Obstacle Avoidance for Redundant Manipulators as Control Problem

Chapter · March 2012

DOI: 10.5772/32651 · Source: InTech

2 authors:

Leon Zlajpah
Jožef Stefan Institute
111 PUBLICATIONS **915** CITATIONS
SEE PROFILE

Tadej Petric
Jožef Stefan Institute
79 PUBLICATIONS **726** CITATIONS
SEE PROFILE

Obstacle Avoidance for Redundant Manipulators as Control Problem

Leon Žlajpah and Tadej Petrič
Jožef Stefan Institute, Ljubljana
Slovenia

1. Introduction

One of the goals of robotics research is to provide control algorithms that allow robotic manipulators to move in an environment with objects. The contacts with these objects may be part of the task, e.g. in the assembly operations, or they may be undesired events. If the task involves some contacts with the environment it is necessary to control the resulting forces. For that purpose, different control approaches have been proposed like hybrid position/force control (Raibert & Craig, 1981) or impedance control (Hogan, 1985), which have also been applied to redundant manipulators (Khatib, 1987; Park et al., 1996; Woernle, 1993; Yoshikawa, 1987). However, in most cases the contact is supposed to occur between the end-effector or the handling object and the object in the workspace. Except in some special cases all other contacts along the body of the robot manipulator are not desired and have to be avoided. In the case when the contacts are not desired, the main issue is how to accomplish the assigned task without any risk of collisions with the workspace objects. A natural strategy to avoid obstacles would be to move the manipulator away from the obstacle into a configuration where the manipulator is not in contact with the obstacle. Without changing the motion of the end-effector, the reconfiguration of the manipulator into a collision-free configuration can be made only if the manipulator has redundant degrees-of-freedom (DOF). The flexibility depends on the degree-of-redundancy, i.e., on the number of redundant DOF. A high degree-of-redundancy is important, especially when the manipulator is working in an environment with many potential collisions with obstacles.
Generally, the obstacle-avoidance (or collision-avoidance) problem can be solved with two classes of strategies: global (planning) and local (control). The global ones, like high-level path planning, guarantee to find a collision-free path from the initial point to the goal point, if such a path exists. They often operate in the configuration space into which the manipulator and all the obstacles are mapped and a collision-free path is found in the unoccupied portion of the configuration space (Lozano-Perez, 1983). However, these algorithms are very computationally demanding and the calculation times are significantly longer than the typical response time of a manipulator. This computational complexity limits their use for practical obstacle avoidance just to simple cases. Furthermore, as global methods do not usually rely on any sensor feedback information, they are only suitable for static and well-defined environments. On the other hand, local strategies treat obstacle avoidance as a control problem. Their aim is not to replace the higher-level, global, collision-free path planning but to make use of the capabilities of low-level control, e.g., they can use the sensor information

to change the path if the obstacle appears in the workspace or it moves. Hence, they are suitable when the obstacle position is not known in advance and must be detected in real-time during the task execution. A significant advantage of local methods is that they are less computationally demanding and more flexible. These characteristics make local methods good candidates for on-line collision avoidance, especially in unstructured environments. However, the drawback is that they may cause globally suboptimal behavior or may even get stuck when a collision-free path cannot be found from the current configuration.

With the problem of the collision avoidance of redundant manipulators, there have been different approaches to the local methods proposed by many researchers in the past (Brock et al., 2002; Colbaugh et al., 1989; Glass et al., 1995; Guo & Hsia, 1993; Khatib, 1986; Kim & Khosla, 1992; Maciejewski & Klein, 1985; McLean & Cameron, 1996; Seraji & Bon, 1999; Volpe & Khosla, 1990). The approach proposed by Maciejewski and Klein (Maciejewski & Klein, 1985) is to assign to the critical point an avoiding task space vector, with which the point is directed away from the obstacle. Colbaugh, Glass and Seraji (Colbaugh et al., 1989; Glass et al., 1995) used configuration control and defined the constraints representing the obstacle-avoidance. The next approach is based on potential functions, where a repulsive potential is assigned to the obstacles and an attractive potential to the goal position (Khatib, 1986; Kim & Khosla, 1992; McLean & Cameron, 1996; Volpe & Khosla, 1990). The fourth approach uses the optimization of an objective function maximizing the distance between the manipulator and the obstacles (Guo & Hsia, 1993). Most of the proposed methods solve the obstacle-avoidance problem at the kinematic level. Velocity null-space control is an appropriate way to control the internal motion of a redundant manipulator. Some of the control strategies are acceleration based or torque based, considering also the manipulator dynamics (Brock et al., 2002; Khatib, 1986; Newman, 1989; Xie et al., 1998). However, it is established that certain acceleration-based control schemes exhibit instabilities (O'Neil, 2002). An alternative approach is the augmented Jacobian, as introduced in Egeland (1987), where the secondary task is added to the primary task so as to obtain a square Jacobian matrix that can be inverted. The main drawback of this technique are the so-called algorithmic singularities. They occur when the secondary task causes a conflict with the primary task. Khatib investigated in depth the use of the second-order inverse kinematic, either at the torque or acceleration level, starting from Khatib (1987) to recent task-prioritised humanoid applications (Mansard et al., 2009; Sentis et al., 2010).

Here, we want to present some approaches to on-line obstacle-avoidance for the redundant manipulators at the kinematic level and some approaches, which are considering also the dynamics of the manipulator.

Like in most of the local strategies that solve the obstacle-avoidance problem at the kinematic level (Colbaugh et al., 1989; Glass et al., 1995; Guo & Hsia, 1993; Kim & Khosla, 1992; Maciejewski & Klein, 1985; Seraji & Bon, 1999), the aim of the proposed strategies is to assign each point on the body of the manipulator, which is close to the obstacle, a motion component in a direction away from the obstacle. The emphasis is given to the definition of the avoiding motion. Usually, the avoiding motion is defined in the Cartesian space. As obstacle avoidance is typically a one-dimensional problem, we use a one-dimensional operational space for each critical point. Consequently, some singularity problems can be avoided when not enough "redundancy" is available locally. Additionally, we propose an approximative calculation of the motion that is faster that the exact one. Another important issue addressed in this paper is how the obstacle avoidance is performed when there are more simultaneously active obstacles in the neighborhood of the manipulator. We propose an algorithm that considers all the obstacles in the neighborhood of the robot.

Most tasks performed by a redundant manipulator are broken down into several subtasks with different priorities. Usually, the task with the highest priority, referred to as the main task, is associated with the positioning of the end-effector in the task space, and other subtasks are associated with the obstacle avoidance and other additional tasks (if the degree-of-redundancy is high enough). However, in some cases it is necessary (e.g., for safety reasons) that the end-effector motion is not the primary task. As, in general, task-priority algorithms do not always allow simple transitions or the changing of priority levels between the tasks (Sciavicco & Siciliano, 2005), we propose a novel formulation of the primary and secondary tasks, so that the desired movement of the end-effector is in fact a secondary task. The primary task is now the obstacle avoidance and it is only active if we approach a pre-defined threshold, i.e., the critical distance to one of the obstacles. While far from the threshold, our algorithm allows undisturbed control of the secondary task. If we approach the threshold, the primary task smoothly takes over and only allows joint control in the null-space of the primary task. A similar approach can also be found in Sugiura et al. (2007), where they used a blending coefficient for blending the end-effector motion with the obstacle-avoidance motion, and in Mansard et al. (2009) where they proposed a generic solution to build a smooth control law for any kind of unilateral constraints.

Next, we propose strategies that are also considering the dynamics. In this case, it is reasonable to define the obstacle avoidance at the force level, i.e., the forces are supposed to generate the motion that is necessary to avoid the obstacle. We discuss three approaches regarding the sensors used to detect the obstacles: no sensors, tactile sensors and proximity sensors or vision. First of all, we want to investigate what happens if the manipulator touches an obstacle, especially how to control the contact forces and how to avoid the obstacle after the contact. Therefore, we propose a strategy that utilizes the self-motion caused by the contact forces to avoid an obstacle after the collision. The main advantage of this strategy is that it can be applied to the systems without any contact or force sensors. However, a prerequisite for this strategy to be effective is that the manipulator is backdrivable. As an alternative for stiff systems (having high-ratio gears, high friction, etc.), we propose using tactile sensors. Finally, we deal with proximity sensors and we propose a virtual forces strategy, where a virtual force component in a direction away from the obstacle is assigned to each point on the body of the manipulator, which is close to an obstacle. Like other classical methods for obstacle avoidance this one prevents any part of the manipulator touching an obstacle. Also here we address the problem of multiple obstacles in the workspace, which have to be simultaneously avoided.

The computational efficiency of all the proposed algorithms (at the kinematic level and considering the dynamics) allows the real-time application in an unstructured or time-varying environment. The efficiency of the proposed control algorithms is illustrated by simulations of highly redundant planar manipulator moving in an unstructured and time-varying environment and by experiments on a real robot manipulator.

2. Background

The robotic systems under study are serial manipulators. We consider redundant systems, i.e., the dimension of the joint space n exceeds the dimension of the task space m. The difference between n and m will be denoted as the degree of redundancy r, $r = n - m$. Note that in this definition the redundancy is not only a characteristics of the manipulator itself but also of the task. This means that a nonredundant manipulator may also become a redundant manipulator for a certain task.

2.1 Modelling

Let the configuration of the manipulator be represented by the n-dimensional vector $\boldsymbol{q}$ of joint positions, and the end-effector position (and orientation) by the m-dimensional vector $\boldsymbol{x}$ of the task positions (and orientations). Then, the kinematics can be described by the following equations

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}) \qquad \dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}} + \mathbf{N}\dot{\boldsymbol{q}} \qquad \ddot{\boldsymbol{q}} = \mathbf{J}^{\#}(\ddot{\boldsymbol{x}} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \mathbf{N}\ddot{\boldsymbol{q}} \tag{1}$$

where $\boldsymbol{f}$ is an m-dimensional vector function representing the manipulator's forward kinematics, $\mathbf{J}$ is the $m \times n$ manipulator's Jacobian matrix, $\mathbf{J}^{\#}$ is the generalized inverse of the Jacobian matrix $\mathbf{J}$ and $\mathbf{N}$ is a matrix representing the projection into the null space of $\mathbf{J}$, $\mathbf{N} = (\mathbf{I} - \mathbf{J}^{\#}\mathbf{J})$.

For the redundant manipulators the static relationship between the m-dimensional generalized force in the task space $\boldsymbol{F}$, and the corresponding n-dimensional generalized joint space force $\boldsymbol{\tau}$ is

$$\boldsymbol{\tau} = \mathbf{J}^T\boldsymbol{F} + \mathbf{N}^T\boldsymbol{\tau} \tag{2}$$

where $\mathbf{N}^T$ is a matrix representing the projection into the null space of $\mathbf{J}^{T\#}$.

Assuming the manipulator consists of rigid bodies the joint space equations of motion can be written in the form

$$\boldsymbol{\tau} = \mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \boldsymbol{\tau}_F \tag{3}$$

where $\boldsymbol{\tau}$ is an n-dimensional vector of control torques, $\mathbf{H}$ is an $n \times n$ inertia matrix, $\boldsymbol{h}$ is an n-dimensional vector of Coriolis and centrifugal forces, $\boldsymbol{g}$ is an n-dimensional vector of gravity forces, and the vector $\boldsymbol{\tau}_F$ represents torques due to external forces acting on the manipulator.

2.2 Kinematic control

For velocity control the following kinematic controller can be used

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}}_c + \mathbf{N}\dot{\boldsymbol{\varphi}} \tag{4}$$

where $\dot{\boldsymbol{x}}_c$ and $\dot{\boldsymbol{\varphi}}$ represent the task space control law and the arbitrary joint velocities, respectively. The task space control $\dot{\boldsymbol{x}}_c$ can be selected as

$$\dot{\boldsymbol{x}}_c = \dot{\boldsymbol{x}}_e + \mathbf{K}_p\boldsymbol{e} \tag{5}$$

where $\boldsymbol{e}$, $\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}$, is the tracking error, $\dot{\boldsymbol{x}}_e$ is the desired task space velocity, and $\mathbf{K}_p$ is a constant gain matrix. To perform the additional subtask, the velocity $\dot{\boldsymbol{\varphi}}$ is used. Let p be a function representing the desired performance criterion. Then, to optimize p we can select $\dot{\boldsymbol{\varphi}}$ as

$$\dot{\boldsymbol{\varphi}} = \mathbf{K}_p\nabla p \tag{6}$$

Here, ∇p is the gradient of p and $\mathbf{K}_p$ is a gain.

2.3 Torque control

To decouple the task space and null-space motion we propose using a controller given in the form

$$\boldsymbol{\tau}_c = \mathbf{H}(\mathbf{J}^{\#}(\ddot{\boldsymbol{x}}_c - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \mathbf{N}(\boldsymbol{\phi} - \dot{\mathbf{N}}\dot{\boldsymbol{q}})) + \boldsymbol{h} + \boldsymbol{g} \tag{7}$$

where $\ddot{\boldsymbol{x}}_c$ and $\boldsymbol{\phi}$ represent the task space and the null-space control law, respectively. The task space control $\ddot{\boldsymbol{x}}_c$ can be selected as

$$\ddot{\boldsymbol{x}}_c = \ddot{\boldsymbol{x}}_d + \mathbf{K}_v \dot{\boldsymbol{e}} + \mathbf{K}_p \boldsymbol{e} \tag{8}$$

where $\boldsymbol{e}$, $\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}$, is the tracking error, $\ddot{\boldsymbol{x}}_d$ is the desired task space acceleration, and $\mathbf{K}_v$ and $\mathbf{K}_p$ are constant gain matrices. The selection of $\mathbf{K}_v$ and $\mathbf{K}_p$ can be based on the desired task space impedance. To perform the additional subtask, the vector $\boldsymbol{\phi}$ is given in the form

$$\boldsymbol{\phi} = \mathbf{N}\ddot{\boldsymbol{\varphi}} + \dot{\mathbf{N}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n \dot{\boldsymbol{e}}_n, \qquad \dot{\boldsymbol{e}}_n = \mathbf{N}(\dot{\boldsymbol{\varphi}} - \dot{\boldsymbol{q}}) \tag{9}$$

where $\dot{\boldsymbol{\varphi}}$ is the desired null space velocity and $\mathbf{K}_n$ is an $n \times n$ diagonal gain matrix. The velocity $\dot{\boldsymbol{\varphi}}$ is defined by the subtask. E.g., let p be a function representing the desired performance criterion. To optimize p we can select $\dot{\boldsymbol{\varphi}}$ as (6).

3. Obstacle-avoidance strategy

The obstacle-avoidance problem is usually defined as how to control the manipulator to track the desired end-effector trajectory while simultaneously ensuring that no part of the manipulator collides with any obstacle in the workspace of the manipulator. To avoid any obstacles the manipulator has to move away from the obstacles into the configuration where the distance to the obstacles is larger (see Fig. 1). Without changing the motion of the end-effector, the reconfiguration of the manipulator into a collision-free configuration can be done only if the manipulator has redundant degrees-of-freedom (DOF). Note that the flexibility or the *degree-of-redundancy* of the manipulator does not depend only on the number of redundant DOF but also on the "location" of the redundant DOF. Namely, it is possible that the redundant manipulator cannot avoid an obstacle, because it is in a configuration where the avoiding motion in the desired direction is not possible. A high degree-of-redundancy is important, especially when the manipulator is working in an environment with many potential collisions with obstacles.
A good strategy for obstacle avoidance is to identify the points on the robotic arm that are near obstacles and then assign to them a motion component that moves those points away from the obstacle, as shown in Fig. 1. The motion of the robot is perturbed only if at least one part of the robot is in the critical neighborhood of an obstacle, i.e., the distance is less than a prescribed minimal distance. We denote the obstacles in the critical neighborhood as the *active obstacles* and the corresponding closest points on the body of a manipulator as the *critical points*. Usually, it is assumed that the motion of the end effector is not disturbed by any obstacle. Otherwise, the task execution has to be interrupted and the higher-level path planning has to recalculate the desired motion of the end-effector. If the path-tracking accuracy is not important control algorithms which move the end-effector around obstacles on-line, can be used.
As the obstacle avoidance is supposed to be done on-line, it is not necessary to know the exact position of the obstacles in advance. Of course, to allow the manipulator to work in an unstructured and/or dynamic environment, some sensors have to be used to determine the position of the obstacles or measure the distance between the obstacles and the body of the manipulator. There are different types of sensor systems that can be used to detect objects in the neighborhood of the manipulator. They can be tactile or proximity sensors. The tactile sensors, like artificial skin, can detect the obstacle only if they touch it. On the other hand, the proximity sensors can sense the presence of an obstacle in the neighborhood. We have

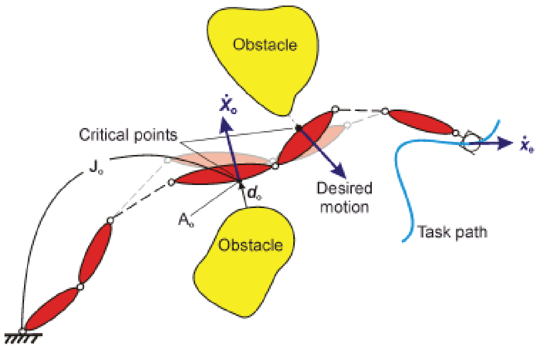


Fig. 1. Manipulator motion in presence of some obstacles

compared the capabilities of a manipulator equipped with both types of sensors. Actually, we have also investigated if and how the manipulator can avoid obstacles without any sensors for the detection of obstacles. As in the case of tactile sensors and when no sensors are used the manipulator has to "touch" the obstacles, we allow a collision with an obstacle. However, after the collision the manipulator should move away from the obstacle and the collision forces should be kept as low as possible.

4. Obstacle avoidance using kinematic control

The proposed velocity strategy considers the obstacle-avoidance problem at the kinematic level. Let $\dot{\boldsymbol{x}}_e$ be the desired velocity of the end-effector, and A_o be the critical point in the neighborhood of an obstacle (see Fig. 1). To avoid a possible collision, one possibility is to assign to A_o such a velocity that it moves away from the obstacle, as proposed in Maciejewski & Klein (1985). Hence, the motion of the end-effector and the critical point can be described by the equations

$$\mathbf{J}\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}_e \qquad\qquad \mathbf{J}_o\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}_o \tag{10}$$

where $\mathbf{J}_o$ is a Jacobian matrix associated with the point point A_o. There are some possibilities to find a common solution for both equations.

4.1 Exact solution

Let $\dot{\boldsymbol{x}}_c$ in (4) equal $\dot{\boldsymbol{x}}_e$. Then, combining (4) and (10) yields

$$\dot{\boldsymbol{\varphi}} = (\mathbf{J}_o\mathbf{N})^{\#}(\dot{\boldsymbol{x}}_o - \mathbf{J}_o\mathbf{J}^{\#}\dot{\boldsymbol{x}}_e) \tag{11}$$

Now, using this $\dot{\boldsymbol{\varphi}}$ in (4) gives the final solution for $\dot{\boldsymbol{q}}$ in the form

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}}_c + (\mathbf{J}_o\mathbf{N})^{\#}(\dot{\boldsymbol{x}}_o - \mathbf{J}_o\mathbf{J}^{\#}\dot{\boldsymbol{x}}_e) \tag{12}$$

because $\mathbf{N}$ is both hermitian and idempotent (Maciejewski & Klein, 1985; Nakamura et al., 1987). The meaning of the terms in the above equation can be easily explained. The first term $\mathbf{J}^{\#}\dot{\boldsymbol{x}}_c$ guarantees the joint motion necessary for the desired end-effector velocity. Also, $\dot{\boldsymbol{x}}_c$ is used in (12) instead of $\dot{\boldsymbol{x}}_e$ to indicate that a task space controller can be used to compensate for any task space tracking errors, e.q.

$$\dot{\boldsymbol{x}}_c = \dot{\boldsymbol{x}}_d + \mathbf{K}\boldsymbol{e}. \tag{13}$$

where $\dot{\boldsymbol{x}}_d$ is the desired task space velocity, $\mathbf{K}$ is an m x m positive-definite matrix and $\boldsymbol{e}$ is the task position error, defined as

$$\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}, \tag{14}$$

where $\boldsymbol{x}_d$ is the desired task space position. The second term in (12), i.e., the homogeneous solution $\dot{\boldsymbol{q}}_h$, represents the part of the joint velocity causing the motion of the point A_o. The term $\mathbf{J}_o\mathbf{J}^{\#}\dot{\boldsymbol{x}}_e$ is the velocity in A_o due to the end-effector motion. The matrix $\mathbf{J}_o\mathbf{N}$ is used to transform the desired critical point velocity from the operational space of the critical point into the joint space. Note that the above solution guarantees that we achieve exactly the desired $\dot{\boldsymbol{x}}_o$ only if the degree of redundancy of the manipulator is sufficient.

4.2 Exact solution using a reduced operational space

The matrix $\mathbf{J}_o\mathbf{N}$ combines the kinematics of the critical point A_o and the null-space matrix of the whole manipulator. Hence, its properties define the flexibility of the system for avoiding the obstacles. We want to point out that the properties of the matrix $\mathbf{J}_o\mathbf{N}$ do not depend only on the position of the point A_o but also on the definition of the operational space associated with the critical point. Usually, it is assumed that all critical points belong to the Cartesian space. Hence, the velocity $\dot{\boldsymbol{x}}_o$ is a 3-dimensional vector and the dimension of the matrix $\mathbf{J}_o\mathbf{N}$ is $3 \times n$. This also implies that 3 degrees-of-redundancy are needed to move one point from an obstacle. Consequently, it seems that a manipulator with 2 degrees-of-redundancy is not capable of avoiding obstacles, and of course, this is not true. For example, consider a planar 3 DOF manipulator that is supposed to move along a line, as shown in Fig. 2. As this is a planar case, the task space is 2-dimensional (e.g. x and y) and the manipulator has 1 degree-of-redundancy. Defining the velocity $\dot{\boldsymbol{x}}_o$ in the same space as the end-effector velocity, i.e., as a 2-dimensional vector, reveals the matrix $\mathbf{J}_o\mathbf{N}$ to have the dimension 2×3. Furthermore, due to 1 degree-of-redundancy the components of the velocity vector $\dot{\boldsymbol{x}}_o$ are not independent. Hence, the rank of $\mathbf{J}_o\mathbf{N}$ is 1, and the pseudoinverse $(\mathbf{J}_o\mathbf{N})^{\#}$ does not exist.
As the obstacle-avoidance strategy only requires motion in the direction of the line connecting the critical point with the closest point on the obstacle, this is a one-dimensional constraint and only one degree-of-redundancy is needed to avoid the obstacle, generally. Therefore, we propose using a reduced operational space for the obstacle avoidance and define the Jacobian $\mathbf{J}_o$ as follows.
Let $\boldsymbol{d}_o$ be the vector connecting the closest points on the obstacle and the manipulator (see Fig. 1) and let the operational space in A_o be defined as one-dimensional space in the direction of $\boldsymbol{d}_o$. Then, the Jacobian, which relates the joint space velocities $\dot{\boldsymbol{q}}$ and the velocity in the direction of $\boldsymbol{d}_o$, can be calculated as

$$\mathbf{J}_{d_o} = \boldsymbol{n}_o^T \mathbf{J}_o \tag{15}$$

where $\mathbf{J}_o$ is the Jacobian defined in the Cartesian space and $\boldsymbol{n}_o$ is the unit vector in the direction of $\boldsymbol{d}_o$, $\boldsymbol{n}_o = \frac{\boldsymbol{d}_o}{\|\boldsymbol{d}_o\|}$. Now, the dimension of the matrix $\mathbf{J}_{d_o}$ is $1 \times n$, and the velocities $\dot{\boldsymbol{x}}_o$ and $\mathbf{J}_{d_o}\mathbf{J}^{\#}\dot{\boldsymbol{x}}_e$ become scalars. Consequently, the computation of $(\mathbf{J}_{d_o}\mathbf{N})^{\#}$ is faster. For example, when calculating the Moore-Penrose pseudoinverse of $(\mathbf{J}_{d_o}\mathbf{N})$ defined as

$$(\mathbf{J}_{d_o}\mathbf{N})^{\#} = (\mathbf{J}_{d_o}\mathbf{N})^T \left(\mathbf{J}_{d_o}\mathbf{N}\,(\mathbf{J}_{d_o}\mathbf{N})^T\right)^{-1} = \mathbf{N}\mathbf{J}_{d_o}^T(\mathbf{J}_{d_o}\mathbf{N}\mathbf{J}_{d_o}^T)^{-1} \tag{16}$$

we do not have to invert any matrix because the term $(\mathbf{J}_{d_o}\mathbf{N}\mathbf{J}_{d_o}^T)$ is a scalar. Going back to our example in Fig. 2, the pseudoinverse $(\mathbf{J}_{d_o}\mathbf{N})^{\#}$ exists and the manipulator can perform the primary task and simultaneously avoid the obstacle, as shown in Fig. 2.

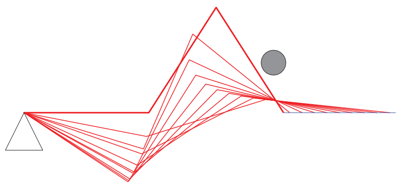


Fig. 2. Planar 3DOF manipulator: tracking of a line and obstacle avoidance using the Jacobian $\mathbf{J}_{d_o}$

4.3 Selection of avoiding velocity

The efficiency of the obstacle-avoidance algorithm also depends on the selection of the desired critical point velocity $\dot{x}_o$. We propose changing $\dot{x}_o$ with respect to the obstacle distance

$$\dot{x}_o = \alpha_v v_o \tag{17}$$

where v_o is the nominal velocity and α_v is the obstacle-avoidance gain defined as

$$\alpha_v = \begin{cases} \left(\frac{d_m}{\|\boldsymbol{d}_o\|}\right)^2 - 1 & \text{for} \quad \|\boldsymbol{d}_o\| < d_m \\ 0 & \text{for} \quad \|\boldsymbol{d}_o\| \geq d_m \end{cases} \tag{18}$$

where d_m is the critical distance to the obstacle (see Fig. 3). If the obstacle is too close ($\|\boldsymbol{d}_o\| \leq d_b$) the main task should be aborted. The distance d_b is subjected to the dynamic properties of the manipulator and can also be a function of the relative velocity $\dot{\boldsymbol{d}}_o$. To assure smooth transitions it is important that the magnitude of $\dot{\boldsymbol{x}}_o$ at d_m is zero. Special attention has to be given to the selection of the nominal velocity v_o. Large values of v_o would cause unnecessarily high velocities and consequently the manipulator would move far from the obstacle. Such motion may cause problems if there are more obstacles in the neighborhood of the manipulator. Namely, the manipulator may bounce between the obstacles. On the other hand, too small value of v_o would not move the critical point of the manipulator away from the obstacle.

For a smoother motion, was is proposed in Maciejewski & Klein (1985) to change the amount of the homogenous solution to be included in the total solution

$$\dot{\boldsymbol{q}}_{EX} = \mathbf{J}^{\#}\dot{\boldsymbol{x}}_c + \alpha_h(\mathbf{J}_{d_o}\mathbf{N})^{\#}(\dot{\boldsymbol{x}}_o - \mathbf{J}_{d_o}\mathbf{J}^{\#}\dot{\boldsymbol{x}}_e) \tag{19}$$

We have selected α_h as

$$\alpha_h = \begin{cases} 1 & \text{for} \quad \|\boldsymbol{d}_o\| \leq d_m \\ \frac{1}{2}(1 - \cos(\pi\frac{\|\boldsymbol{d}_o\| - d_m}{d_i - d_m})) & \text{for} \quad d_m < \|\boldsymbol{d}_o\| < d_i \\ 0 & \text{for} \quad d_i \leq \|\boldsymbol{d}_o\| \end{cases} \tag{20}$$

where d_i is the distance where the obstacle influences the motion. From Fig. 3 it can be seen that in the region between d_b and d_m the complete homogenous solution is included in the motion specification and the avoidance velocity is inversely related to the distance. Between d_m and d_i the avoidance velocity is zero and only a part of the homogenous solution is included. As the homogenous solution compensates for the motion in the critical point due to the end-effector motion, the relative velocity between the obstacle and the critical point

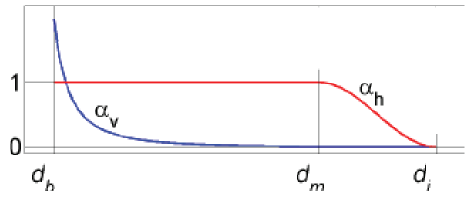


Fig. 3. The obstacle avoidance gain α_v and the homogenous term gain α_h versus the distance to the obstacle

decreases when approaching from d_i to d_m, if the obstacle is not moving, of course. With such a selection of α_v and α_h smooth velocities can be obtained.
The control law (19) can be used for a single obstacle. When more than one obstacle is active at one time then the worst-case obstacle (nearest) has to be used, which results in discontinuous velocities and may cause oscillations in some cases. Namely, when switching between active obstacles the particular homogenous solutions are not equal and a discontinuity in the joint velocities occurs. To improve the behavior we propose to use a weighted sum of the homogenous solution of all the active obstacles

$$\dot{\boldsymbol{q}}_{EX} = \mathbf{J}^{\#}\dot{\boldsymbol{x}}_c + \sum_{i=1}^{n_o} w_i \alpha_{h,i} \dot{\boldsymbol{q}}_{h,i} \tag{21}$$

where n_o is the number of active obstacles, and w_i, $\alpha_{h,i}$ and $\dot{\boldsymbol{q}}_{h,i}$ are the weighting factor, the gain and the homogenous solution for the i-th active obstacle, respectively. The weighting factors w_i are calculated as

$$w_i = \frac{d_i - \|\boldsymbol{d}_{o,i}\|}{\sum_{i=1}^{n_o}(d_i - \|\boldsymbol{d}_{o,i}\|)} \tag{22}$$

Although the actual velocities in the critical points differ from the desired ones, using $\dot{\boldsymbol{q}}_{EX}$ improves the behavior of the system and when one point is much closer to the obstacle than another, then its weight approaches 1 and the velocity in that particular point is close to the desired value.
As an illustration we present a simulation of a planar manipulator with 4 revolute joints. The primary task is to move along a straight line from point P_1 to point P_2, and the motion is obstructed by an obstacle. The task trajectory has a trapezoid velocity profile with an acceleration of $4ms^{-2}$ and a max. velocity of $0.4ms^{-1}$. We chose the the critical distance $d_m = 0.2m$. The simulation results using the exact velocity controller EX (19) are presented in Fig. 4(a).

4.4 Approximate solution

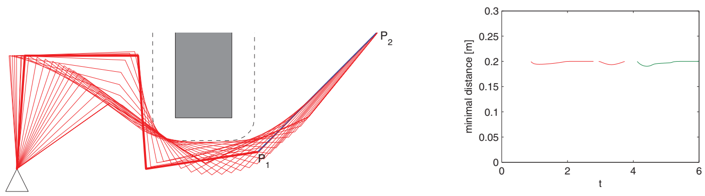
Another possible solution for $\dot{\varphi}$ is to calculate joint velocities that satisfy the secondary goal as

$$\dot{\varphi} = \mathbf{J}_{d_o}^{\#}\dot{x}_o \tag{23}$$

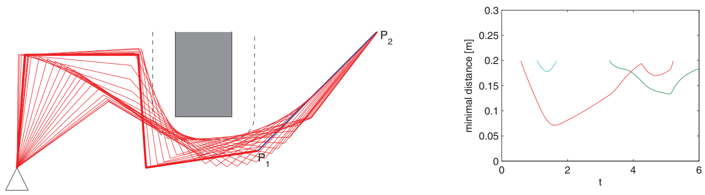
without compensating for the contribution of the end-effector motion and then substitute $\dot{\varphi}$ into (4), which yields

$$\dot{\boldsymbol{q}}_{AP} = \mathbf{J}^{\#}\dot{\boldsymbol{x}}_c + \mathbf{N}\mathbf{J}_{d_o}^{\#}\dot{x}_o \tag{24}$$

This approach avoids the singularity problem of $(\mathbf{J}_{d_o}\mathbf{N})$ (Chiaverini, 1997). The formulation (24) does not guarantee that we will achieve exactly the desired $\dot{x}_o$ even if the degree of redundancy is sufficient because $\mathbf{J}_{d_o}\mathbf{N}\mathbf{J}_{d_o}^{\#}\dot{x}_o$ is not equal to $\dot{x}_o$, in general.



a) EX controller



b) AP controller

Fig. 4. Planar 4DOF manipulator tracking a line while avoiding obstacles using velocity control (the dotted line indicates the critical distance)

To avoid the obstacle the goal velocity in A_o is represented by the vector $\dot{\boldsymbol{x}}_o$. Using the original method (12) the velocity in A_o is exactly $\dot{\boldsymbol{x}}_o$. The joint velocities $\dot{\boldsymbol{q}}_{EX}$ assure that the component of the velocity at point A_o (i.e., $\mathbf{J}_o\dot{\boldsymbol{q}}_{EX}$) in the direction of $\dot{\boldsymbol{x}}_o$ is as required. The approximate solution $\dot{\boldsymbol{q}}_{AP}$ gives, in most cases, a smaller magnitude of the velocity in the direction of $\dot{\boldsymbol{x}}_o$ (see $\mathbf{J}_o\dot{\boldsymbol{q}}_{AP}$). Therefore, the manipulator moves closer to the obstacle when $\dot{\boldsymbol{q}}_{AP}$ is used. This is not so critical, because the minimal distance also depends on the nominal velocity v_o, which can be increased to achieve larger minimal distances. Additionally, the approximate solution possesses certain advantages when many active obstacles have to be considered. The joint velocities can be calculated as

$$\dot{\boldsymbol{q}}_{AP} = \mathbf{J}^{\#}\dot{\boldsymbol{x}}_c + \mathbf{N}\sum_{i=1}^{n_o}\mathbf{J}^{\#}_{d_o,i}\dot{x}_{o,i} \tag{25}$$

where n_o is the number of active obstacles, and therefore, the matrix $\mathbf{N}$ has to be calculated only once. Of course, pseudoinverses $\mathbf{J}^{\#}_{o,i}$ have to be calculated for each active obstacle. For the same system and task as before, the simulation results using the approximate velocity controller AP (24) are presented in Fig. 4(b). We can see that in the case of the AP controller, the links are coming closer to the obstacle as in the case of the EX controller. However, when changing the desired critical point velocity $\dot{x}_o$, i.e., using a higher order in (18), the minimal distance can be increased.

4.5 Obstacle avoidance as a primary task

For a redundant system multiple tasks can be arranged in priority. Let us consider two tasks, T_a and T_b

$$\boldsymbol{x}_a = \boldsymbol{f}_a(\boldsymbol{q}) \qquad\qquad \boldsymbol{x}_b = \boldsymbol{f}_b(\boldsymbol{q}) \tag{26}$$

For each of the tasks, the corresponding Jacobian matrices can be defined as $\mathbf{J}_a$ and $\mathbf{J}_b$, and their corresponding null-space projections as $\mathbf{N}_a$ and $\mathbf{N}_b$. Assuming that task T_a is the primary task, Eq. (4) can be rewritten as

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}_a\dot{\boldsymbol{x}}_a + \mathbf{N}_a\mathbf{J}^{\#}_b\dot{\boldsymbol{x}}_b \tag{27}$$

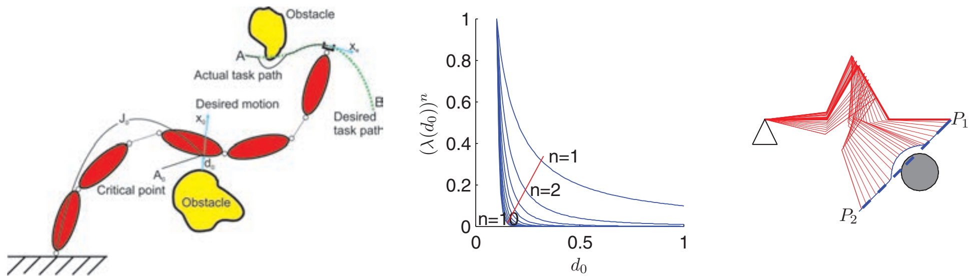


Fig. 5. Robot motion in the presence of some obstacle (left figure). Obstacle-avoidance proximity gain to the power of n (middle figure) and the planar 4DOF manipulator tracking line while avoiding the obstacle (right figure).

In many cases it would be of benefit to have the possibility to change the priority of particular subtasks. Using formulation (27) this cannot be done in a smooth way. Therefore, we propose a new definition of the velocity $\dot{\boldsymbol{q}}$. The velocity $\dot{\boldsymbol{q}}$ is now defined as

$$\dot{\boldsymbol{q}} = \mathbf{J}_a^{\#}\dot{\boldsymbol{x}}_a + \mathbf{N}_a'\mathbf{J}_b^{\#}\dot{\boldsymbol{x}}_b, \tag{28}$$

where the matrix $\mathbf{N}_a'$ is given as

$$\mathbf{N}_a' = \mathbf{I} - \lambda(\boldsymbol{x}_a)\mathbf{J}^{\#}\mathbf{J}, \tag{29}$$

where $\lambda(\boldsymbol{x}_a)$ is a scalar measure of how "active" is the primary task T_a, scaling the vector $\boldsymbol{x}_a$ to the interval $[0,1]$. When the primary task T_a is active the $\lambda(\boldsymbol{x}_a) = 1$ and the value when the task T_a is not active $\lambda(\boldsymbol{x}_a) = 0$.
The proposed algorithm allows a smooth transition in both ways, i.e., between observing the task T_a and the task T_b in null-space of the task T_a or just the unconstrained movement of the task T_b. It can be used for different robotic tasks. When applied to obstacle avoidance, task T_a is the obstacle avoidance and the end-effector motion is the task T_b. Before, we have assumed that the end-effector motion is not disturbed by an obstacle. Now, it is assumed that the motion of the end effector can be disturbed by any obstacle (see Fig. 5). If such a situation occurs, usually the task execution has to be interrupted and higher-level path planning has to be employed to recalculate the desired motion of the end effector. However, if the end-effector path tracking is not essential, we can use the proposed control (28). Consequently, no end-effector path recalculation or higher-level path planning is needed.
Let the primary task T_a be the motion in the direction $\boldsymbol{d}_0$ and the motion of the end-effector be the task T_b. Using the reduced operational space yields

$$\mathbf{J}_a = \mathbf{J}_o, \tag{30}$$

$$\mathbf{J}_b = \mathbf{J}. \tag{31}$$

Next, let the avoiding velocity $\dot{\boldsymbol{x}}_{d_0}$ be defined as

$$\dot{\boldsymbol{x}}_{d_0} = \lambda\left(\boldsymbol{d}_0\right)\boldsymbol{v}_0, \tag{32}$$

where $\boldsymbol{v}_0$ is a nominal velocity and $\lambda\left(\boldsymbol{d}_0\right)$ is defined as

$$\lambda(\boldsymbol{d}_0) = \begin{cases} \left(\dfrac{d_m}{||\boldsymbol{d}_0||}\right)^n, \ n = 1,2,3... & ||\boldsymbol{d}_0|| \geq d_m \\ 1 & ||\boldsymbol{d}_0|| < d_m \end{cases} \tag{33}$$

where $n = 1, 2, 3...$ and d_m is the critical distance to the obstacle. Then eq. (28) can be rewritten in the form

$$\dot{\boldsymbol{q}} = \mathbf{J}_o^{\dagger} \lambda \left(\boldsymbol{d}_0 \right) \boldsymbol{v}_0 + \mathbf{N}_0' \mathbf{J}^{\#} \dot{\boldsymbol{x}}_c. \tag{34}$$

Here, $\dot{\boldsymbol{x}}_c$ is the task controller for the end-effector tracking and $\mathbf{N}_0'$ is given by

$$\mathbf{N}_0' = \mathbf{I} - \lambda \left(\boldsymbol{d}_0 \right) \mathbf{J}_o^{\dagger} \mathbf{J}_o. \tag{35}$$

Formulation (34) allows unconstrained joint movement, while $\lambda(\boldsymbol{d}_0)$ is close to zero ($\lambda(\boldsymbol{d}_0) \approx 0$). Thus, the robot can track the desired task space path, while it is away from the obstacle. On the other hand, when the robot is close to the obstacle ($\lambda(\boldsymbol{d}_0) \approx 1$), the null space in (35) takes the form $\mathbf{N}_0' = \mathbf{N}_0$, and only allows movement in the null space of the primary task, i.e., the obstacle-avoidance task. In this case, we can still move the end effector, but the tracking error can increase due to the obstacle-avoiding motion.

4.6 Singular configurations

An important issue in the control of redundant manipulators is singular configurations where the associated Jacobian matrices lose the rank. Usually, only the configuration of the whole manipulator is of interest, but in obstacle avoidance we have also to consider the singularities of two manipulator substructures defined by the critical point A_o: (a) the part of the manipulator between the base and the point A_o and (b) the part between A_o and the end-effector. Although it can be assumed that the end-effector Jacobian $\mathbf{J}^{\#}$ is not singular along the desired end-effector path (otherwise the primary task can not be achieved), this is not always true for the matrix $\mathbf{J}_o\mathbf{N}$. Namely, when part (a) is in the singular configuration then $\mathbf{J}_o$ is not of full rank and when part (b) is in the singular configuration then $\mathbf{J}_o$ retains the rank but $\mathbf{J}_{d_o}\mathbf{N}$ becomes singular. Hence, when approaching either singular configuration the values of $\dot{\boldsymbol{q}}_h$ become unacceptably large. As the manipulator is supposed to move in an unstructured environment it is practically impossible to know when $\mathbf{J}_o\mathbf{N}$ will become singular. Therefore, a very important advantage of the proposed Jacobian $\mathbf{J}_{d_o}$ compared to $\mathbf{J}_o$ is that the system has significantly fewer singularities when $\mathbf{J}_{d_o}$ is used.

5. Obstacle avoidance using forces

Impedance control approaches for obstacle-avoidance were first introduced by Hogan (Hogan, 1985). These approaches make use of the additive property of impedances to supplement an impedance controller with additional disturbance forces to avoid the obstacles. The disturbance forces are generated from the artificial potential field. Lee demonstrated this approach with his reference adaptive impedance controller and gave promising results for a simulated 2-DOF robot (Lee et al., 1997).
The advantage of these approaches is that the dynamic behavior of the manipulator as it interacts with obstacles is adjustable through the gains in the obstacle-induced disturbing force. However, this approach is tightly coupled with the control scheme and requires the use of a compliance controller, which may not be desirable, depending on the task.

5.1 Obstacle avoidance without any sensors

First we want to investigate what happens if the manipulator collides with an obstacle. We are especially interested in how to control the manipulator so that it avoids the obstacle after the collision and how to minimize the contact forces. When the task itself includes contacts with the environment (e.g. assembly), the manipulator is equipped with an appropriate sensor

to measure the contact forces and the controller includes a force control loop. But when the contacts between the manipulator and an obstacle can take place anywhere on the body of the manipulator, it is questionable whether the forces arising from such contacts can be measured. Assuming that the contact forces are not measurable and no other sensors to detect the obstacles are present, the contact forces have to be considered in the controller as disturbances. Now the problem is, how to generate a motion that would move a part of the manipulator away from the obstacle. To solve this problem we propose to follow a very basic principle: *an action causes a reaction*. In other words, if a manipulator acts with a force on an obstacle then the obstacle acts with a force in the opposite direction on the manipulator and our idea is to take advantage of this reaction force to move the manipulator away from the obstacle. To make such a motion possible, the control must not force the manipulator to oppose the reaction force (e.g. by preserving the configuration). This means that the system should be compliant to these external forces. A prerequisite for such an approach is that the manipulator is *backdrivable*, meaning that any force at the manipulator is immediately felt at the motors, so the manipulator reacts rapidly, drawing back from the source of the force.
To decouple the task space and the null-space motion we propose to use the controller (7). Actually, from the obstacle-avoidance point of view, any controller for the redundant manipulators could be used provided that the controller outputs are the joint torques. However, this is not enough to decouple the task space and null-space motion. Namely, torques applied through the null-space of $\mathbf{J}^{T\#}$, i.e., $\mathbf{N}^T\boldsymbol{\tau}$, can affect the end-effector acceleration, depending on the choice of the generalized inverse. It turns out for redundant systems that only the so-called "dynamically consistent" generalized inverse $\bar{\mathbf{J}}$ defined as

$$\bar{\mathbf{J}} = \mathbf{H}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{H}^{-1}\mathbf{J}^T)^{-1} \tag{36}$$

is the unique generalized inverse that decouples the task space and the null-space motion, i.e., assures that the task space acceleration is not affected by any torques applied through the null space of $\bar{\mathbf{J}}^T$, and that the end-effector forces do not produce any accelerations in the null-space of $\mathbf{J}$ (Featherstone & Khatib, 1997).
Using the inertia weighted generalized inverse in Eq. (7) yields

$$\boldsymbol{\tau} = \mathbf{H}(\bar{\mathbf{J}}(\ddot{\boldsymbol{x}}_c - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \bar{\mathbf{N}}(\boldsymbol{\phi} - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}})) + \boldsymbol{h} + \boldsymbol{g} \tag{37}$$

Combining (3) and (37), and considering Eqs. (8) and (9) yields

$$\mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \boldsymbol{\tau}_F = \mathbf{H}(\bar{\mathbf{J}}(\ddot{\boldsymbol{x}}_d + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \bar{\mathbf{N}}(\ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}})) + \boldsymbol{h} + \boldsymbol{g} \tag{38}$$

which simplifies to

$$\bar{\mathbf{J}}(\ddot{\boldsymbol{e}} + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e}) + \bar{\mathbf{N}}(-\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}}) = -\mathbf{H}^{-1}\boldsymbol{\tau}_F \tag{39}$$

where $\boldsymbol{\tau}_F$ represents the influence of all external forces caused by the contacts with obstacles

$$\boldsymbol{\tau}_F = \sum_{i=1}^{a_o} \mathbf{J}_{o,i}^T \boldsymbol{F}_{o,i} \tag{40}$$

Premultiplying (39) with $\mathbf{J}$ yields

$$\ddot{\boldsymbol{e}} + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} = -\mathbf{J}\mathbf{H}^{-1}\boldsymbol{\tau}_F \tag{41}$$

Note that the contact forces affect the motion in the task space, which results in the tracking error of the end-effector. Of course, if the obstacle avoidance is successful the contact forces vanish and the task position error converges to zero. The proper choice of $\mathbf{K}_v$ and $\mathbf{K}_p$ assures the asymptotical stability of the homogeneous part of the system, $\ddot{\boldsymbol{e}} + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} = 0$. A detailed analysis of the influence of the external forces is given in Žlajpah & Nemec (2003).
Next we analyse the behavior in null-space. Premultiplying (39) with $\bar{\mathbf{N}}$ yields

$$-\bar{\mathbf{N}}\mathbf{H}^{-1}\boldsymbol{\tau}_F = \bar{\mathbf{N}}(-\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}}) \tag{42}$$

Rearranging Eq. (42) and using the relations $\ddot{\boldsymbol{e}}_n = \bar{\mathbf{N}}(\ddot{\boldsymbol{\varphi}} - \ddot{\boldsymbol{q}}) + \dot{\bar{\mathbf{N}}}(\dot{\boldsymbol{\varphi}} - \dot{\boldsymbol{q}})$ and $\bar{\mathbf{N}}\mathbf{H}^{-1} = \mathbf{H}^{-1}\bar{\mathbf{N}}^T$, we obtain

$$\bar{\mathbf{N}}(\ddot{\boldsymbol{e}}_n + \mathbf{K}_n\dot{\boldsymbol{e}}_n) = -\bar{\mathbf{N}}\mathbf{H}^{-1}\bar{\mathbf{N}}^T\boldsymbol{\tau}_F = -\mathbf{H}_n^{\ddagger}\boldsymbol{\tau}_F \tag{43}$$

where $\mathbf{H}_n$ is the *null-space effective inertia matrix* describing the inertial properties of the system in the null-space

$$\mathbf{H}_n = \bar{\mathbf{N}}^T\mathbf{H}\bar{\mathbf{N}}$$

and $\mathbf{H}_n^{\ddagger}$ is the generalized inverse of $\mathbf{H}_n$, defined as

$$\mathbf{H}_n^{\ddagger} = \bar{\mathbf{N}}\mathbf{H}^{-1}\bar{\mathbf{N}}^T$$

The selection of the null space dynamic properties is subjected to the subtask that the manipulator should perform. Actually, in most cases it is required that the null space velocity tracks a given desired null space velocity $\dot{\boldsymbol{\varphi}}$. For good null space velocity tracking, the gain matrix $\mathbf{K}_n$ should be high, which means that the system is stiff in the null space. On the other hand, when the manipulator collides with an obstacle, the self-motion is initiated externally by the contact force. As we have already mentioned, in this case the system should be compliant in the null-space. Therefore, the gain matrix $\mathbf{K}_n$ should be low. As both requirements for $\mathbf{K}_n$ are in conflict, a compromise has to made when selecting $\mathbf{K}_n$ or the on-line adaptation of $\mathbf{K}_n$ has to be used. The problem with the second possibility is that we have to be able to detect the current state, i.e., if the manipulator is in contact with an object. If this is possible, then it is easy to set low gains when the collision occurs and high gains when the manipulator is "free". If we cannot detect the contact and the probability of collisions is high, then it is rational to select a low $\mathbf{K}_n$, but we must be aware that too low values of $\mathbf{K}_n$ may cause instability in the null-space. The lower bounds for $\mathbf{K}_n$ can be obtained with a Lyapunov analysis (Žlajpah & Nemec, 2003).
As an illustration we use the same example as before, where the manipulator is supposed to have 4 revolute joints. The primary task is to move along a straight line from point P_1 to point P_2, and the motion is obstructed by an obstacle. The task trajectory has a trapezoid velocity profile with an acceleration of $4ms^{-2}$ and a max. velocity of $0.4ms^{-1}$. The control algorithm is given by Eq. (37) (CF). The task space controller parameters (Eq. 8) are $\mathbf{K}_p = 1000\mathbf{I}\mathrm{s}^{-2}$ and $\mathbf{K}_v = 80\mathbf{I}\mathrm{s}^{-1}$, which are tuned to ensure good tracking of the task trajectory (stiff task space behavior). The null space controller is a modified version of the controller (9)) $\boldsymbol{\phi} = -\mathbf{K}_n\dot{\boldsymbol{q}}$, i.e., the desired null-space velocity is set to zero. We have compared two sets of null-space gains: (i) Low $\mathbf{K}_n$, which ensures compliant null-space behavior, and (ii) Medium $\mathbf{K}_n$, which makes the null-space more stiff. The simulation results are presented in Figs. 6 and 7.
Although the manipulator has finished the desired task in both cases, we can see that making the null-space more compliant results in decreased contact forces. As expected, the impact force does not depend on the stiffness in the null space and cannot be decreased by increasing

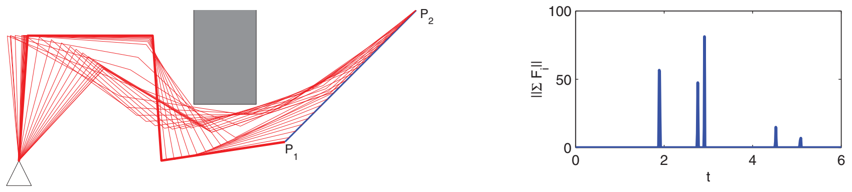


Fig. 6. Planar 4DOF manipulator tracking a line while avoiding obstacles without using any sensors to detect the obstacles (CF); compliant in null space

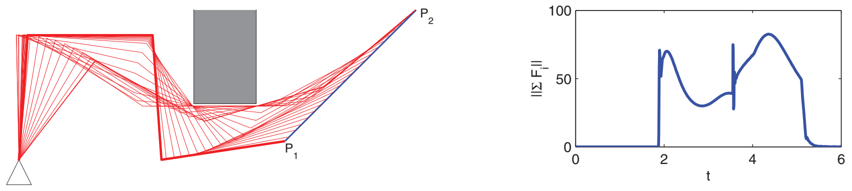


Fig. 7. Planar 4DOF manipulator tracking a line while avoiding obstacles without using any sensors to detect the obstacles (CF); stiff in null space

the compliancy in the null space. Actually, the impact forces are the main problem with this type of control. Namely, the magnitude of the impact forces cannot be controlled with the controller, the controller can decrease the contact forces after the impact. The magnitude of the impact forces depends primarily on the kinetic energy of the bodies that collide and the stiffness of the contact. Therefore, this approach to obstacle avoidance is limited to cases where the manipulator is moving slowly or the manipulator body (or obstacles) is covered with soft material, which reduces the impact forces. Note that the impact forces (the magnitude and the time of occurrence) are not the same in these examples due to the different close-loop dynamics of the system.

5.2 Obstacle avoidance with tactile sensors

As already mentioned, the obstacle-avoidance approach based on contact forces without any contact sensors cannot be applied to manipulators that are not backdrivable, like manipulators with high-ratio gears. The alternative strategy is a sensor based motion control, where a kind of tactile sensors mounted on a manipulator detect the contact with an obstacle. The main advantage of using the tactile sensor information in control is that the obstacle avoidance becomes “active”, meaning the avoiding motion is initiated by the controller. So, it can be applied to any manipulator, irrespective of the backdrivability of the manipulator.
With tactile sensors the collisions can be detected. Our approach is to identify the points on the manipulator that are in contact with obstacles and to assign to them additional virtual force components that move the points away from the obstacle (see Fig. 8). We propose that these forces are not included into the close-loop controller, but they are applied as the virtual external forces to the manipulator.
Suppose that $\boldsymbol{F}_o$ is acting at point A_o somewhere on the link i (see Fig. 8).
The static relation between the force $\boldsymbol{F}_o$ and the corresponding joint torques $\boldsymbol{\tau}_o$ is

$$\boldsymbol{\tau}_o = \mathbf{J}_o^T \boldsymbol{F}_o \tag{44}$$

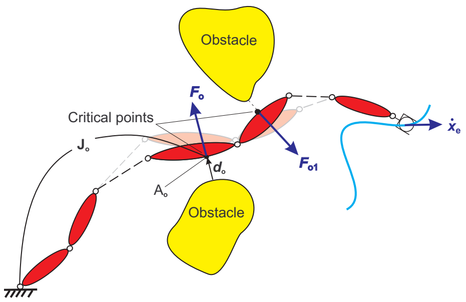


Fig. 8. Manipulator motion in the presence of some obstacles: links in contact are moved away from obstacles by additional virtual forces

where $\mathbf{J}_o$ is a Jacobian matrix associated with the point A_o. Applying $\boldsymbol{\tau}_o$ to the system yields the equation of motion in the form

$$\boldsymbol{\tau} = \mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \boldsymbol{\tau}_F - \boldsymbol{\tau}_o \tag{45}$$

which is the same as Eq. (3), except that the torques due to the virtual forces forces $\boldsymbol{\tau}_F$ are added. In general, by applying the torques $\boldsymbol{\tau}_o$ as defined in Eq. (44), the motion of the end-effector and the self-motion of the manipulator are influenced. Note that one of the goals of obstacle avoidance is to disturb the end-effector motion as little as possible. As we are adding virtual forces, it is not necessary that the applied virtual forces correspond to the “real" forces. Therefore, only torques that do not influence the end-effector motion are proposed to be added to generate the obstacle-avoidance motion. In general, the force $\boldsymbol{F}_o$ can be substituted by a force acting in the task space

$$\boldsymbol{F}_{oe} = \mathbf{J}^{T\#}\mathbf{J}_o^T\boldsymbol{F}_o \tag{46}$$

and by joint torques acting in the null-space of $\mathbf{J}^{T\#}$

$$\boldsymbol{\tau}_{oN} = \mathbf{N}^T\mathbf{J}_o^T\boldsymbol{F}_o = \mathbf{N}^T\boldsymbol{\tau}_o \tag{47}$$

Substituting $\boldsymbol{\tau}_{oN}$ for $\boldsymbol{\tau}_o$ in Eq. (45) yields

$$\boldsymbol{\tau} = \mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \boldsymbol{\tau}_F - \bar{\mathbf{N}}^T\boldsymbol{\tau}_o \tag{48}$$

The efficiency of the obstacle-avoidance algorithm depends on the selection of the desired force $\boldsymbol{F}_o$. In our approach, the virtual force $\boldsymbol{F}_o$ depends on the location of the critical point A_o, i.e., the locations of the tactile sensors that detect the contact. As the positions of the sensors are known in advance, it is easy to get the corresponding Jacobian matrix $\mathbf{J}_o$ for each sensor. Additionally, each sensor also has a predefined avoiding direction $\boldsymbol{n}_o$. We define the virtual forces $\boldsymbol{F}_o$ as

$$\boldsymbol{F}_o = \alpha_t f_o \boldsymbol{n}_o \tag{49}$$

where α_t is the obstacle-avoidance gain. We propose that α_t depends on the duration of the contact and that the achievedvalue is preserved for some time after the contact between the manipulator and the obstacle is lost (see Fig. 9)

$$\alpha_t = \begin{cases} 0 & \text{for} \quad t < t_s \\ e^{T_i(t-t_s)} & \text{for} \quad t_s \leq t < t_e \\ e^{T_i(t_e-t_s)}e^{-T_d(t-t_e)} & \text{for} \quad t_e \leq t \end{cases} \tag{50}$$

where T_i and T_d are the constants for the increase of the gain value and the delayed action, and t_s and t_e represent the time when the contact occurs and the time when the contact is lost, respectively. With the appropriate selection of T_i and T_d a robust behavior can be obtained.

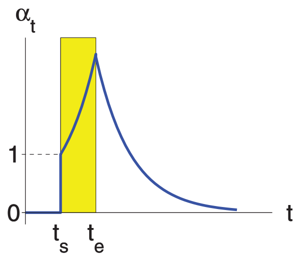


Fig. 9. The obstacle-avoidance gain α_t versus the duration of the contact

The next issue is how the obstacle avoidance is performed when there are more simultaneously active obstacles in the neighborhood of the manipulator. Using the proposed obstacle-avoidance formulation the problem of many simultaneously active obstacles can be solved very efficiently. The additivity of the torques makes it possible to avoid multiple obstacles by using the sum of the torques due to the relevant virtual forces,

$$\boldsymbol{\tau}_o = \sum_{i=1}^{a_o} \mathbf{J}_{o,i}^T \boldsymbol{F}_{o,i} \tag{51}$$

and considering this in Eq. (47) yields

$$\boldsymbol{\tau}_{o,N} = \bar{\mathbf{N}}^T \sum_{i=1}^{a_o} \mathbf{J}_{o,i}^T \boldsymbol{F}_{o,i} \tag{52}$$

where a_o is the number of active obstacles. It is clear that when more than one obstacle is active it is necessary to calculate only the transpose of the Jacobian $\mathbf{J}_{o,i}^T$ for each critical point and not the generalized inverse $\bar{\mathbf{J}}_{o,i}$, as is the case with velocity-based strategies. The redundancy of the manipulator is considered in the term $\bar{\mathbf{N}}^T$, which does not depend on the location of particular critical points $A_{o,i}$. Hence, there is no limitation on a_o regarding the degree-of-redundancy. In the case when a_o is greater than the degree-of-redundancy, the manipulator is pushed into a configuration where the virtual forces compensate each other, i.e., $\boldsymbol{\tau}_o = 0$. Actually, such situations can occur even when a_o is less than the degree-of-redundancy, e.g. when one link is under the influence of more than one obstacle. The force formulation has its advantages computationally, e.g. in Eq. (52) the term $\bar{\mathbf{N}}^T$ is calculated only once.
For the close-loop control the controller (37) is used again. Augmenting (38) with virtual forces the behavior of the system with contact sensors is described by

$$\mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \boldsymbol{\tau}_F - \bar{\mathbf{N}}^T\boldsymbol{\tau}_o = \mathbf{H}(\bar{\mathbf{J}}(\ddot{\boldsymbol{x}}_d + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \bar{\mathbf{N}}(\ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}})) + \boldsymbol{h} + \boldsymbol{g} \tag{53}$$

Actually, the term $\bar{\mathbf{N}}^T\boldsymbol{\tau}_o$ is also part of the controller and should be on the right-hand side of Eq. (53), but we put it on the left-hand side to emphasize its role. Eq. (53) simplifies to

$$\bar{\mathbf{J}}(\ddot{\boldsymbol{e}} + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e}) + \bar{\mathbf{N}}(-\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}}) = -\mathbf{H}^{-1}(\bar{\mathbf{N}}^T\boldsymbol{\tau}_o + \boldsymbol{\tau}_F) \tag{54}$$

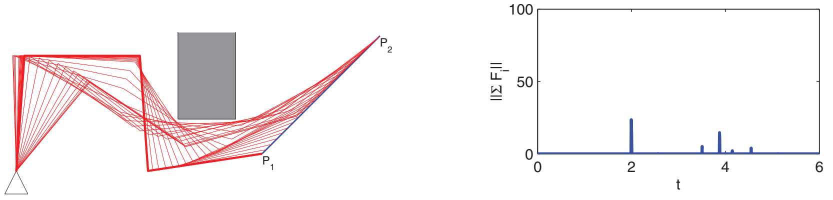


Fig. 10. Planar 4DOF manipulator tracking a line while avoiding obstacles using tactile sensors and virtual forces (TF controller)

Premultiplying (54) with $\mathbf{J}$ yields

$$\ddot{e} + \mathbf{K}_v \dot{e} + \mathbf{K}_p e = -\mathbf{J}\mathbf{H}^{-1}\boldsymbol{\tau}_F \tag{55}$$

which is the same as Eq. (41), and premultiplying (54) with $\bar{\mathbf{N}}$ yields after simplifications

$$\bar{\mathbf{N}}\ddot{e}_n + \bar{\mathbf{N}}\mathbf{K}_n \dot{e}_n = -\mathbf{H}_n^{\ddagger}(\boldsymbol{\tau}_o + \boldsymbol{\tau}_F) \tag{56}$$

From Eqs. (55) and (56) we can see that the task space and the null-space motion are influenced by the contact forces. However, these forces decrease after the impact because of the avoiding motion.
Next, we present the simulation results of the same task as before for the system where the tactile sensors (TF) were used to detect the obstacle. The close loop controller was the same as for the CF case (ii) approach, which assures stiff null space behavior. The avoidance motion was generated using Eqs. (49) – (50) with the parameters $f_o = 100N$, $T_i = 6s$ and $T_d = 8s$. The results, see Fig. 10, show that with tactile sensors we can decrease the contact forces after the contact, but the impact force cannot be decreased. Unfortunately, the magnitude of the impact forces is not controllable and hence this approach also requires low velocities and a soft contact.

5.3 Obstacle avoidance with proximity sensors

When proximity sensors are used to detect obstacles, the collisions between the manipulator and obstacles can be avoided. Also here, our approach is to identify the points on the manipulator that are near obstacles and to assign to them force components that move the points away from the obstacle (see Fig. 8). The strategy is similar to those given in (Brock et al., 2002), although it was developed independently.
Suppose that $\boldsymbol{F}_o$ is acting at point A_o somewhere on the link i (see Fig. 8). Applying $\boldsymbol{\tau}_o$ (see Eq. 44) to the system yields the equation of motion in the form

$$\boldsymbol{\tau} = \mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \boldsymbol{\tau}_o \tag{57}$$

which is the same as Eq. (3), except that the real external forces $\boldsymbol{\tau}_F$ are replaced by the virtual forces $\boldsymbol{\tau}_o$. As we are using virtual forces, only torques that do not influence the end-effector motion are considered

$$\boldsymbol{\tau}_{oN} = \mathbf{N}^T \mathbf{J}_o^T \boldsymbol{F}_o = \mathbf{N}^T \boldsymbol{\tau}_o \tag{58}$$

Substituting $\boldsymbol{\tau}_{oN}$ for $\boldsymbol{\tau}_o$ in Eq. (57) yields

$$\boldsymbol{\tau} = \mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \bar{\mathbf{N}}^T \boldsymbol{\tau}_o \tag{59}$$

The main difference compared to the system with tactile sensors is how the virtual forces $\boldsymbol{F}_o$ are generated. Now, the virtual force $\boldsymbol{F}_o$ depends on the location of the critical point and the closest point on the obstacle. Clearly, the location of all the objects in the workspace of the manipulator has to be known. This information can be provided by a higher control level that has access to sensory data. As sensor systems are not our concern, we will assume that the sensor can detect obstacles in the workspace of the manipulator and that it outputs the direction and the distance to the closest point obstacle, and the position of the critical point on the manipulator.
Let A_o be the critical point and $\boldsymbol{d}$ the vector connecting the closest point on the obstacle and A_o (see Fig. 8). To avoid a possible collision a virtual force $\boldsymbol{F}_o$ is assigned to A_o, defined as

$$\boldsymbol{F}_o = \alpha_f f_o \boldsymbol{n}_o \tag{60}$$

where f_o is a scalar gain representing the nominal force, $\boldsymbol{n}_o$ is the unit vector in the direction of $\boldsymbol{d}$, and α_f is the obstacle-avoidance gain. The gain α_f should depend on the distance to the obstacle, as shown in Fig. 11.

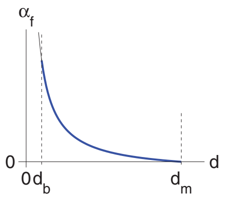


Fig. 11. The obstacle-avoidance gain α_f versus the distance to the obstacle

There are two distances characterizing the changes in the value of the gain: the critical distance d_m and the abort distance d_b. If the distance between the manipulator and the object is greater than d_m then the motion of the manipulator is not perturbed. When the distance is decreasing, the force should increase smoothly (to assure smooth transitions it is important that the magnitude of $\boldsymbol{F}_o$ at d_m is zero). However, if the manipulator is too close to the obstacle (less than d_b) the main task should be, for safety reasons, aborted so that the manipulator can avoid the obstacles (such a situation can occur, especially if moving obstacles are present in the workspace). The distance d_b is subjected to the dynamic properties of the manipulator and can also be a function of the relative velocity between the critical point on the manipulator and the obstacle. The gain α_f can be chosen arbitrarily as long as it has the prescribed form. We propose using the following function

$$\alpha_f = \begin{cases} \left(\frac{d_m}{\|\boldsymbol{d}_o\|}\right)^2 - 1 & \text{for} \quad \|\boldsymbol{d}_o\| < d_m \\ 0 & \text{for} \quad \|\boldsymbol{d}_o\| \geq d_m \end{cases} \tag{61}$$

Special attention has to be given to the selection of the nominal force f_o. Large values of f_o can cause unnecessarily high accelerations and velocities. Consequently, the manipulator could move far from the obstacle. Such a motion may cause problems if there are more obstacles in the neighborhood of the manipulator. Namely, the manipulator may bounce between the obstacles. On the other hand, too small values of f_o would not move the critical point sufficiently away from the obstacle.

When there are more simultaneously active obstacles in the neighborhood of the manipulator, the sum of the torques due to the relevant virtual forces is used,

$$\boldsymbol{\tau}_o = \sum_{i=1}^{a_o} \mathbf{J}_{o,i}^T \boldsymbol{F}_{o,i} \tag{62}$$

and considering this in Eq. (58) yields

$$\boldsymbol{\tau}_{o,N} = \bar{\mathbf{N}}^T \sum_{i=1}^{a_o} \mathbf{J}_{o,i}^T \boldsymbol{F}_{o,i} \tag{63}$$

where a_o is the number of active obstacles.
To decouple the task space and the null-space motion we propose as before the controller (37). Combining (48) and (37), and considering Eqs. (8) and (9) yields

$$\mathbf{H}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{g} - \bar{\mathbf{N}}^T\boldsymbol{\tau}_o = \mathbf{H}(\bar{\mathbf{J}}(\ddot{\boldsymbol{x}}_d + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \bar{\mathbf{N}}(\ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}})) + \boldsymbol{h} + \boldsymbol{g} \tag{64}$$

As before, the term $\bar{\mathbf{N}}^T\boldsymbol{\tau}_o$ is also part of the controller and should be on the right-hand side of Eq. (64), but we put it on the left-hand side to emphasize its role.
One of the reasons for using the above control law is that the null-space velocity controller (9) can be used for other lower-priority tasks. Hence, to optimize p we can select $\dot{\boldsymbol{\varphi}}$ in (9) as

$$\dot{\boldsymbol{\varphi}} = \mathbf{H}^{-1}k_p\nabla p \tag{65}$$

Note that when the weighted generalized inverse of $\mathbf{J}$ is used, the desired null space velocity has to be multiplied by the inverse of the weighting matrix (in our case $\mathbf{H}^{-1}$) to assure the convergence of the optimization of p (Nemec, 1997).
Next we analyse the behavior of the close-loop system. Premultiplying Eq. (64) by $\mathbf{H}^{-1}$ yields

$$\ddot{\boldsymbol{q}} - \mathbf{H}^{-1}\bar{\mathbf{N}}^T\boldsymbol{\tau}_o = \bar{\mathbf{J}}(\ddot{\boldsymbol{x}}_d + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \bar{\mathbf{N}}(\ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}})$$

and by using (1) the above equation can be rewritten in the form

$$\bar{\mathbf{J}}(\ddot{\boldsymbol{e}} + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e}) + \bar{\mathbf{N}}(-\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}}) = -\mathbf{H}^{-1}\bar{\mathbf{N}}^T\boldsymbol{\tau}_o \tag{66}$$

The dynamics of the system in the task space and in the null-space can be obtained by premultiplying Eq. (66) by $\mathbf{J}$ and $\bar{\mathbf{N}}$, respectively. Premultiplying (66) with $\mathbf{J}$ yields

$$\ddot{\boldsymbol{e}} + \mathbf{K}_v\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} = 0 \tag{67}$$

since $\mathbf{J}\bar{\mathbf{J}} = \mathbf{I}$, $\bar{\mathbf{N}}\mathbf{H}^{-1} = \mathbf{H}^{-1}\bar{\mathbf{N}}^T$, and $\mathbf{J}\bar{\mathbf{N}} = \mathbf{0}$. The proper choice of $\mathbf{K}_v$ and $\mathbf{K}_p$ assures the asymptotical stability of the system. Furthermore, it can be seen that the virtual forces do not affect the motion in the task space. Next, premultiplying (66) by $\bar{\mathbf{N}}$ yields

$$\bar{\mathbf{N}}(-\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{\varphi}} + \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{\varphi}} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\bar{\mathbf{N}}}\dot{\boldsymbol{q}}) = -\bar{\mathbf{N}}\mathbf{H}^{-1}\bar{\mathbf{N}}^T\boldsymbol{\tau}_o \tag{68}$$

because $\bar{\mathbf{N}}\bar{\mathbf{J}} = \mathbf{0}$. Rearranging Eq. (68) and using the relation $\ddot{\boldsymbol{e}}_n = \bar{\mathbf{N}}(\ddot{\boldsymbol{\varphi}} - \ddot{\boldsymbol{q}}) + \dot{\bar{\mathbf{N}}}(\dot{\boldsymbol{\varphi}} - \dot{\boldsymbol{q}})$ we obtain

$$\bar{\mathbf{N}}(\ddot{\boldsymbol{\varphi}} - \ddot{\boldsymbol{q}} + \dot{\bar{\mathbf{N}}}(\dot{\boldsymbol{\varphi}} - \dot{\boldsymbol{q}}) + \mathbf{K}_n\dot{\boldsymbol{e}}_n) = -\bar{\mathbf{N}}\mathbf{H}^{-1}\bar{\mathbf{N}}^T\boldsymbol{\tau}_o = -\mathbf{H}_n^{\ddagger}\boldsymbol{\tau}_o \tag{69}$$

From Eq. (69) we can see that the obstacle-avoidance motion is not controlled directly by $\boldsymbol{F}_o$. The force $\boldsymbol{F}_o$ initiates the motion, but the resulting motion depends mainly on the null-space controller (9). Although the null-space controller has in this case the same structure as in the case of the contact forces approach, the gains $\mathbf{K}_n$ can be selected to meet the requirements of the subtask the manipulator should perform, i.e., in most cases to track the desired null space velocity. Hence, the gain matrix $\mathbf{K}_n$ should be high. Consequently, to perform satisfactory obstacle avoidance the nominal force f_o must be higher to predominate over the velocity controller when necessary.
In the following the simulation results for the simple task are shown. The task space controller parameters are the same as in the previous example. To avoid the obstacles the proximity sensor distance was selected as $d_m = 0.2m$. The virtual forces were calculated using Eq. (40) with the parameter $f_o = 800N$. The simulation results, see Fig. 12, clearly show that the obstacle is avoided without a deviation of the end-effector from the assigned task.

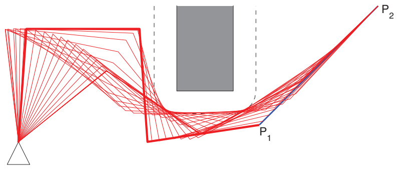


Fig. 12. Planar 4DOF manipulator tracking a line while avoiding obstacles using a virtual forces approach (VF)

6. Experimental results

The proposed algorithms were tested on the laboratory manipulator (see Fig. 13 and two KUKA LWR robots (see Fig. 14). The laboratory manipulator was specially developed for testing the different control algorithms. To be able to test the algorithms for the redundant systems the manipulator has four revolute DOF acting in a plane. The link lengths of the manipulator are $l = (0.184, 0.184, 0.184, 0.203)m$ and the link masses are $m = 0.83, 0.44, 0.18, 0.045)kg$. The manipulator is a part of the integrated environment for the design of the control algorithms and the testing of these algorithms on a real system (Žlajpah, 2001).

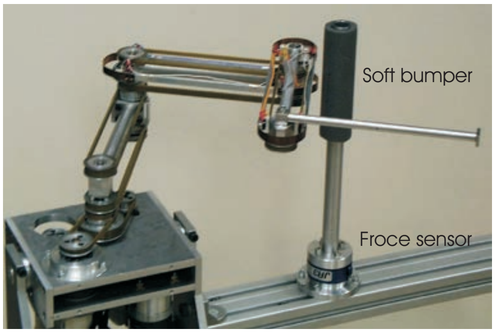


Fig. 13. Experimental manipulator with external force sensor

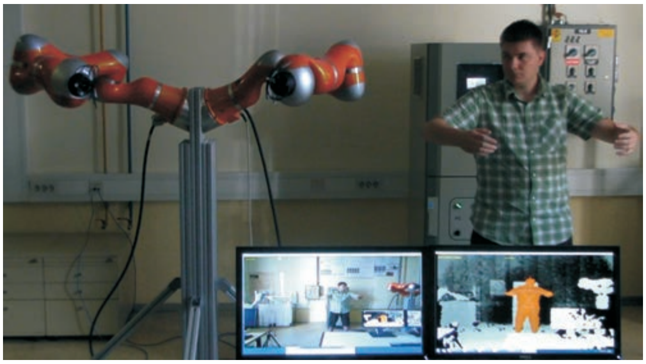


Fig. 14. Experimental setup with two KUKA LWR robots for bimanual movement imitation.

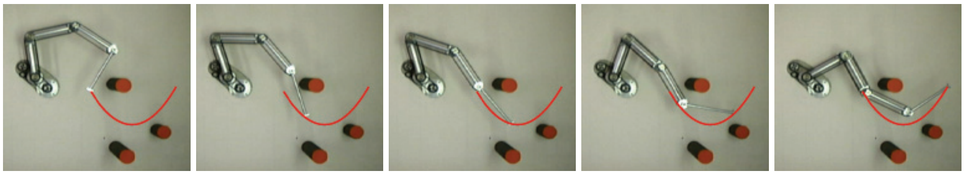


Fig. 15. Planar 4DOF manipulator tracking a path in an unstructured environment with three obstacles using velocity-based control (sampled every 1s)

6.1 Velocity controller

The task in these experiments was tracking the path $\boldsymbol{x}_d = [0.4 - 0.2\sin(2\pi t/8), -0.1 + 0.1\sin(2\pi t/4)]^T$ and the motion of the manipulator was obstructed by three obstacles (see Fig. 15). In the current implementation the vision system is using a simple USB WebCam that can recognize the scene and output the position of all the obstacles in less that 0.04s. To avoid the obstacles the proximity sensor distance was selected as $d_m = 0.08m$. The rate of the velocity controller was 200Hz (the necessary joint velocities for the avoiding motion are calculated in less than 0.5ms). The experimental results are given in Fig. 15. As we can see, the obstacles were successfully avoided.

6.2 Obstacle avoidance as a primary task

We applied our algorithm to two Kuka LWR robots as shown in Fig. 14. Our algorithm is used as a low-level control to prevent self-collision, i.e., a collision between the robots themselves. As mentioned previously, the desired movement of the robot is a secondary task. The task of collision avoidance is only observed if we approach a pre-defined threshold. While far from the threshold the algorithm allows direct control of the separate joints ($\dot{\boldsymbol{q}}_n$). If we approach the threshold, the task of collision avoidance smoothly takes over and only allows joint control in the null space projection of this task. Note that $\dot{\boldsymbol{q}}_n$ is in joint space.
The task for both arms in this experiment was to follow the human demonstrator in real-time. The human motion is captured using the Microsoft Kinect sensor. Microsoft Kinect is based on arange camera developed by PrimeSense, which interprets 3D scene information from a continuously-projected infrared structured light. By processing the depth image, the PrimeSense application programming interface (API) enables tracking of the user's movement in real time. Imitating the motion of the user's arm requires some basic understanding of

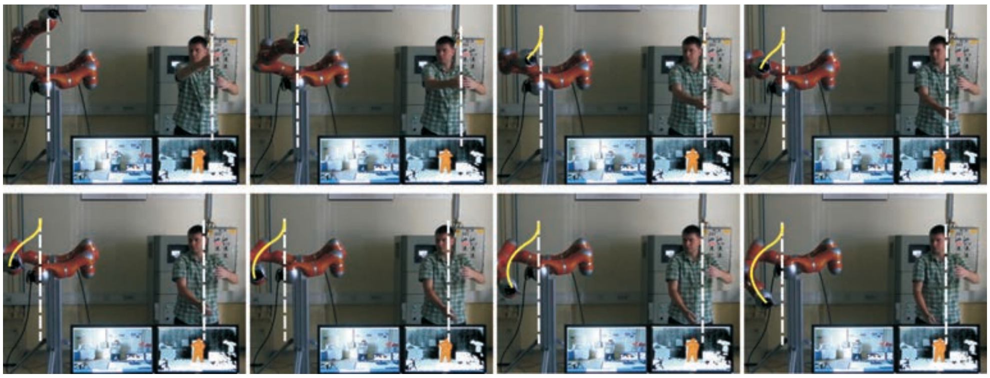


Fig. 16. A sequence of still photographs shows the movement of two Kuka LWR robots, while they successfully avoids each other. The desired movement for the robots is imitated in real time using the Microsoft Kinect sensor for the tracking.

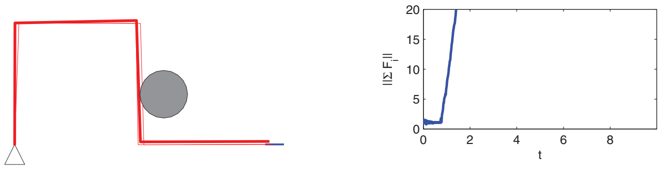
human physiology. The posture of each arm may be described by four angles - three angles in the shoulder joint and one in the elbow. The shoulder joint enables the following motion (Hayes et al., 2001): arm flexion, arm abduction and external rotation. These angles are calculated from the data obtained with Microsoft Kinect.
A sequence for successful self collision-avoidance is shown in Fig. 16. Here, we can see that the robot angles are similar when the humans hands are away from the threshold, i.e., the robots are not close together. On the other hand, when close together, the robots properly adapt their motion to prevent a collision.

6.3 Contact forces

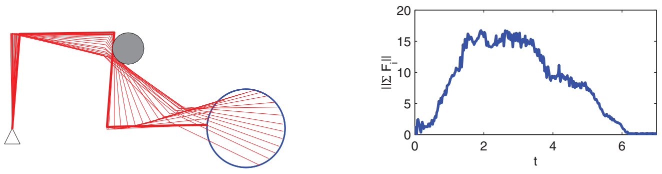
First we tested the behavior of the manipulator when no sensors were used to detect the obstacles. The desired task was to track the circular path and the motion of the manipulator was obstructed by an obstacle (the rod; see Fig. 13). To be able to monitor the contact forces the rod was mounted on a force sensor. Note that the force information measured by this sensor was not used in the close loop controller.
The controller was based on the algorithm (7) with the task controller (8) and null space controller (9) and we compared three sets of null space controller parameters. In the first case (a) the controller assured very stiff null space behavior, in the second (b) the controller assured medium stiffness in null space, and in the last case (c) the manipulator was compliant in the null space. To prevent high impact forces, the bumper was covered with soft material and the manipulator joint velocities were low.
The experimental results are shown in Fig. 17. First we can see that in case (a) the manipulator pushes the bumper more and more and finally, the task has to be aborted due to the very large contact forces. Next, comparing the responses one can see that although the motion is almost equal in both cases, the forces are lower in case (b) (compliant in null-space). Summarizing, the experimental results proved that the contact forces can be decreased by increasing the null space compliance.

6.4 Tactile sensors

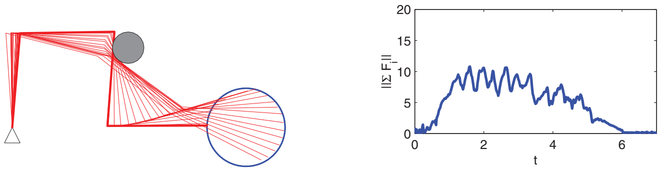
Next we tested the efficiency of the tactile sensors. In our experiments we used simple bumpers on each link as tactile sensors. The sensor can detect an object when it touches the



(a) stiff null-space behavior



(b) compliant null-space behavior



(c) more compliant null-space behavior

Fig. 17. Planar 4DOF manipulator tracking a circle while avoiding obstacles without any sensor for obstacle detection

object. In our case each switch needs a force of $2N$ to trigger it. This means that the sensor can detect an object if the contact force is greater then $\sim 2-4N$, depending on the particular contact position on the bar. The main drawback of this type of sensor is that it can only detect the link and the side of the link where the contact occurs, and not the exact position of the contact.

The desired task in these experiments was tracking the linear path and as before, the motion of the manipulator was obstructed by the rod. The virtual forces were calculated using Eqs. (49) – (50). As our sensor can detect only the side of the link where the contact occurs and not the exact position of the contact, we used the middle of the link as the approximation for the contact point and the avoiding direction was perpendicular to the link.

The experimental results are shown in Fig. 18. The figures show the contact forces norm $||\boldsymbol{F}_{\text{ext}}||$ and the configurations of the manipulator. The results clearly show that the manipulator avoids the obstacle after the collision. The behavior of the manipulator after the contact with the obstacle depends mainly on the particular nominal virtual force f_o and the time constants T_i and T_d. Tuning these parameters results in different behaviors of the system. With experiments we have found that it is possible to tune these parameters so that the manipulator slides along the obstacle with minimal impact forces and chattering.

6.5 Virtual forces

Finally, we tested the VF strategy. The desired task was tracking the linear path and the motion of the manipulator was obstructed by a different number of obstacles. To detect the obstacles a vision system was used. In the current implementation the vision system used a simple USB WebCam, which can recognize the scene and output the position of all obstacles in less than

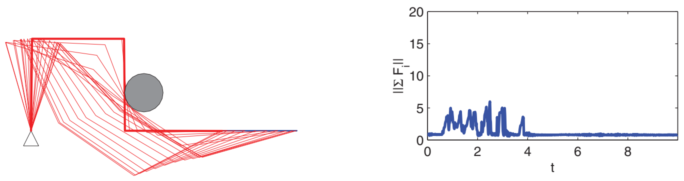
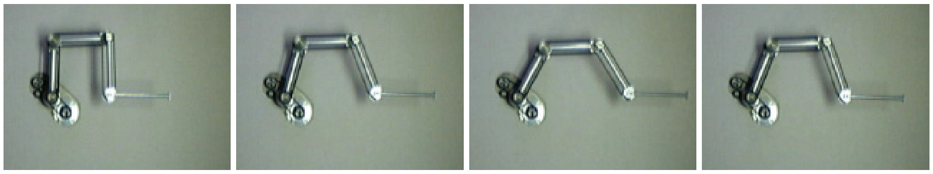
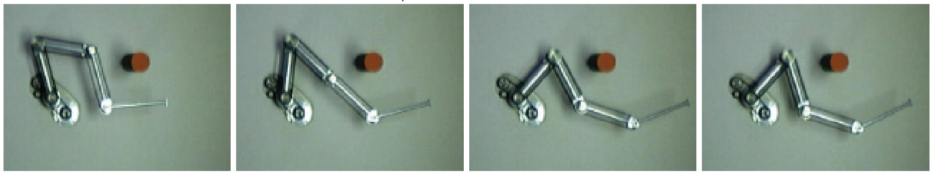


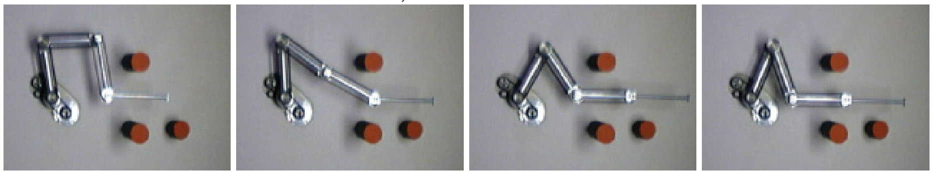
Fig. 18. Planar 4DOF manipulator tracking a circle while and avoiding obstacles using bumper and a virtual forces controller



a) no obstacles



b) one obstacle



c) three obstacles

Fig. 19. Planar 4DOF manipulator tracking a path in an unstructured environment using a vision system to detect obstacles

0.04s. To avoid the obstacles the proximity sensor distance was selected as $d_m = 0.18m$ and the nominal virtual force as $f_o = 10N$. The rate of the torque controller was 400Hz (the necessary virtual forces for the avoiding motion were calculated in less than 0.5ms). The experimental results are given in Fig. 19. We can clearly see the difference in the motion when no obstacles are present and when one or more obstacles are present.

7. Conclusion

The presented approaches for on-line obstacle avoidance for redundant manipulators are a) based on redundancy resolution at the velocity level or b) considering also the dynamics of the manipulator. The primary task is determined by the end-effector trajectories and for the obstacle avoidance the internal motion of the manipulator is used. The goal is to assign each point on the body of the manipulator, which is close to the obstacle, a motion component in a direction that is away from the obstacle.

In the case of kinematic control (a) this is a velocity component. We have shown that it reasonable to define the avoiding motion in a one-dimensional operational space. In this way some singularity problems can be avoided when not enough "redundancy" is available locally. Additionally, the calculation of the pseudoinverse of the Jacobian matrix $\mathbf{J}_o$ is simpler as it includes scalar division instead of a matrix inversion. Using an approximate calculation of the avoiding velocities has its advantages computationally and it makes it easier to consider more obstacles simultaneously.

The second group of control algorithms (b) used in case b) is based on real or virtual forces. We compare three approaches regarding the sensors used to detect the obstacles: proximity sensors or vision, tactile sensors and no sensors. When proximity sensors are used we propose virtual forces strategy, where a virtual force component in a direction away from the obstacle is assigned to each point on the body of the manipulator, which is close to an obstacle. The algorithm based on the virtual forces avoids the problem of singular configurations and can be also easily applied when many obstacles are present. Additionally, the proposed control scheme enables us to use the null-space velocity controller for additional subtasks like the optimization of a performance criterion. Next, we have shown that under certain conditions obstacle avoidance can also be done without any information about the position and the size of the obstacles. This can be achieved by using a strategy that utilizes the self-motion caused by the contact forces to avoid an obstacle after the collision. Of course, an obstacle can be avoided only after a contact. The necessary prerequisite for this strategy to be effective is that the manipulator is backdrivable. As an alternative for the stiff systems we propose the use of tactile sensors. Here, a tactile sensor detects an obstacle and the controller generates the avoiding motion. The drawback of the last two control approaches is that they do not prevent the collision with the obstacle. Hence, they can only be applied if the collision occurs at a low speed so that the impact forces are not too high.

For the tasks where end-effector tracking is not essential for performing a given task we proposed a modified prioritized task control at the velocity level. The proposed approach enables a soft continuous transition between two different tasks. The obstacle-avoidance task only takes place when the desired movement approaches a given threshold, and then smoothly switches the priority of the tasks. The usefulness of this approach was shown on a two Kuka LWR robot to prevent a collision between them.

The computational efficiency of the proposed algorithms allows real-time application in an unstructured or time-varying environment. The simulations of highly redundant planar manipulators and the experiments on a four-link planar manipulator confirm that the proposed control algorithms assure an effective obstacle avoidance in an unstructured environment.

8. References

Brock, O., Khatib, O. & Viji, S. (2002). Task-Consistent Obstacle Avoidance of Motion Behavior for Mobile Manipulation, *Proc. IEEE Conf. Robotics and Automation*, Washington D.C, pp. 388 – 393.

Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators, *IEEE Trans. on Robotics and Automation* 13(3): 398 – 410.

Colbaugh, R., Seraji, H. & Glass, K. (1989). Obstacle Avoidance for Redundant Robots Using Configuration Control, *J. of Robotic Systems* 6(6): 721 – 744.

Egeland, O. (1987). Task-space tracking with redundant manipulators, *Robotics and Automation, IEEE Journal of* 3(5): 471 –475.

Featherstone, R. & Khatib, O. (1997). Load Independance of the Dynamically Consistent Inverse of the Jacobian Matrix, *Int. J. of Robotic Research* 16(2): 168 – 170.

Glass, K., Colbaugh, R., Lim, D. & Seraji, H. (1995). Real-Time Collision Avoidance for Redundant Manipulators, *IEEE Trans. on Robotics and Automation* 11(3): 448 – 457.

Guo, Z. & Hsia, T. (1993). Joint Trajectory Generation for Redundant Robots in an Environment with Obstacles, *J. of Robotic Systems* 10(2): 119 – 215.

Hayes, K., Walton, J. R., Szomor, Z. R. & Murrell, G. A. (2001). Reliability of five methods for assessing shoulder range of motion., *The Australian journal of physiotherapy* 47(4): 289–294.

Hogan, N. (1985). Impedance Control: An Approach to Manipulation: Part 1: Theory, Part 2: Implementation, Part 3: Applications, *Trans. of ASME J. of Dynamic Systems, Measurement, and Control* 107: 1 – 24.

Khatib, O. (1986). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *Int. J. of Robotic Research* 5: 90 – 98.

Khatib, O. (1987). A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation, *IEEE Trans. on Robotics and Automation* 3(1): 43 – 53.

Kim, J. & Khosla, P. (1992). Real-Time Obstacle Avoidance Using Harmonic Potential Functions, *IEEE Trans. on Robotics and Automation* 8(3): 338 – 349.

Žlajpah, L. & Nemec, B. (2003). Force strategies for on-line obstacle avoidance for redundant manipulators, *Robotica* 21(6): 633 – 644.

Lee, S., Yi, S.-Y., Park, J.-O. & Lee, C.-W. (1997). Reference adaptive impedance control and its application to obstacle avoidance trajectory planning, *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, Vol. 2, pp. 1158 – 1162 vol.2.

Lozano-Perez, T. (1983). Spatial Planning: A Configuration space approach, *IEEE Trans. on Computers* C-32(2): 102 – 120.

Maciejewski, A. & Klein, C. (1985). Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments, *Int. J. of Robotic Research* 4(3): 109 – 117.

Mansard, N., Khatib, O. & Kheddar, A. (2009). A unified approach to integrate unilateral constraints in the stack of tasks, *Robotics, IEEE Transactions on* 25(3): 670 –685.

McLean, A. & Cameron, S. (1996). The Virtual Springs Method: Path and Collision Avoidance for Redundant Manipulators, *Int. J. of Robotic Research* 15(4): 300 – 319.

Nakamura, Y., Hanafusa, H. & Yoshikawa, T. (1987). Task-Priority Based Redundancy Control of Robot Manipulators, *Int. J. of Robotic Research* 6(2): 3 – 15.

Nemec, B. (1997). Force Control of Redundant Robots, *in* M. Guglielmi (ed.), *Preprits of 5th IFAC Symp. on Robot Control, SYROCO'97*, Nantes, pp. 215 – 220.

Newman, W. S. (1989). Automatic Obstacle Avoidance at High Speeds via Reflex Control, *Proc. IEEE Conf. Robotics and Automation*, Scottsdale, pp. 1104 – 1109.

O'Neil, K. (2002). Divergence of linear acceleration-based redundancy resolution schemes, *Robotics and Automation, IEEE Transactions on* 18(4): 625 – 631.

Park, J., Chung, W. & Youm, Y. (1996). Design of Compliant Motion Controllers for Kinematically Redundant Manipulators, *Proc. IEEE Conf. Robotics and Automation*, pp. 3538 – 3544.

Raibert, M. H. & Craig, J. J. (1981). Hybrid Position/Force Control of Manipulators, *Trans. of ASME J. of Dynamic Systems, Measurement, and Control* 102: 126 – 133.

Sciavicco, L. & Siciliano, B. (2005). *Modelling and Control of Robot Manipulators (Advanced Textbooks in Control and Signal Processing)*, Advanced textbooks in control and signal processing, 2nd edn, Springer.

Sentis, L., Park, J. & Khatib, O. (2010). Compliant control of multicontact and center-of-mass behaviors in humanoid robots, *Robotics, IEEE Transactions on* 26(3): 483 –501.

Seraji, H. & Bon, B. (1999). Real-Time Collision Avoidance for Position-Controlled Manipulators, *IEEE Trans. on Robotics and Automation* 15(4): 670 – 677.

Sugiura, H., Gienger, M., Janssen, H. & Goerick, C. (2007). Real-time collision avoidance with whole body motion control for humanoid robots, *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 2053 –2058.

Volpe, R. & Khosla, P. (1990). Manipulator Control with Superquadratic Artificial Potential FUnctions: Theory and Experiments, *IEEE Trans. on Systems, Man, Cybernetics* 20(6).

Žlajpah, L. (2001). Integrated environment for modelling, simulation and control design for robotic manipulators, *Journal of Intelligent and Robotic Systems* 32(2): 219 – 234.

Woernle, C. (1993). Nonlinear Control of Constrained Redundant Manipulators, *in* J. A. et al. (ed.), *Computational Kinematics*, Kluwer Academic Publishers, pp. 119 – 128.

Xie, H., Patel, R., Kalaycioglu, S. & Asmer, H. (1998). Real-Time Collision Avoidance for a Redundant Manipulator in an Unstructured Environment, *Proc. Intl. Conf. On Intelligent Robots and Systems IROS'98*, Victoria, Canada, pp. 1925 – 1930.

Yoshikawa, T. (1987). Dynamic Hybrid Position / Force Control of Robot Manipulators Description of Hand Constraints and Calculation of Joint Driving, *IEEE Trans. on Robotics and Automation* 3(5): 386 – 392.