Development of variable RCC and ITS application, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS), Vol. 3 (Victoria 1998) pp. 1326–1332

26.32 H. Kazerooni: Direct-drive active compliant end effector (active RCC), IEEE J. Robot. Autom. **4**(3), 324–333 (1988)

26.33 R.H. Sturges, S. Laowattana: Fine motion planning through constraint network analysis, Proc. Int. Symp. Assem. Task Plan. (1995), 160–170

26.34 D.E. Whitney: Historic perspective and state of the art in robot force control, Int. J. Robot. Res. **6**(1), 3–14 (1987)

26.35 M. Peshkin: Programmed compliance for error corrective assembly, IEEE Trans. Robot. Autom. **6**(4), 473–482 (1990)

26.36 J.M. Schimmels, M.A. Peshkin: Admittance matrix design for force-guided assembly, IEEE Trans. Robot. Autom. **8**(2), 213–227 (1992)

26.37 J.M. Schimmels: A linear space of admittance control laws that guarantees force assembly with friction, IEEE Trans. Robot. Autom. **13**(5), 656–667 (1997)

26.38 S. Hirai, T. Inatsugi, K. Iwata: Learning of admittance matrix elements for manipulative operations, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (1996) pp. 763–768

26.39 S. Lee, H. Asada: A perturbation/correlation method for force guided robot assembly, IEEE Trans. Robot. Autom. **15**(4), 764–773 (1999)

26.40 H. Asada: Representation and learning of nonlinear compliance using neural nets, IEEE Trans. Robot. Autom. **9**(6), 863–867 (1993)

26.41 J. Simons, H. Van Brussel, J. De Schutter, J. Verhaert: A self-learning automaton with variable resolution for high precision assembly by industrial robots, IEEE Trans. Autom. Contr. **27**(5), 1109–1113 (1982)

26.42 V. Gullapalli, J.A. Franklin, H. Benbrahim: Acquiring robot skills via reinforcement learning, IEEE Contr. Syst. **14**(1), 13–24 (1994)

26.43 Q. Wang, J. De Schutter, W. Witvrouw, S. Graves: Derivation of compliant motion programs based on human demonstration, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1996) pp. 2616–2621

26.44 R. Cortesao, R. Koeppe, U. Nunes, G. Hirzinger: Data fusion for robotic assembly tasks based on human skills, IEEE Trans. Robot. Autom. **20**(6), 941–952 (2004)

26.45 K. Ikeuchi, T. Suehiro: Toward an assembly plan from observation. Part I: task recognition with polyhedral objects, IEEE Trans. Robot. Autom. **10**(3), 368–385 (1994)

26.46 H. Onda, H. Hirokawa, F. Tomita, T. Suehiro, K. Takase: Assembly motion teaching system using position/forcesimulator-generating control program, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (1997) pp. 938–945

26.47 H. Onda, T. Suehiro, K. Kitagaki: Teaching by demonstration of assembly motion in VR – non-deterministic search-type motion in the teaching stage, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (2002) pp. 3066–3072

26.48 T. Lozano-Pérez, M.T. Mason, R.H. Taylor: Automatic synthesis of fine-motion strategies for robot, Int. J. Robot. Res. **31**(1), 3–24 (1984)

26.49 B.R. Donald: *Error Detection and Recovery in Robotics* (Springer, Berlin, Heidelberg 1989)

26.50 J. Canny: On computability of fine motion plans, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1989) pp. 177–182

26.51 M. Erdmann: Using backprojections for fine motion planning with uncertainty, Int. J. Robot. Res. **5**(1), 19–45 (1986)

26.52 J.C. Latombe: *Robot Motion Planning* (Kluwer Academic, Dordrecht 1991)

26.53 B. Dufay, J.C. Latombe: An approach to automatic programming based on inductive learning, Int. J. Robot. Res. **3**(4), 3–20 (1984)

26.54 H. Asada, S. Hirai: Towards a symbolic-level force feedback: recognition of assembly process states, Proc. Int. Symp. Robot. Res. (1989) pp. 290–295

26.55 J. Xiao, R. Volz: On replanning for assembly tasks using robots in the presence of uncertainties, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1989) pp. 638–645

26.56 J. Xiao: Replanning with compliant rotations in the presence of uncertainties, Proc. Int. Symp. Intell. Contr. (Glasgow 1992) pp. 102–107

26.57 G. Dakin, R. Popplestone: Simplified fine-motion planning in generalized contact space, Proc. Int. Symp. Intell. Contr. (1992) pp. 281–287

26.58 G. Dakin, R. Popplestone: Contact space analysis for narrow-clearance assemblies, Proc. Int. Symp. Intell. Contr. (1993) pp. 542–547

26.59 J. Rosell, L. Basañez, R. Suárez: Compliant-motion planning and execution for robotic assembly, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1999) pp. 2774–2779

26.60 R. Taylor: The Synthesis of Manipulator Control Programs from Task-Level Specifications. Ph.D. Thesis (Stanford University, Stanford 1976)

26.61 R.A. Brooks: Symbolic error analysis and robot planning, Int. J. Robot. Res. **1**(4), 29–68 (1982)

26.62 R.A. Smith, P. Cheeseman: On the representation and estimation of spatial uncertaity, Int. J. Robot. Res. **5**(4), 56–68 (1986)

26.63 S.-F. Su, C. Lee: Manipulation and propagation of uncertainty and verification of applicability of actions in assembly tasks, IEEE Trans. Syst. Man Cybern. **22**(6), 1376–1389 (1992)

26.64 S.-F. Su, C. Lee, W. Hsu: Automatic generation of goal regions for assembly tasks in the presence of uncertainty, IEEE Trans. Robot. Autom. **12**(2), 313–323 (1996)

26.65 J. Hopcroft, G. Wilfong: Motion of objects in contact, Int. J. Robot. Res. **4**(4), 32–46 (1986)

26.66 F. Avnaim, J.D. Boissonnat, B. Faverjon: A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1988) pp. 1656–1661

26.67 R. Brost: Computing metric and topological properties of c-space obstacles, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1989) pp. 170–176

26.68 B. Donald: A search algorithm for motion planning with six degrees of freedom, Artif. Intell. **31**(3), 295–353 (1987)

26.69 L. Joskowicz, R.H. Taylor: Interference-free insertion of a solid body into a cavity: an algorithm and a medical application, Int. J. Robot. Res. **15**(3), 211–229 (1996)

26.70 H. Hirukawa: On motion planning of polyhedra in contact, Workshop Algorithm. Fund. Robot. (Toulouse 1996) pp. 381–391

26.71 S.J. Buckley: Planning compliant motion strategies, Proc. Int. Symp. Intell. Contr. (1988) pp. 338–343

26.72 E. Sacks: Path planning for planar articulated robots using configuration spaces and compliant motion, IEEE Trans. Robot. Autom. **19**(3), 381–390 (2003)

26.73 C. Laugier: Planning fine motion strategies by reasoning in the contact space, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1989) pp. 653–659

26.74 B.J. McCarragher, H. Asada: A discrete event approach to the control of robotic assembly tasks, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1993) pp. 331–336

26.75 B.J. McCarragher, H. Asada: The discrete event modeling and trajectory planning of robotic assembly tasks, ASME J. Dyn. Syst. Meas. Contr. **117**, 394–400 (1995)

26.76 R. Suárez, L. Basañez, J. Rosell: Using configuration and force sensing in assembly task planning and

execution, Proc. Int. Symp. Assem. Task Plan. (1995) pp. 273–279
26.77 H. Hirukawa, Y. Papegay, T. Matsui: A motion planning algorithm for convex polyhedra in contact under translation and rotation, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Diego, 1994) pp. 3020–3027
26.78 X. Ji, J. Xiao: Planning motion compliant to complex contact states, Int. J. Robot. Res. **20**(6), 446–465 (2001)
26.79 L.E. Kavraki, P. Svestka, J.C. Latombe, M. Overmars: Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
26.80 B. Hannaford, P. Lee: Hidden Markov model analysis of force/torque information in telemanipulation, Int. J. Robot. Res. **10**(5), 528–539 (1991)
26.81 G.E. Hovland, B.J. McCarragher: Hidden Markov models as a process monitor in robotic assembly, Int. J. Robot. Res. **17**(2), 153–168 (1998)
26.82 T. Takahashi, H. Ogata, S. Muto: A method for analyzing human assembly operations for use in automatically generating robot commands, Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Vol. 2 (Atlanta 1993) pp. 695–700
26.83 P. Sikka, B.J. McCarragher: Rule-based contact monitoring using examples obtained by task demonstration, Proc. 15th Int. Jt. Conf. Artif. Intell. (Nagoya 1997) pp. 514–521
26.84 E. Cervera, A. Del Pobil, E. Marta, M. Serna: Perception-based learning for motion in contact in task planning, J. Intell. Robot. Syst. **17**(3), 283–308 (1996)
26.85 L.M. Brignone, M. Howarth: A geometrically validated approach to autonomous robotic assembly, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (Lausanne 2002) pp. 1626–1631
26.86 M. Nuttin, J. Rosell, R. Suárez, H. Van Brussel, L. Basañez, J. Hao: Learning approaches to contact estimation in assembly tasks with robots, Proc. 3rd Eur. Workshop Learn. Robot. (Heraklion 1995)
26.87 L.J. Everett, R. Ravari, R.A. Volz, M. Skubic: Generalized recognition of single-ended contact formations, IEEE Trans. Robot. Autom. **15**(5), 829–836 (1999)
26.88 M. Skubic, R.A. Volz: Identifying single-ended contact formations from force sensor patterns, IEEE Trans. Robot. Autom. **16**(5), 597–603 (2000)
26.89 S. Hirai, H. Asada: Kinematics and statics of manipulation using the theory of polyhedral convex cones, Int. J. Robot. Res. **12**(5), 434–447 (1993)
26.90 H. Hirukawa, T. Matsui, K. Takase: Automatic determination of possible velocity and applicable force of frictionless objects in contact from a geometric model, IEEE Trans. Robot. Autom. **10**(3), 309–322 (1994)
26.91 B.J. McCarragher, H. Asada: Qualitative template matching using dynamic process models for state transition recognition of robotic assembly, ASME J. Dyn. Syst. Meas. Contr. **115**(2), 261–269 (1993)
26.92 T.M. Schulteis, P.E. Dupont, P.A. Millman, R.D. Howe: Automatic identification of remote environments, Proc. ASME Dyn. Syst. Contr. Div. (Atlanta 1996) pp. 451–458
26.93 A.O. Farahat, B.S. Graves, J.C. Trinkle: Identifying contact formations in the presence of uncertainty, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (Pittsburg 1995) pp. 59–64
26.94 J. Xiao, L. Zhang: Contact constraint analysis and determination of geometrically valid contact formations from possible contact primitives, IEEE Trans. Robot. Autom. **13**(3), 456–466 (1997)
26.95 H. Mosemann, T. Bierwirth, F.M. Wahl, S. Stoeter: Generating polyhedral convex cones from contact graphs for the identification of assembly process states, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (2000) pp. 744–749
26.96 J. Xiao, L. Zhang: Towards obtaining all possible contacts – growing a polyhedron by its location uncertainty, IEEE Trans. Robot. Autom. **12**(4), 553–565 (1996)
26.97 M. Spreng: A probabilistic method to analyze ambiguous contact situations, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Atlanta 1993) pp. 543–548
26.98 N. Mimura, Y. Funahashi: Parameter identification of contact conditions by active force sensing, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Diego 1994) pp. 2645–2650
26.99 B. Eberman: A model-based approach to Cartesian manipulation contact sensing, Int. J. Robot. Res. **16**(4), 508–528 (1997)
26.100 T. Debus, P. Dupont, R. Howe: Contact state estimation using multiple model estimation and hidden Markov model, Int. J. Robot. Res. **23**(4–5), 399–413 (2004)
26.101 J. De Geeter, H. Van Brussel, J. De Schutter, M. Decréton: Recognizing and locating objects with local sensors, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Minneapolis 1996) pp. 3478–3483
26.102 J. De Schutter, H. Bruyninckx, S. Dutré, J. De Geeter, J. Katupitiya, S. Demey, T. Lefebvre: Estimating first-order geometric parameters and monitoring contact transitions during force-controlled compliant motions, Int. J. Robot. Res. **18**(12), 1161–1184 (1999)
26.103 T. Lefebvre, H. Bruyninckx, J. De Schutter: Polyhedral contact formation identification for autonomous compliant motion: exact nonlinear bayesian filtering, IEEE Trans Robot. **21**(1), 124–129 (2005)
26.104 T. Lefebvre, H. Bruyninckx, J. De Schutter: Online statistical model recognition and state estimation for autonomous compliant motion systems, IEEE Trans. Syst. Man Cybern. Part C Special Issue on "Pattern Recognition for Autonomous Manipulation in Robotic Systems" **35**(1), 16–29 (2005)

26.105 K. Gadeyne, T. Lefebvre, H. Bruyninckx: Bayesian hybrid model-state estimation applied to simultaneous contact formation recognition and geometrical parameter estimation, Int. J. Robot. Res. **24**(8), 615–630 (2005)
26.106 K. Kitagaki, T. Ogasawara, T. Suehiro: Methods to detect contact state by force sensing in an edge mating task, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Atlanta 1993) pp. 701–706
26.107 T. Lefebvre, H. Bruyninckx, J. De Schutter: Task planning with active sensing for autonomous compliant motion, Int. J. Robot. Res. **24**(1), 61–82 (2005)
26.108 T. Debus, P. Dupont, R. Howe: Distinguishability and identifiability testing of contact state models, Adv. Robot. **19**(5), 545–566 (2005)
26.109 J. Park, R. Cortesao, O. Khatib: Multi-contact compliant motion control for robotic manipulators, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (New Orleans 2004) pp. 4789–4794
26.110 W. Meeussen, J. De Schutter, H. Bruyninckx, J. Xiao, E. Staffetti: Integration of planning and execution in force controlled compliant motion, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (2005)
26.111 D.E. Koditschek: Exact robot navigation by means of potential functions: some topological considerations, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Raleigh 1987) pp. 1–6
26.112 E. Rimon, D.E. Koditschek: Exact robot navigation using artificial potential fields, IEEE Trans. Robot. Autom. **8**(5), 501–518 (1992)
26.113 E. Rimon, D.E. Koditschek: The construction of analytic diffeomorphisms for exact robot navigation on star worlds, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Scottsdale 1989) pp. 21–26
26.114 J. Barraquand, J.-C. Latombe: Robot motion planning: a distributed representation approach, Int. J. Robot. Res. **10**(6), 628–649 (1991)
26.115 C.I. Connolly, J.B. Burns, R. Weiss: Path planning using Laplace's equation, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Cincinnati 1990) pp. 2102–2106
26.116 C.I. Connolly, R.A. Grupen: One the applications of harmonic functions to robotics, J. Robot. Syst. **10**(7), 931–946 (1993)
26.117 S.H.J. Feder, E.J.-J. Slotine: Real-time path planning using harmonic potentials in dynamic environments, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Albuquerque 1997) pp. 811–874
26.118 J.-O. Kim, P. Khosla: Real-time obstacle avoidance using harmonic potential functions, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Sacramento 1991) pp. 790–796
26.119 K. Sato: Collision avoidance in multi-dimensional space using Laplace potential, Proc. 15th Conf. Robot. Soc. Jpn. (1987), 155–156
26.120 S.M. LaValle, P. Konkimalla: Algorithms for computing numerical optimal feedback motion strategies, Int. J. Robot. Res. **20**(9), 729–752 (2001)
26.121 A. van der Schaft, H. Schumacher: *An Introduction to Hybrid Dynamical Systems* (Springer, Belin, Heidelberg 2000)
26.122 W. Choi, J.-C. Latombe: A reactive architecture for planning and executing robot motions with incomplete knowledge, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS), Vol. 1 (Osaka 1991) pp. 24–29
26.123 D. Conner, H. Choset, A. Rizzi: Integrated planning and control for convex-bodied nonholonomic systems using local feedback control policies, Proc. Robot.: Sci. Syst. (Philadelphia 2006)
26.124 D.C. Conner, A.A. Rizzi, H. Choset: Composition of local potential functions for global robot control and navigation, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (Las Vegas 2003) pp. 3546–3551
26.125 S.R. Lindemann, S.M. LaValle: Smooth feedback for car-like vehicles in polygonal environments, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Rome 2007)
26.126 L. Yang, S.M. LaValle: The sampling-based neighborhood graph: a framework for planning and executing feedback motion strategies, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Taipei 2003)
26.127 C. Belta, V. Isler, G.J. Pappas: Discrete abstractions for robot motion planning and control in polygonal environments, IEEE Trans. Robot. Autom. **21**(5), 864–871 (2005)
26.128 R.R. Burridge, A.A. Rizzi, D.E. Koditschek: Sequential composition of dynamically dexterous robot behaviors, Int. J. Robot. Res. **18**(6), 534–555 (1999)
26.129 S.R. Lindemann, S.M. LaValle: Computing smooth feedback plans over cylindrical algebraic decompositions, Proc. Robot.: Sci. Syst. (RSS) (Philadephia 2006)
26.130 Y. Yang, O. Brock: Elastic roadmaps: globally task-consitent motion for autonomous mobile manipulation, Proc. Robot.: Sci. Syst. (RSS) (Philadelphia 2006)
26.131 S. Quinlan, O. Khatib: Elastic bands: connecting path planning and control, Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Vol. 2 (Atlanta 1993) pp. 802–807
26.132 M. Khatib, H. Jaouni, R. Chatila, J.-P. Laumond: How to implement dynamic paths, Proc. Int. Symp. Exp. Robot. (1997) pp. 225–236, Preprints
26.133 O. Brock, O. Khatib: Elastic strips: a framework for motion generation in human environments, Int. J. Robot. Res. **21**(12), 1031–1052 (2002)
26.134 M.H. Raibert, J.J. Craig: Hybrid position/force control of manipulators, J. Dyn. Syst. Meas. Contr. **103**(2), 126–133 (1981)
26.135 R. Featherstone, S. Sonck, O. Khatib: A general contact model for dynamically-decoupled force/motion control, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Detroit 1999) pp. 3281–3286
26.136 J. Park, R. Cortes ao, O. Khatib: Multi-contact compliant motion control for robotic manipulators, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (New Orleans 2004) pp. 4789–4794

26.137 J. Park, O. Khatib: Multi-link multi-contact force control for manipulators, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Barcelona 2005)

26.138 D.C. Ruspini, O. Khatib: A framework for multi-contact multi-body dynamic simulation and haptic display, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (Takamatsu 2000) pp. 1322–1327

26.139 K.-S. Chang, R. Holmberg, O. Khatib: The augmented object model: cooperative manipluation and parallel mechanism dynamics, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Francisco 2000) pp. 470–475

26.140 O. Khatib: Object Manipulation in a Multi-Effector Robot System. In: *Robotics Research 4*, ed. by R. Bolles, B. Roth (MIT Press, Cambridge 1988) pp. 137–144

26.141 D. Williams, O. Khatib: The virtual linkage: a model for internal forces in multi-grasp manipulation, Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Vol. 1 (Altanta 1993) pp. 1030–1035

26.142 J.A. Adams, R. Bajcsy, J. Kosecka, V. Kuma, R. Mandelbaum, M. Mintz, R. Paul, C. Wang, Y. Yamamoto, X. Yun: Cooperative material handling by human and robotic agents: module development and system synthesis, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (Pittsburgh 1995) pp. 200–205

26.143 S. Hayati: Hybrid postiion/force control of multi-arm cooperating robots, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Francisco 1986) pp. 82–89

26.144 D. Jung, G. Cheng, A. Zelinsky: Experiments in realizing cooperation between autonomous mobile robots, Proc. Int. Symp. Exp. Robot. (1997) pp. 513–524

26.145 T.-J. Tarn, A.K. Bejczy, X. Yun: Design of dynamic control of two cooperating robot arms: Closed chain formulation, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1987) pp. 7–13

26.146 M. Uchiyama, P. Dauchez: A symmetric hybrid position/force control scheme for the coordination of two robots, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Philadelphia 1988) pp. 350–356

26.147 X. Yun, V.R. Kumar: An approach to simultaneious control fo trajecotry and integraction forces in dual-arm configurations, IEEE Trans. Robot. Autom. **7**(5), 618–625 (1991)

26.148 Y.F. Zheng, J.Y.S. Luh: Joint torques for control of two coordinated moving robots, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Francisco 1986) pp. 1375–1380

26.149 J. Russakow, O. Khatib, S.M. Rock: Extended operational space formation for serial-to-parallel chain (branching) manipulators, Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Vol. 1 (Nagoya 1995) pp. 1056–1061

26.150 K.-S. Chang, O. Khatib: Operational space dynamics: efficient algorithms for modelling and control of branching mechanisms, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (San Francisco 2000) pp. 850–856

26.151 K. Kreutz-Delgado, A. Jain, G. Rodriguez: Recursive formulation of operational space control, Int. J. Robot. Res. **11**(4), 320–328 (1992)

26.152 B. Bayle, J.-Y. Fourquet, M. Renaud: A coordination strategy for mobile manipulation, Proc. Int. Conf. Intell. Auton. Syst. (Venice 2000) pp. 981–988

26.153 B. Bayle, J.-Y. Fourquet, M. Renaud: Generalized path generation for a mobile manipulator, Proc. Int. Conf. Mech. Des. Prod. (Cairo 2000) pp. 57–66

26.154 B. Bayle, J.-Y. Fourquet, M. Renaud: Using manipulability with nonholonomic mobile manipulators, Proc. Int. Conf. Field Serv. Robot. (Helsinki 2001) pp. 343–348

26.155 J. Peters, S. Schaal: Reinforcement learning for operational space control, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (Rome 2007)

26.156 T. Simeon, J. Cortes, A. Sahbani, J.P. Laumond: A manipulation planner for pick and place operations under continuous grasps and placements, Proc. IEEE Int. Conf. Robot. Autom. (2002)

26.157 A. Miller, P. Allen: Examples of 3D grasp quality computations, Robot. Autom. 1999. Proc. 1999 IEEE Int. Conf. on, Vol. 2 (1999)

26.158 A.T. Miller: GraspIt: A Versatile Simulator for Robotic Grasping. Ph.D. Thesis (Department of Computer Science, Columbia University 2001)

26.159 G. Wilfong: Motion panning in the presence of movable obstacles, Proc. ACM Symp. Computat. Geom. (1988) pp. 279–288

26.160 M. Erdmann, T. Lozano-Perez: On multiple moving objects, IEEE Int. Conf. Robot. Autom. (San Francisco 1986) pp. 1419–1424

26.161 P.C. Chen, Y.K. Hwang: Pracitcal path planning among movable obstacles, Proc. IEEE Int. Conf. Robot. Autom. (1991) pp. 444–449

26.162 M. Stilman, J.J. Kuffner: Navigation among movable obstacles: real-time reasoning in complex environments, Int. J. Human Robot. **2**(4), 1–24 (2005)

26.163 M. Stilman, J.-U. Shamburek, J.J. Kuffner, T. Asfour: Manipulation planning among movable obstacles, Proc. IEEE Int. Conf. Robot. Autom. (ICRA'07) (2007)

26.164 D. Nieuwenhuisen, A.F. van der Stappen, M.H. Overmars: An effective framework for path planning amidst movable obstacles, Workshop Algorithm. Fund. Robot. (2006)

26.165 K.M. Lynch, M.T. Mason: Stable pushing: mechanics, controllability, and planning, Int. J. Robot. Res. **15**(6), 533–556 (1996)

26.166 P. Tang, J. Xiao: Generation of point-contact state space between strictly curved objects. In: *Robotics Science and Systems II*, ed. by G.S. Sukhatme, S. Schaal, W. Burgard, D. Fox (MIT Press, Cambridge 2007) pp. 239–246

26.167 H. Nakagaki, K. Kitagaki, T. Ogasawara, H. Tsukune: Study of deformation and insertion tasks of a flexible wire, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1997) pp. 2397–2402

26.168 W. Kraus Jr., B.J. McCarragher: Case studies in the manipulation of flexible parts using a hybrid position/force approach, Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (1997) pp. 367–372

26.169 J.Y. Kim, D.J. Kang, H.S. Cho: A flexible parts assembly algorithm based on a visual sensing system, Proc. Int. Symp. Assem. Task Plan. (2001) pp. 417–422

26.170 B.J. Unger, A. Nocolaidis, P.J. Berkelman, A. Thompson, R.L. Klatzky, R.L. Hollis: Comparison of 3-D haptic peg-in-hole tasks in real and virtual environments, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (2001) pp. 1751–1756