Obstacle Avoidance with Industrial Robots

Chapter · March 2015

DOI: 10.1007/978-3-319-14705-5_5

4 authors:

Tadej Petric
Jožef Stefan Institute
79 PUBLICATIONS **712** CITATIONS
SEE PROFILE

Andrej Gams
Jožef Stefan Institute
89 PUBLICATIONS **1,125** CITATIONS
SEE PROFILE

Nejc Likar
Jožef Stefan Institute
11 PUBLICATIONS **70** CITATIONS
SEE PROFILE

Leon Zlajpah
Jožef Stefan Institute
111 PUBLICATIONS **908** CITATIONS
SEE PROFILE

Obstacle Avoidance with Industrial Robots

T. Petrič, A. Gams, N. Likar and L. Žlajpah

Abstract One of the important features that a robot must possess when working in an unstructured environment is the ability to deal with objects. Such objects can be a part of the task, e.g., in assembly operations, or they can represent an obstacle. In the case when contact with the objects is not desired, the main issue is how to perform the desired task without any risk of collisions with the objects in the workspace. A general strategy for obstacle avoidance is to reconfigure the robot so that it is not in the contact with the obstacle. However, a reconfiguration without changing the task motion is only feasible if the robot has sufficient redundant degrees of freedom (DOFs). In this chapter we present different approaches to the control methods of redundant robot manipulators performing multiple tasks with obstacle avoidance. The pros and cons of the presented methods and the differences between them are also discussed. The performance of the methods is also demonstrated by simulation and on real robots.

Keywords Redundant robots · Obstacle avoidance · Kinematic control · Prioritized task control · Dynamic movement primitives

T. Petrič · A. Gams · N. Likar · L. Žlajpah (✉)
Department for Automation, Biocybernetics and Robotics, Jožef Stefan Institute,
Jamova Cesta 39, Ljubljana, Slovenia
e-mail: leon.zlajpah@ijs.si

T. Petrič
e-mail: tadej.petric@ijs.si

A. Gams
e-mail: andrej.gams@ijs.si

N. Likar
e-mail: nejc.likar@ijs.si

G. Carbone and F. Gomez-Barvo (eds.), *Motion and Operation Planning of Robotic Systems*, Mechanisms and Machine Science 29,
DOI 10.1007/978-3-319-14705-5_5

1 Introduction

In this chapter we give a brief overview of the most commonly applied obstacle-avoidance algorithms. In general, the algorithms can be divided into global and local. While the former rely on planning, the latter are control-based. We present different control-based approaches, that rely on kinematic algorithms to avoid the obstacles with the end-effector or with any other part of the body of the robot. We also discuss how to include obstacle-avoidance algorithms in novel trajectory-generation methods, such as dynamic movement primitives.

Just as with humans, robotic mechanisms have to act in environments with other objects and agents moving around, interacting with them, influencing the very same environment. The environment can be highly structured, like an industrial setting, or it can be very cluttered, like a kitchen or a workshop. Contact between the robot and an object is very likely to happen in any environment. The contacts can be part of the task, but they may very well also be an undesired event, and consequently, it is necessary to give the highest priority to avoiding them. Different obstacle algorithms have been proposed for this to ensure that tasks that demand no contact with objects, perceived as obstacles either at the end-effector or at any other point of the robot, can be successfully fulfilled.

A natural strategy of obstacle avoidance is to move the manipulator into a configuration where it is not in contact with the obstacle. In order to avoid interference with the motion of the end-effector, redundant degrees of freedom (DOFs) have to be utilized to achieve a collision-free configuration. The amount of flexibility depends on the degree of redundancy, i.e., on the number of redundant DOFs. The kinematic control of redundant mechanisms, where the redundancy is defined as the difference between the required and available DOFs, was thoroughly studied [1–4].

Two different strategy classes can be employed when solving the obstacle-avoidance problem, i.e., global and local. Global strategies rely on planning. They guarantee to find a collision-free path from the initial point to the goal point, if such a path exists. Typically, they are applied in the configuration space, which is also where the manipulator and all the obstacles are mapped. A collision-free path is found in the unoccupied portion of the configuration space [5–7]. One of the major drawbacks is that such methods rely on the assumption that the environment is not changing, as the computational complexity of the algorithms prevents any re-calculation within the typical response time of a manipulator. Despite efforts to reduce the computational complexity of such global algorithms [8–10], these methods cannot offer ability for real-time implementations. This limits their applicability to static and well-defined environments.

Local strategies, on the other hand, treat obstacle avoidance as a control problem. They exploit the capabilities of low-level control, e.g., they can use the sensor information to change the path if an obstacle appears or moves in the workspace. They are primarily suitable when the obstacle position is not known in advance, but is detected in real-time during the task's execution. In this sense, they are not meant to replace the global, higher-level path-planning methods. Local methods are also

computationally less demanding than global methods. However, local methods may cause suboptimal behavior or may even become stuck when a collision-free path cannot be found from the current configuration.

The collision avoidance of redundant manipulators was thoroughly studied [11–20]. The approach proposed by Maciejewski and Klein [17] is to assign to the critical point an avoiding task-space motion, with which the point is then moved away from the obstacle. Colbaugh et al. [12, 13] used configuration control and they defined the constraints representing the obstacle avoidance. On the other hand, Khatib [15] proposed to use potential fields where obstacles generate repulsive forces that prevent the roboto to come too close to the obstacle. Similar approaches were used later by several authors proposed potential functions where a repulsive potential is assigned to obstacles and an attractive potential is assigned to the goal position [16, 18, 20–25]. Yet another approach uses the optimization of an objective function maximizing the distance between the manipulator and the obstacles [14].

Many of the methods are applied at the kinematic level of control, using null-space velocity control for the internal motion of a redundant manipulator. However, some of the control strategies are acceleration based or torque based, considering also the manipulator dynamics [11, 15, 26, 27]. It has been established that certain acceleration-based control schemes exhibit instabilities [28]. An alternative is the augmented Jacobian, as introduced in [2]. Here, a secondary task is added to the primary task to obtain a square and, therefore, an invertible Jacobian matrix. The drawback to this technique is the algorithmic singularities, which occur when the secondary task causes a conflict with the primary task. The use of the second-order inverse kinematic, either at the torque or acceleration level, was thoroughly explored by Khatib [29], resulting in the recent task-prioritized humanoid applications [30–32].

Most of the local obstacle-avoidance strategies at the kinematic level aim at assigning a motion component away from the obstacle for every point on the manipulator close to the obstacle [12–14, 16, 17, 19]. A similar situation applies to the presented proposed strategies. The emphasis of the presentation is on the definition of the avoiding motion. The latter is typically defined in Cartesian space, and this can be used to define the obstacle avoidance as a simple one-dimensional problem, with a one-dimensional operational space for each critical point. This avoids singularity issues when the redundancy level is locally too low. Alternatively, an approximative calculation can be used for the avoiding motion. In contrast to the exact avoiding motion as proposed in [17], the obtained velocity direction does not exactly coincide with the direction away from the obstacle [33]; however, the calculation is faster. In the case of multiple obstacles the situation is even more complex and more specific methods have to be applied, which also consider the relationship between the obstacles and the required avoidance movements. In the chapter we discuss strategies that consider multiple, simultaneously active obstacles in the neighborhood of the robot.

Control of a manipulator, that is redundant with respect to the task can be broken down to control subtasks with different priorities. The main, also called the primary, task is commonly associated with the end-effector pose (position and orientation). Other sub-tasks, such as obstacle avoidance, joint configuration, etc., are then given

lower priorities. Sometimes, this is not the case. For example, the safety of the robot or objects/people in its workspace could be more important, and should also be fulfilled if the end-effector motion is disturbed. In dynamical environments the priority of the tasks can also change with time. In general, task-priority algorithms do not provide a simple means of changing the priority of tasks or transitions between them [34]. In the chapter we present a formulation that makes the end-effector pose the secondary task and obstacle avoidance the primary one. The novelty is in making the primary task (the obstacle avoidance) active only when necessary, i.e., only when the robot crosses a predefined distance-to-the-obstacle threshold. In this aspect, while far from the obstacle, the algorithm allows undisturbed control of the secondary task (as if it were the primary task) [35–37]. Upon reaching the threshold distance, the primary task (obstacle-avoidance) smoothly takes over and only allows motion in the null-space of the primary task. A similar approach was proposed by Sugiura et al. [38], who proposed a blending solution for the end-effector motion, and by Mansard et al. [30], with a generic solution to build a smooth control law for any kind of unilateral constraints.

The last approach we present is solving the obstacle-avoidance problem with the use of novel methods of generating and encoding trajectories with dynamical systems. We show how DMPs offer the means for on-line modulation and adaption of the trajectory in order to take into account the dynamic events from the environment. Introducing a coupling term to the dynamical equations encoding the trajectory, we can modulate its spatial evolution to avoid an obstacle. The choice of the coupling term may be specialized for a given task. Various aspects and applications of the proposed dynamical systems approach are discussed and evaluated.

The computational efficiency of the proposed algorithms, both at the kinematic level using classic control, and using the dynamical systems, allows real-time application in cluttered and/or time-varying environments. We demonstrate the applicability with simulations of a 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 redundant serial manipulators. We consider the robot as a redundant system when the dimension of the joint space n exceeds the dimension of the task space m. The difference between n and m is denoted as the degree of redundancy $r = n - m$. Note that this definition of 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 specific task.

The relationship between the configuration variable $\boldsymbol{q}$ and the task variable $\boldsymbol{x}$ can be described by the following equation

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}) \tag{1}$$

where $\boldsymbol{f}$ is an m-dimensional vector function. The corresponding relationship between the joint velocities $\dot{\boldsymbol{q}}$ and the task velocities $\dot{\boldsymbol{x}}$ is obtained by differentiating (1)

$$\dot{\boldsymbol{x}} = \mathbf{J}\dot{\boldsymbol{q}} \tag{2}$$

where $\mathbf{J}$ is the $m \times n$ Jacobian matrix. The control problem is how to generate the motion in joints that will result in the desired task-space motion. At the velocity kinematic level this means calculating $\dot{\boldsymbol{q}}$ using the desired task-space velocities $\dot{\boldsymbol{x}}$. For a non-redundant manipulator ($n = m$) and when the robot is not in a singular configuration $\dot{\boldsymbol{q}}$ ($\mathbf{J}$ has full rank, $\text{rank}(\mathbf{J}) = n$) the joint velocities $\dot{\boldsymbol{q}}$ can be calculated from (2) as

$$\dot{\boldsymbol{q}} = \mathbf{J}^{-1}\dot{\boldsymbol{x}} \tag{3}$$

where $\mathbf{J}^{-1}$ is the inverse of the Jacobian matrix $\mathbf{J}$. To avoid any drifts, a task-space controller is usually implemented for $\dot{\boldsymbol{x}}$, namely

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_e + \mathbf{K}\boldsymbol{e} \tag{4}$$

where $\dot{\boldsymbol{x}}_e$ is the desired task-space velocity, $\boldsymbol{e}$, $\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}$, is the task-space error, and $\mathbf{K}$ is a positive definite gain matrix.

In the case of a kinematically redundant manipulator, the manipulator possesses more DOFs then required to execute a task, i.e., the dimension of the joint space n exceeds the dimension of the task space m, $n > m$. It is obvious that the Jacobian $\mathbf{J}$ is no longer a square matrix, but an $m \times n$ matrix, and hence the inverse $\mathbf{J}^{-1}$ does not exist and (3) cannot be used. The classic general solution of (2) for a kinematically redundant manipulator is

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}} + \mathbf{N}\dot{\boldsymbol{\varphi}} \tag{5}$$

where $\mathbf{J}^{\#}$ is a generalized inverse of the Jacobian matrix $\mathbf{J}$, $\mathbf{N}$ is a matrix representing the projection into the null space of $\mathbf{J}$, and $\dot{\boldsymbol{\varphi}}$ is an arbitrary n-dimensional joint-velocity vector. From (5) it is clear that $\mathbf{N}$ projects the velocity $\dot{\boldsymbol{q}}_n$ into the null-space of $\mathbf{J}$ and the corresponding motion does not affect the task motion. Remarkably, there is an infinite number of solutions $\dot{\boldsymbol{q}}$. In most cases it is required to pursue a minimum-norm velocity leading, to the selection of the Moor-Penrose inverse $\mathbf{J}^{+}$, $\mathbf{J}^{+} = \mathbf{J}^{T}(\mathbf{J}\mathbf{J}^{T})^{-1}$, as the generalized inverse in (5)

$$\dot{\boldsymbol{q}} = \mathbf{J}^{+}\dot{\boldsymbol{x}} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\dot{\boldsymbol{\varphi}} \tag{6}$$

The first r.h.s. term in (6), i.e., the particular solution, provides the least-squares solution, i.e., it minimizes $\|\dot{\boldsymbol{x}} - \mathbf{J}\dot{\boldsymbol{q}}\|$, with a minimum joint-velocity norm. With the second r.h.s. term in (6) different joint velocities $\dot{\boldsymbol{q}}$ can be obtained that result in the same end-effector velocity $\dot{\boldsymbol{x}}$. This additional joint motion can be exploited to

achieve some additional goals, i.e., some kind of optimization, obstacle avoidance, to fulfill some functional constraints or to execute additional constraint tasks. To perform this additional subtask, the velocity $\dot{\boldsymbol{\varphi}}$ is used. Then the secondary task is defined by some motion $\boldsymbol{x}_t = \boldsymbol{f}_t(\boldsymbol{q})$ like in the case of obstacle avoidance, the velocity $\dot{\boldsymbol{\varphi}}$ can be defined as

$$\dot{\boldsymbol{\varphi}} = \mathbf{J}^{+}\dot{\boldsymbol{x}} \tag{7}$$

Another possibility is to define $\dot{\boldsymbol{\varphi}}$ as

$$\dot{\boldsymbol{\varphi}} = \mathbf{K}_p \nabla p, \tag{8}$$

where, p is a function representing the desired performance criterion, ∇p is the gradient of p, and $\mathbf{K}_p$ is a gain. So, using (8) the optimization of p can be achieved.

3 Obstacle-Avoidance Strategy

The obstacle-avoidance problem usually defines how to control the manipulator in order 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 possible obstacles the manipulator has to move away from them into a configuration where the distance between them becomes larger, as shown in Fig. 1. Reconfiguration of the manipulator without changing the motion of the end-effector is only possible if the manipulator has redundant DOFs. Note that in some cases it is possible that the redundant manipulator cannot avoid an obstacle, because it might be in a configuration where the avoiding motion in the desired direction is not feasible. Having a high degree of redundancy reduces the chance of getting into a such configuration, especially if the manipulator is working in an environment that has many potential collisions with obstacles.

Usually, the basic strategy for obstacle avoidance is to identify the points on the robotic arm that are near obstacles and then assign to them the motion component that moves those points away from the obstacle, as shown in Fig. 1. The robot motion (configuration) is changed if at least one part of the robot is at a critical distance from an obstacle. We denote the obstacles that are closer to the critical distance as the *active obstacles* and the corresponding closest points on the body of the manipulator as the *critical points*.

For industrial robots it is usually assumed that the motion of the end-effector is not disturbed by any obstacle. If such a situation occurs, either the task execution has to be interrupted and the higher-level path planning has to recalculate the desired motion of the end-effector or if the path-tracking accuracy is not important the control algorithms that move the end-effector around obstacles on-line can be used.

Since the position of the obstacle is usually not known in advance, the obstacle-avoidance algorithm must work in real-time. In order to ensure these requirements

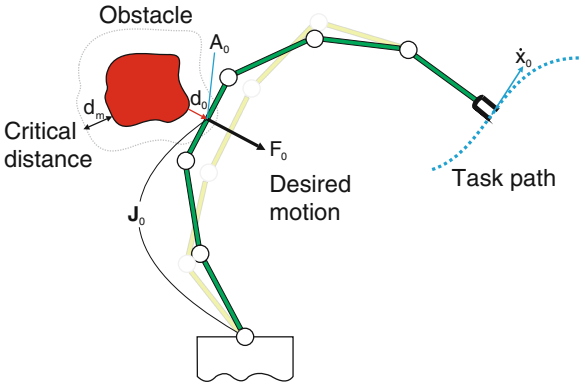

Fig. 1 Manipulator motion in the presence of some obstacles

some sensors have to be used to determine the position of the obstacles or to measure the distance between the obstacles and the body of the manipulator. There is a variety of sensor systems that can be used for such obstacle detection. In many cases a vision system is used to detect obstacles. Another possibility is offered by tactile sensors, like artificial skin, which can detect the obstacle only if they touch it, or by proximity sensors, which can sense the presence of an obstacle in the neighborhood.

4 Obstacle Avoidance Using Kinematic Control

The basic strategy for obstacle avoidance considers the obstacle-avoidance problem at the kinematic level. We denote $\dot{\boldsymbol{x}}_e$ as the desired velocity of the end-effector, and A_o as the critical point on the obstacle (see Fig. 1). To avoid a possible collision, one possibility is to assign a velocity to A_o such that it would move the manipulator away from the obstacle, as proposed in [17]. Here, the motion of the end-effector and the critical point can be defined as

$$\mathbf{J}\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}_e \qquad \mathbf{J}_o\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}_o \tag{9}$$

where $\mathbf{J}_o$ is a Jacobian matrix associated with the point A_o. In the following, different possibilities for finding the solution for both equations will be presented.

4.1 Exact Solution

Let $\dot{\boldsymbol{x}}$ in (5) be equal to $\dot{\boldsymbol{x}}_e$. Then, by combining (5) and (9) we obtain

$$\dot{\boldsymbol{\varphi}} = (\mathbf{J}_o\mathbf{N})^{\#}(\dot{\boldsymbol{x}}_o - \mathbf{J}_o\mathbf{J}^{\#}\dot{\boldsymbol{x}}_e) \tag{10}$$

Using $\dot{\boldsymbol{\varphi}}$ in (5) gives the final solution for $\dot{\boldsymbol{q}}$ in the form

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

Note that $\mathbf{N}$ is both hermitian and idempotent [4, 17]. Here the first term $\mathbf{J}^{\#}\dot{\boldsymbol{x}}$ guarantees the tacking of the desired end-effector. Also, $\dot{\boldsymbol{x}}$ is used in (11) instead of $\dot{\boldsymbol{x}}_e$ to indicate that a task-space controller can be used to compensate for any task-space tracking errors

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_d + \mathbf{K}\boldsymbol{e}. \tag{12}$$

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

$$\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}. \tag{13}$$

Here, $\boldsymbol{x}_d$ is the desired task-space position. The second term in (11), 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's 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 with Reduced Operational Space

The system's ability to avoid obstacles is defined with the matrix $\mathbf{J}_o\mathbf{N}$, which combines the kinematics of the critical point A_o and the null-space matrix of the whole manipulator. Here, the properties of the matrix $\mathbf{J}_o\mathbf{N}$ depend on the position of the point A_o and also on the definition of the operational space associated with the critical point A_o. Usually, all the critical points are defined in Cartesian space, which implies that 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 means that at least 3 DOFs are needed to move one point from an obstacle. Consequently, it might seem that a manipulator with two redundant DOFs is not capable of avoiding obstacles. However, we know from our experience that this is not true. For example, consider a planar 3 DOF manipulator that can move along a straight line and only the positions of the end-effector are important. In this case, the task space is 2-dimensional and the manipulator has one free 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 one 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 one, and the pseudo-inverse $(\mathbf{J}_o\mathbf{N})^{\#}$ does not give a feasible solution, at least the desired avoiding velocity $\dot{\boldsymbol{x}}_o$ cannot be achieved.

On the other hand, 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 for which only one degree of redundancy is needed. Therefore, we propose using a reduced operational space [39] 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 that relates the joint-space velocities $\dot{\boldsymbol{q}}$ and the velocity in the direction of $\boldsymbol{d}_o$ can be calculated as

$$\mathbf{J}_{d_o} = \boldsymbol{n}_o^T \mathbf{J}_o \tag{14}$$

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 also much faster [33, 35, 39]. Note that in this case 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.

4.3 Selection of Avoiding Velocity

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

$$\dot{x}_o = \alpha_v v_o \tag{15}$$

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

$$\alpha_v = \begin{cases} \left(\frac{d_m}{\|\boldsymbol{d}_o\|}\right)^2 - 1 & \text{for} \quad \|\boldsymbol{d}_o\| < d_m \\ 0 & \text{for} \quad \|\boldsymbol{d}_o\| \geq d_m \end{cases} \tag{16}$$

where d_m is the critical distance to the obstacle. If the obstacle is too close ($\|\boldsymbol{d}_o\| \leq d_b$) the main task should be stopped. 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 ensure 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, which results in a rapid movement far from the obstacle. Such motion is undesirable and may cause problems, especially if there are more obstacles in close proximity. Namely, the manipulator may bounce between them. On the other hand, too small a value of v_o would not move the manipulator away from the critical point, which is undesirable as well. Selecting the right v_o is a trade-off between how quickly and how smoothly the robot avoids the obstacle.

For smoothing the motion Maciejewski et al. [17] proposed a factor α_h, which changed the amount of homogenous solution to be included in the total solution

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}} + \alpha_h(\mathbf{J}_{d_o}\mathbf{N})^{\#}(\dot{\boldsymbol{x}}_o - \mathbf{J}_{d_o}\mathbf{J}^{\#}\dot{\boldsymbol{x}}_e) \tag{17}$$

In our case we have selected α_h as

$$\alpha_h = \begin{cases} 1 & \text{for} \quad \|\boldsymbol{d}_o\| \leq d_m \\ \frac{1}{2}\left(1 - \cos\left(\pi \frac{\|\boldsymbol{d}_o\| - d_m}{d_i - d_m}\right)\right) & \text{for} \quad d_m < \|\boldsymbol{d}_o\| < d_i \\ 0 & \text{for} \quad d_i \leq \|\boldsymbol{d}_o\| \end{cases} \tag{18}$$

where d_i is the distance at which the obstacle influences the motion. Note 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 decreases when approaching from d_i to d_m, if the obstacle is not moving. With such a selection of α_v and α_h, smooth velocities can be obtained.

The control law given by (17) was derived for a single obstacle. When more than one obstacle is active at the same time, then the worst-case obstacle, which is the nearest, has to be used. This solution may result in discontinuous velocities and may cause oscillations in some cases. In particular when switching between active obstacles the particular homogenous solutions are not equal and a discontinuity in the joint velocities may occur. To improve this behavior we propose using a weighted sum of the homogenous solution of all the active obstacles

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}} + \sum_{i=1}^{n_o} w_i \alpha_{h,i} \dot{\boldsymbol{q}}_{h,i} \tag{19}$$

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 ith active obstacle, respectively. The weighting factors w_i are calculated as

$$w_i = \frac{d_i - \|\boldsymbol{d}_{o,i}\|}{\sum_{i=1}^{n_o}(d_i - \|\boldsymbol{d}_{o,i}\|)} \tag{20}$$

Although the actual velocities in the critical points differ from the desired ones, using an exact solution significantly improves the performance.

As an illustration we present the simulation of a planar manipulator with five revolute joints. The primary task is to move along a straight line from point P_1 to point P_2. The desired trajectory is shown by the green line in Fig. 2. The task

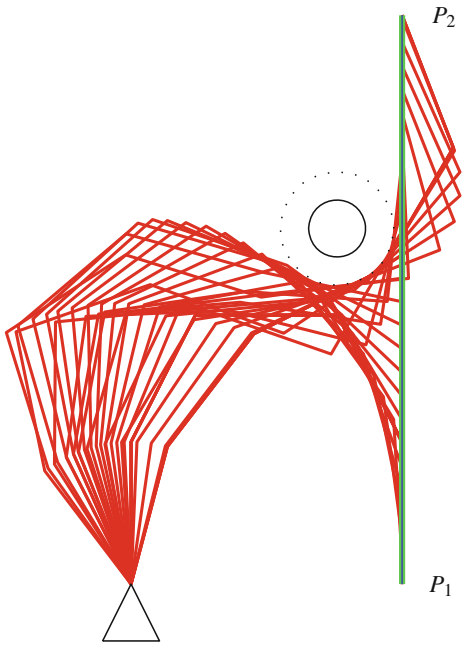

Fig. 2 Planar 5 DOF manipulator: tracking of a line from point P_1 to P_2 and obstacle avoidance using an exact solution

trajectory has a trapezoid velocity profile with an acceleration of $4\,\mathrm{ms}^{-2}$ and a max. velocity of $0.4\,\mathrm{ms}^{-1}$. We chose the critical distance $d_m = 0.2\,\mathrm{m}$ and the radius of the obstacle was $r = 0.2\,\mathrm{m}$. The initial configuration of the manipulator was selected such that the motion was obstructed by an obstacle. The simulation results using the exact velocity controller EX (17) are presented in Figs. 2 and 3.

In the top plot in Fig. 3 we can see that the critical distance d_m is always above the predefined threshold $d_0 = 0.2$. However, in the middle plot we can see that with the exact method in some cases the joint velocities may not be smooth, which may also reflect in the tracking accuracy, as shown in the bottom plot. Even so, note that the tracking accuracy of the end-effector is in the range of 10^{-6}.

4.4 Approximate Solution

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

$$\dot{\boldsymbol{\varphi}} = \mathbf{J}_{d_o}^{\#} \dot{x}_o \tag{21}$$

without compensating for the contribution of the end-effector motion and then substituting $\dot{\boldsymbol{\varphi}}$ into (5) yields

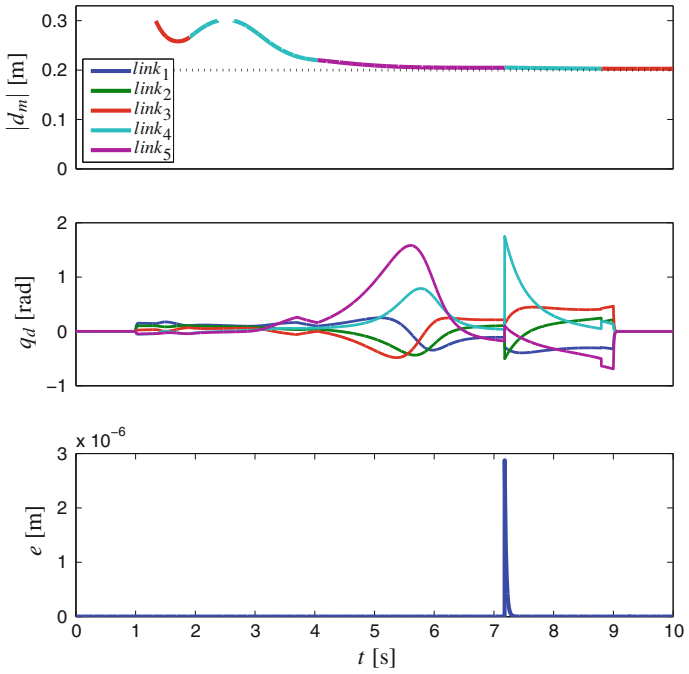

Fig. 3 *Top plot* shows the distance between the obstacle and the nearest link. *Middle plot* shows the joint velocities and the *bottom plot* shows the end-effector tracking error

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}} + \mathbf{N}\mathbf{J}_{d_o}^{\#}\dot{x}_o \tag{22}$$

This approach avoids the singularity problem of $(\mathbf{J}_{d_o}\mathbf{N})$ [1]. The formulation (22), however, does not guarantee that the desired $\dot{x}_o$ will be exactly achieved even if the degree of redundancy is sufficient. This is because in general $\mathbf{J}_{d_o}\mathbf{N}\mathbf{J}_{d_o}^{\#}\dot{x}_o$ is not equal to $\dot{x}_o$.

To avoid the obstacle the goal velocity in A_o is represented by the vector $\dot{\boldsymbol{x}}_o$. Using the original method (11) the velocity in A_o is exactly $\dot{\boldsymbol{x}}_o$. The joint velocities in the exact solution ensure that the component of the velocity at point A_o (i.e., $\mathbf{J}_o\dot{\boldsymbol{q}}$) in the direction of $\dot{\boldsymbol{x}}_o$ is as required. The approximate solution gives, in most cases, a smaller magnitude of the velocity in the direction of $\dot{\boldsymbol{x}}_o$. Therefore, the manipulator moves closer to the obstacle when an approximate solution is used. This is not so critical, because the minimum distance also depends on the nominal velocity v_o, which can be increased to achieve larger minimum distances, if needed. Additionally, the approximate solution possesses certain advantages when many active obstacles have to be considered. The joint velocities can be calculated as

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\#}\dot{\boldsymbol{x}} + \mathbf{N}\sum_{i=1}^{n_o}\mathbf{J}_{d_o,i}^{\#}\dot{x}_{o,i} \tag{23}$$

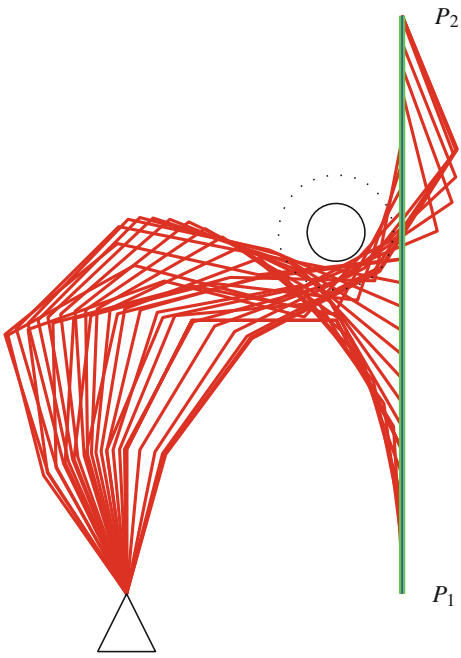

Fig. 4 Planar 5 DOF manipulator: tracking of a line from point P_1 to P_2 and obstacle avoidance using an approximate solution

where n_o is the number of active obstacles and, therefore, the matrix $\mathbf{N}$ has to be calculated only once. However, the pseudo-inverses $\mathbf{J}_{o,i}^{\#}$ have to be calculated for each active obstacle.

We have implemented the approximate velocity controller AP (22) for the same system and the task as shown in Figs. 2 and 3. The results are presented in Figs. 4 and 5. We can see that the links are coming closer to the obstacle compared to the case of using the exact controller. Note that discontinuities in the joint velocities may also occur here, and that the tracking error of the end-effector is in the same range as in the case of the exact controller.

4.5 Experimental Results

To support the simulation results we applied the obstacle-avoidance control using the approximated solution (23) to the 7 DOF Kuka LWR robot. The primary task for the robot was manipulating the ball in the Cartesian task space and the secondary task was avoiding human contact (a human was treated as an obstacle for the robot). The experimental setup is shown in Fig. 6.

The human motion is captured using the Microsoft Kinect sensor. Microsoft Kinect is based on a range camera developed by PrimeSense, which interprets 3D scene information from a continuously projected infrared structured light.

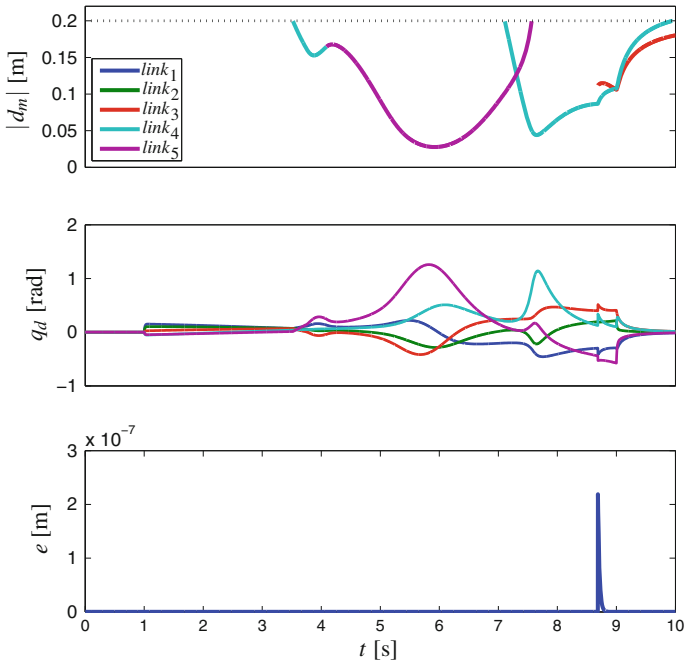

Fig. 5 *Top plot* shows the distance between the obstacle and the nearest link. *Middle plot* shows the joint velocities and the *bottom plot* shows the end-effector tracking error

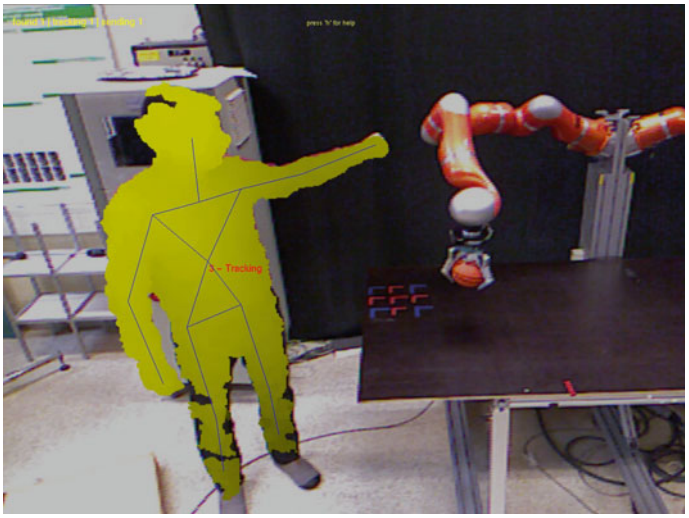

Fig. 6 Experimental setup for the manipulation task with the KUKA LWR robot, while avoiding the human in the robot workspace. The picture is taken with the Microsoft Kinect camera. Note that the picture from the Microsoft Kinect camera is mirrored

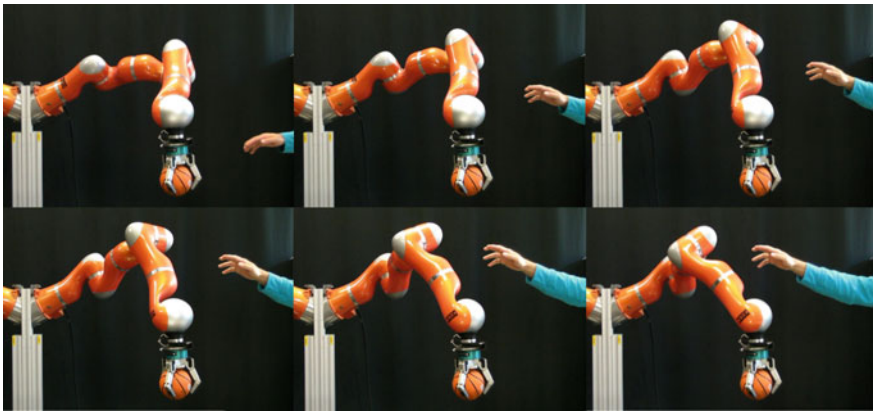

Fig. 7 The image sequence shows obstacle avoidance using an approximate solution

By processing the depth image, the PrimeSense API enables the tracking of human limb movements in real time. To acquire the closest points (interpreted as point obstacles) between the human and the robot, we calibrated the Microsoft Kinect sensor to the robot base coordinate system. To obtain the proper transformation matrix, we recorded at least four pairs of points in both coordinate systems. During the calibration procedure the human placed his hand at the same locations as the robot end-effector and the position of the human hand and the position of end-effector were measured in the Kinect and robot base coordinate systems, respectively. The transformation matrix was calculated using least-squares fitting of two points set, as described in [40].

The results are shown as a sequence of photos in Fig. 7, where we can see a successful pose adaptation in order to prevent human contact, while maintaining the position of the end-effector.

5 Obstacle Avoidance as a Primary Task

The development of multi-arm robot mechanisms and humanoid robots emphasized the importance of being able to perform multiple tasks simultaneously [41–43], like controlling multiple points on the robot structure, stability, pose control or obstacle avoidance. Whether it is feasible that the robot can achieve all the goals at the same time depends on the one hand upon the robot's dexterity and its configuration, and on the other hand upon the goals themselves. Although highly redundant robot manipulators can perform multiple tasks, it is not likely that all the tasks can be fulfilled simultaneously or at least not all the time. For example, the robot may be able to perform all the tasks in one configuration, but when the robot moves to another configuration, some goals may become conflicting with the motion. In this case, it is

impossible to satisfy all the goals and the conflict can be handled in the framework of the task priority, where the tasks are arranged by their relevance. The priority indicates how important a task is compared to others and it can also imply some other things, like how important it is to execute the task accurately. Typically, the lower-priority tasks are less important and they are fulfilled completely only if not they are interfering with higher-priority tasks. The task with the highest priority is usually referred to as the primary task.

With multiple tasks it is important to know the relationship between the tasks. Assuming that each task can be executed per se, i.e., a feasible solution exists for all the tasks, this is not a guarantee that all tasks can be executed simultaneously. Namely, the motion necessary to perform one task can disturb the execution of other tasks and, hence, some tasks may become unfeasible with respect to others. The dependency between tasks can be determined by analyzing the range of the associated Jacobian inverse mappings [44–46]. It is important to know the relationship between the mapping, but it is not essential for the solution. When two tasks are disturbing each other, then it is necessary to ensure that the task with higher priority is fulfilled and then we should try to fulfil the lower-priority task as well as possible.

For a redundant robot one possible solution for obstacle avoidance is to consider the obstacle-avoidance task as a primary task T_a, and the end-effector tracking as a secondary task T_b defined by

$$\boldsymbol{x}_a = \boldsymbol{f}_a(\boldsymbol{q}) \qquad \boldsymbol{x}_b = \boldsymbol{f}_b(\boldsymbol{q}) \tag{24}$$

For each of the tasks, the corresponding Jacobian matrices can be defined as $\mathbf{J}_a$ and $\mathbf{J}_b$, with the corresponding null-space projections denoted by $\mathbf{N}_a$ and $\mathbf{N}_b$. Assuming that task T_a is the primary task, Eq. (5) can be rewritten as

$$\dot{\boldsymbol{q}} = \mathbf{J}_a^{\#}\dot{\boldsymbol{x}}_a + \mathbf{N}_a\mathbf{J}_b^{\#}\dot{\boldsymbol{x}}_b \tag{25}$$

Previously, 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. If such a situation occurs, the task execution usually 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 (25). Consequently, no end-effector path recalculation or higher-level path planning is needed.

Figure 8 shows an example of the prioritized control where we can see that in this case the robot can avoid obstacles even if they appear on the Cartesian task path. The same parameter set was used as in Sect. 4, except for the obstacle diameter, which was now set to $r = 0.4\,\mathrm{m}$. In Fig. 9 we can also see that the critical distance d_m is exactly the same as the predefined $d_0 = 0.2$, which was expected since the obstacle-avoidance task is now the task with the highest priority. In contrast, in this particular example, we can see that such an approach has a disadvantage when compared to the global path search algorithms since the resulting motion may be suboptimal and as a result it may become stuck.

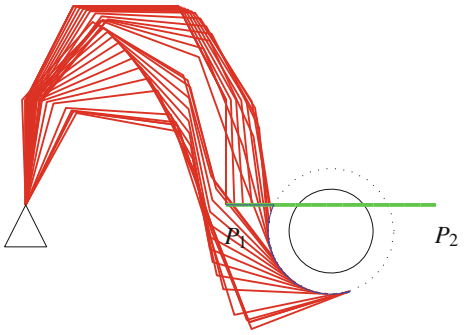

Fig. 8 Planar 5 DOF manipulator: tracking of a line from point P_1 to P_2 is a secondary task and obstacle avoidance is the primary task

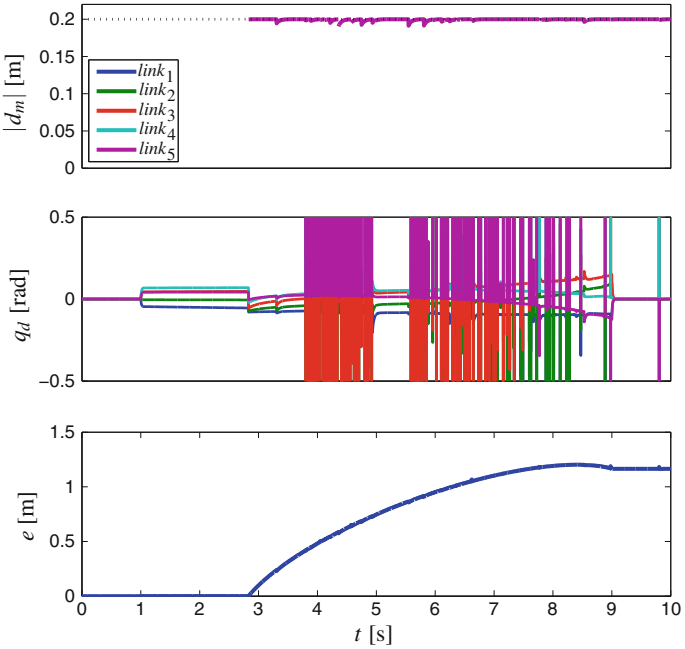

Fig. 9 *Top plot* shows the distance between the obstacle and the nearest link. *Middle plot* shows the joint velocities and the *bottom plot* shows the end-effector tracking error

5.1 Smooth Transition Between Tasks

Another important aspect that should be considered with multiple tasks is the ability to change the task priority. When a robot is working in a changing environment, it may happen that the situation requires that one task becomes more important than before. A good example is obstacle avoidance, where the priority of the avoiding task

may depend on the type of obstacle and on the distance to the obstacle. Therefore, it is beneficial if the control method enables a smooth change of task priorities. Using formulation (25) this cannot be done in a smooth way. Therefore, we propose a new definition of the velocity $\dot{\boldsymbol{q}}$ [35]. The velocity $\dot{\boldsymbol{q}}$ is now defined as

$$\dot{\boldsymbol{q}} = \mathbf{J}_a^{\#}\dot{\boldsymbol{x}}_a + \mathbf{N}'_a\mathbf{J}_b^{\#}\dot{\boldsymbol{x}}_b, \tag{26}$$

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

$$\mathbf{N}'_a = \mathbf{I} - \lambda(\boldsymbol{x}_a)\mathbf{J}^{\#}\mathbf{J}, \tag{27}$$

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 λ is $\lambda(\boldsymbol{x}_a) = 1$, and when the task T_a is not active, it is $\lambda(\boldsymbol{x}_a) = 0$.

The proposed algorithm allows a smooth transition in both ways, i.e., between observing the task T_a and the task T_b in the null-space of the task T_a or just the unconstrained movement of the task T_b. The proposed approach is general and can be used for different robotic tasks.

For obstacle avoidance using (26), we define the primary task T_a to be the motion in the direction $\boldsymbol{d}_0$ and the motion of the end-effector to be the task T_b. Using the reduced operational space yields

$$\mathbf{J}_a = \mathbf{J}_{d_o}, \tag{28}$$

$$\mathbf{J}_b = \mathbf{J}. \tag{29}$$

Next, (26) can be rewritten in the form

$$\dot{\boldsymbol{q}} = \mathbf{J}_{d_o}^{\#}\dot{\boldsymbol{x}}_o + \mathbf{N}'_0\mathbf{J}^{\#}\dot{\boldsymbol{x}}. \tag{30}$$

Here, $\dot{\boldsymbol{x}}$ is the task controller for the end-effector tracking and let $\lambda(\boldsymbol{d}_0) = \alpha_h$, then $\mathbf{N}'_0$ is given by

$$\mathbf{N}'_0 = \mathbf{I} - \alpha_h\mathbf{J}_o^{\dagger}\mathbf{J}_o. \tag{31}$$

Formulation (30) allows an unconstrained joint movement while α_h is close to zero ($\alpha_h \approx 0$). 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 ($\alpha_h \approx 1$), the null space in (31) 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.

Simulation results using the control algorithm (30) are presented in Figs. 10 and 11. We can see in Fig. 10 and in the top plot of Fig. 11 that in the case of a smooth transition between tasks the tracking error may become significant while

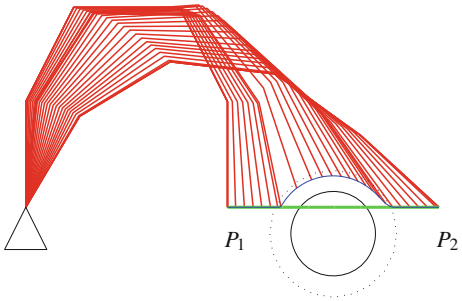

Fig. 10 Planar 5 DOF manipulator: smooth transition between the primary task of obstacle avoidance and the secondary task of tracking a line from point P_1 to point P_2

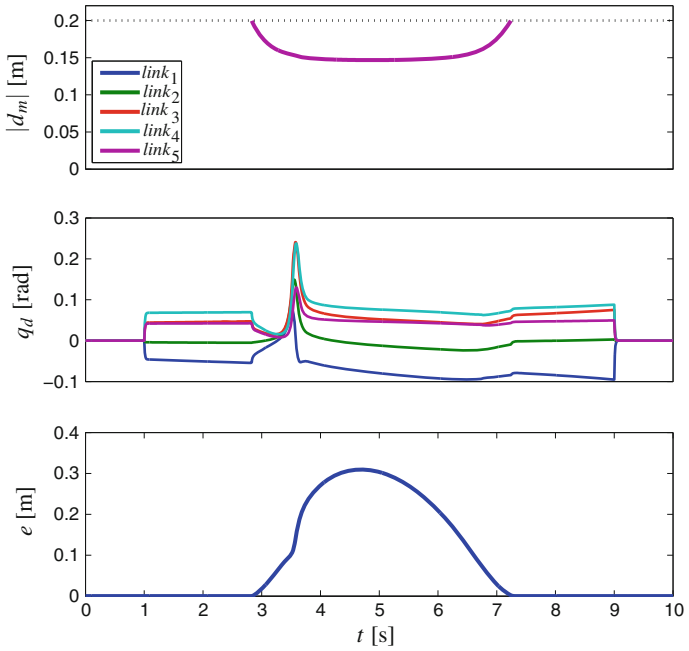

Fig. 11 *Top plot* shows the distance between the obstacle and the nearest link. *Middle plot* shows the joint velocities and the *bottom plot* shows the end-effector tracking error

the robot is close to the obstacle. The main reason for such behaviour is that in this case the obstacle-avoidance becomes primary and the end-effector tracking is the secondary task projected into the null space of the obstacle-avoidance task. Even though this may seen impractical, it is useful in situations when the obstacle is in the path of the end-effector. Since by using such control, the robot can avoid obstacles in real-time without using any additional path-planners if obstacles appear on the end-effector path during the motion.

An important observation is also that for this particular task and for the same configuration and parameter set as used in the example presented in Fig. 9, the robot does not become stuck in the local minimum. The main reason for such behaviour is that the transition to obstacle avoidance is now smooth and consistent. However, as we can see in the top plot in Fig. 11, as a consequence the robot comes closer to the obstacle. Note that this minimal distance to the obstacle could be increased by increasing the value of d_0.

5.2 Prioritized Damped Least-Squares Inverse

Another possibility for simultaneous end-effector tracking and obstacle-avoidance simultaneously is to treat them equally. Let us stack all the tasks the robot should perform $x_i, i = 1, \ldots, k$ into an extended task vector

$$\boldsymbol{x}_E = \left[\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_k^T\right]^T \tag{32}$$

Then, the relation between the task space velocities and the joint velocities is given as

$$\dot{\boldsymbol{x}}_E = \mathbf{J}_E \dot{\boldsymbol{q}} \tag{33}$$

where the extended Jacobian is given in the form

$$\mathbf{J}_E = \left[\mathbf{J}_1^T, \mathbf{J}_2^T, \ldots, \mathbf{J}_k^T\right]^T \tag{34}$$

The solution to (33) (denoted later as E) is given in the form

$$\dot{\boldsymbol{q}} = \mathbf{J}_E^{\#} \dot{\boldsymbol{x}}_E \tag{35}$$

As all the tasks are included in $\dot{\boldsymbol{x}}_E$ there is no need to consider the homogenous part of the solution, i.e., the null-space velocity, to solve these tasks. If the rank of $\mathbf{J}_E$ equals at least the dimension of all the tasks, $\text{rank}(\mathbf{J}_E) \geq m_t$, then the solution to (35) results in $\dot{\boldsymbol{q}}$, which fulfill all the tasks.

Even though the approaches proposed by [2, 44, 47–49], for the calculation of joint velocities in the case of multiple prioritized tasks, solve the inverse kinematic problem when the system of equations is not ill-conditioned, it is likely that during the execution of multiple tasks the manipulator moves toward the configuration where one of the Jacobian matrices is near singularity and, consequently, the obtained joint velocities $\dot{\boldsymbol{q}}$ become unfeasible. To overcome the problem of unfeasible velocities we could apply the damped least-squares (DLS) technique. Applying DLS to the extended Jacobian method gives feasible joint velocities. However, if the rank of the extended Jacobian $\mathbf{J}_E$ is not sufficient with respect to the dimensions of all the tasks

$$rank(\mathbf{J}_E) < \sum_{i=1}^{k} m_i \tag{36}$$

then (35) results in a "best fit" (in a least-squares sense) solution. Since in (35) all the tasks are treated equally, it is not possible to prioritize some of the tasks in favor of others. To overcome this drawback we propose an approach in the framework of a DLS extended Jacobian [48, 50].

The basis of this method is a combination of the extended Jacobian approach (35) and the DLS inverse technique. The proposed solution is given in the form

$$\dot{\boldsymbol{q}} = \mathbf{J}_E^{\#} \dot{\boldsymbol{x}}_E \tag{37}$$

where

$$\mathbf{J}_E^{\#} = \mathbf{J}_E^T (\mathbf{J}_E \mathbf{J}_E^T + \lambda^2 \mathbf{P})^{-1} \tag{38}$$

and (P) is an $m_t \times m_t$ diagonal matrix

$$\mathbf{P} = \begin{bmatrix} p_1 \mathbf{I}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & p_2 \mathbf{I}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & p_k \mathbf{I}_k \end{bmatrix} \tag{39}$$

where p_i are scalars depending on the desired priority of the task T_i, and $\mathbf{I}_i$ are $m_i \times m_i$ unit matrices. We denote this method as the *priority weighted damped least-squares* Jacobian method (denoted later as PWDLS). The proposed solution (38) with priority factors (39) minimizes

$$\lambda^2 \|\dot{\boldsymbol{q}}\| + \sum_{i=1}^{k} p_k \|\dot{\boldsymbol{x}}_k - \mathbf{J}_k \dot{\boldsymbol{q}}\| \tag{40}$$

The method is similar to the method proposed in [50] except that the weighting factors are defined by the priority of the tasks. For improving the performance it is essential to suitably select the factors in the damping term in (38). To focus on the priority issue of the problem, we assume that the optimal value for the damping factor λ has been selected using one of the well-known methods [48, 51–54]. To determine the optimal value of λ all the values p_i are set to 1, i.e., $\mathbf{P} = \mathbf{I}$.

When dealing with the priority in the framework of redundancy resolution, the terms *primary task*, *secondary task*, and so on, imply that the control has fulfilled the primary task first, and next the secondary task, without disturbing the primary task. This philosophy is used by all redundancy-resolution schemes dealing with prioritized tasks. None of the redundancy schemes can deal with the information about "how much" one task is more important than the other. For example, even for

the obstacle-avoidance schemes, where the distance to the obstacle can be used as a measure of the importance of particular critical points, this information is actually used only to order the critical points. On the other hand, the parameters p_i can be used to quantify the relative importance of the tasks T_i. So, it is possible to quantify the priorities of the tasks [55]. It is obvious that the following relation must hold

$$\text{Priority}(T_i) > \text{Priority}(T_j) \Leftrightarrow p_i < p_j, \quad i, j \in \{1, \ldots, k\} \tag{41}$$

To gain more insight into the relation between the tasks T_i one can compare the desired task velocities $\dot{\boldsymbol{x}}$ and the task velocities $\dot{\boldsymbol{x}}_a$ obtained as a solution of (42)

$$\dot{\boldsymbol{x}}_{Ea} = \mathbf{J}_E \dot{\boldsymbol{q}} = \mathbf{J}_E \mathbf{J}_E^{\#} \dot{\boldsymbol{x}}_e = \mathbf{A} \dot{\boldsymbol{x}} \tag{42}$$

The $m_a \times m_a$ matrix $\mathbf{A}$ represents the mapping between $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{x}}_a$ and can be divided into several submatrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \ldots & \mathbf{A}_{1,k} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \ldots & \mathbf{A}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{k,1} & \mathbf{A}_{k,2} & \ldots & \mathbf{A}_{k,k} \end{bmatrix} \tag{43}$$

where $\mathbf{A}_{i,j}$ are $m_i \times m_j$ matrices. Remarkably, the diagonal matrices $\mathbf{A}_{i,i}$ represent the transformation of the task velocity $\dot{\boldsymbol{x}}_i$ in the space of the task T_i, and the off-diagonal submatrices represent the influence between the tasks. Note that as p_i are not equal, $\mathbf{A}$ is a non-symmetric matrix. The explanation is apparent, the task with higher priority influences the task with lower priority more a vice versa.

An example of using the (42) algorithm is shown in Figs. 12 and 13. Here we can see similar behaviour as when using a smooth transition between tasks, e.g., Figs. 10 and 11. By comparing the results, the main difference between those two approaches while using the same parameter set is that in the case of PWDLS the robot comes closer to the obstacle.

In the following we present how the selection of p_i influences the solution of (42). For a better understanding we present a 4 DOF planar manipulator with revolute

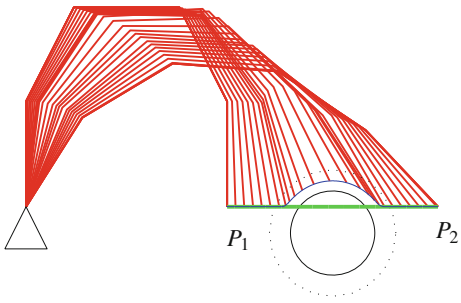

Fig. 12 Planar 5 DOF manipulator: tracking of a line from point P_1 to point P_2 and obstacle avoidance using PWDLS Jacobi

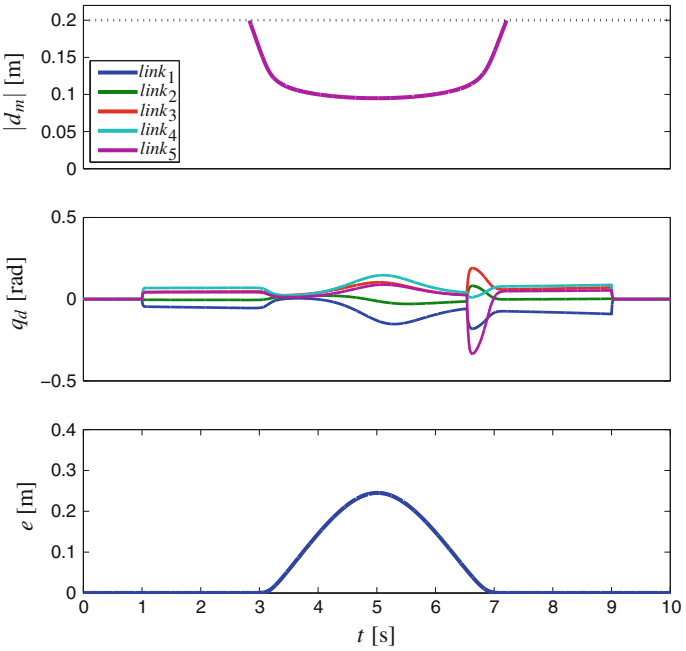

Fig. 13 *Top plot* shows the distance between the obstacle and the nearest link. *Middle plot* shows the joint velocities and the *bottom plot* shows the end-effector tracking error

joints where three control points have to be moved in different directions due to the obstacles near the robot. Note that in this example, the distance between each obstacle and the robot body is the same for all obstacles. Consequently, the desired avoiding motion is similar for all the critical points (except the direction, of course). We assume that only the positions of the control points are important and so the tasks are 2-dimensional, $m_i = 2$. Consequently, $m_t = 6$ and $n = 4$. As $\mathbf{J}_E$ has more rows than columns, the system is overdetermined and no exact solution exists. Figure 14 shows the situation for four different selections of P. The case a) presents the solution without prioritizing tasks (as a classic extended Jacobian approach). The

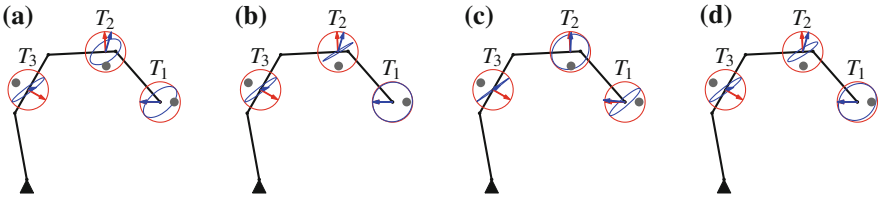

Fig. 14 Influence of different priority factors in (42) for three tasks and for four priorities sets: **a** $p = [1, 1, 1]$, **b** $p = [1, a, a^2]$, **c** $p = [a, 1, a^2]$, **d** $p = [a, a^2, 1]$, where $a = 5$ and $\lambda = 10^{-8}$. The *circles* represent the geometrical representation of submatrices $\mathbf{A}_{i,i}$: unit sphere (*red*) $\rightarrow$ ellipsoid (*blue*). *Red* vectors are the desired task velocities $\dot{\boldsymbol{x}}$ and *blue* vectors are the resulting task vectors $\dot{\boldsymbol{x}}_a$

other three cases show the situation when each of the tasks becomes the main task. Note that the motion in a particular control point is not only due to the desired motion in that point but in other control points the desired motion contributes to. Actually, in case (d) most of the motion in control point 2 is due to the motion of the other two tasks. As one can see, with a suitable selection of p_i the proposed method makes it possible to achieve the desired behavior of the whole system.

As the priority can be defined by changing the controller parameters rather a by changing the controller structure, the proposed method is also suitable when the priority has to change during the tasks' execution. Note that the priority change can be done continuously and no discontinuity in the joint-space solution $\dot{\boldsymbol{q}}$ is experienced. A method for determining the actual values of p_i is beyond the scope of this chapter. In general, it depends on the needs of all the tasks and the specific circumstances during the tasks' execution.

5.3 Experimental Results

To demonstrate the properties of the algorithm given with (42) we extended the task of the bimanual cooperation of two Kuka LWR robots equipped with Barret-Hand grippers holding a plate while balancing a bottle [56] with the task of preventing human contact. As in the case of the experiment in Sect. 4, the human motion was obtained using the Microsoft Kinect sensor. The results are shown in Fig. 16 and as a sequence of photographs in Fig. 15, where we can see that robots are able to successfully perform multiple tasks simultaneously, i.e., preventing human-robot contact and preserving the plate's orientation.

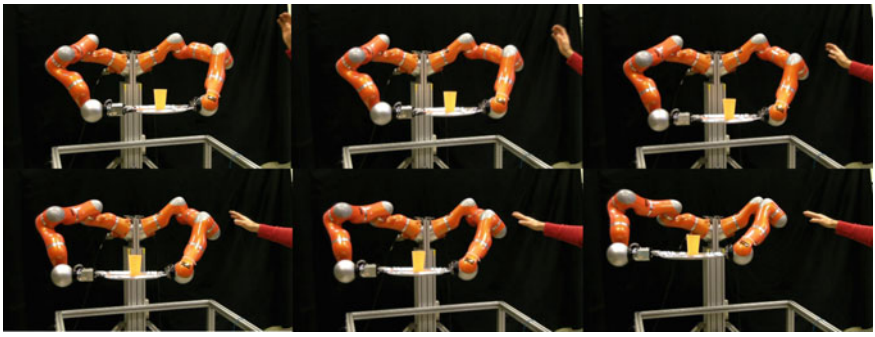

Fig. 15 A sequence of still photographs shows the movement of two Kuka LWR robots, while they successfully avoid a human arm that is approaching the robot in the robot's work space. The detection and tracking of the human arm was done in real time using a Microsoft Kinect sensor

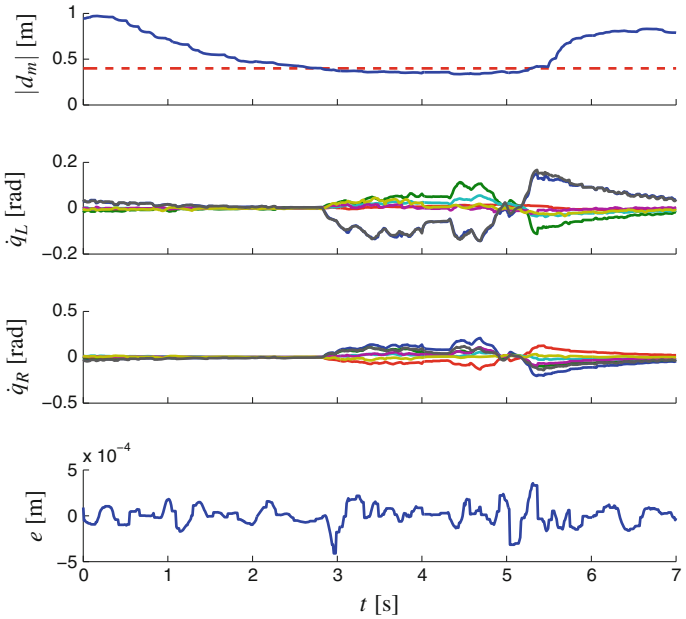

Fig. 16 Results of a bimanual cooperation of two Kuka LWR robots equipped with Barret-Hand grippers holding a plate while balancing a bottle with the task of preventing human contact. The *top plot* shows the closest distance between human and nearest robot link. *Second* and *third plot* shows the joint velocities for obstacle avoidance for left and right robot respectively. *Bottom plot* shows the task error of balancing a bottle on a plate

6 Obstacle Avoidance Using Dynamical Systems

In this section we introduce dynamic movement primitives, which can be used to encode arbitrary trajectories, and are often associated with the learning-by-demonstration approach of controlling robots. We first provide the basics of the dynamic motor primitives, followed by obstacle-avoidance modulation. The obstacle avoidance in the DMP framework presented here is a modified approach of [57]. Simulated and real-world results are presented.

6.1 Dynamic Movement Primitives

The theoretical foundations of the dynamic movement primitives (DMPs) trajectory representation was developed by Ijspeert et al. [58]. Here the discussion is limited to discrete movement primitives, which can encode control policies for discrete point-to-point movements. See [59–61] for the discussion of rhythmic DMPs. The representation proposed by Ijspeert et al. is based on a set of nonlinear

differential equations with a well-defined attractor dynamics. We used the most current formulation as outlined in [57]. For a single degree of freedom denoted by y, which can either be one of the internal joint angles or one of the external task-space coordinates, the following system of linear differential equations with constant coefficients denotes a dynamic movement primitive

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x), \tag{44}$$

$$\tau \dot{y} = z. \tag{45}$$

$f(x)$ is defined as a linear combination of nonlinear radial basis functions

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \tag{46}$$

$$\Psi_i(x) = \exp\left(-h_i\,(x - c_i)^2\right), \tag{47}$$

where c_i are the centers of radial basis functions distributed along the trajectory and $h_i > 0$ their widths. Provided that the parameters $\alpha_z,\ \beta_z,\ \tau > 0$ and $\alpha_z = 4\beta_z$, the linear part of the system (44) and (45) is critically damped and has a unique attractor point at $y = g,\ z = 0$. A phase variable x is used in (44), (46) and (47). It is utilized to avoid the direct dependency of f on time. Its dynamics is defined by

$$\tau \dot{x} = -\alpha_x x, \tag{48}$$

with the initial value $x(0) = 1$. α_x is a positive constant.

The weight vector $\mathbf{w}$, composed of weights w_i, defines the shape of the encoded trajectory. [58, 62] describe the learning of the weight vector. Multiple DOFs are realized by maintaining separate sets of (44)–(47), while a single canonical system given by (48) is used to synchronize them.

6.2 Obstacle Avoidance

A control policy given by the DMP can encode either separate joint trajectories, or external task-space coordinates. Obstacle avoidance in Cartesian space is easier to implement since the trajectory is usually planned in Cartesian space as well. Let us assume a three degree-of-freedom DMP system that encodes point-to-point reaching in Cartesian space. The 3-D position vector of the 3 DOF discrete dynamical system is encoded by $\mathbf{y} = [y_1, y_2, y_3]^T$. The objective is to generate a reaching movement to a goal state $\mathbf{g} = [g_1, g_2, g_3]^T$. On the way to the goal state, an obstacle is positioned at $\mathbf{o} = [o_1, o_2, o_3]^T$ and needs to be avoided. A suitable coupling term $\mathbf{C}_t = [C_{t,1}, C_{t,2}, C_{t,3}]^T$ for the obstacle avoidance can be formulated as follows:

$$\mathbf{C}_t = \gamma\ \mathrm{sig}\,(\|\mathbf{o} - \mathbf{y}\|)\ \mathbf{R}\dot{\mathbf{y}}\ (\pi - \phi)\ \exp\,(-\beta\phi), \tag{49}$$

where

$$\phi = \arccos\left(\frac{(\mathbf{o}-\mathbf{y})^T\dot{\mathbf{y}}}{\|\mathbf{o}-\mathbf{y}\|\|\dot{\mathbf{y}}\|}\right), \tag{50}$$

$$\text{sig}(x) = \frac{1}{1+e^{\eta(x-d)}}, \tag{51}$$

$$\mathbf{R} = \exp\left(\left(\frac{\pi}{2}-\phi\right)\mathbf{n}\right), \tag{52}$$

$$\mathbf{n} = \frac{(\mathbf{o}-\mathbf{y})\times\dot{\mathbf{y}}}{\|\mathbf{o}-\mathbf{y}\|\|\dot{\mathbf{y}}\|}. \tag{53}$$

γ, β, and η are the scaling factors and d is the distance at which the obstacle should start affecting the robot's motion. The coupling term as defined above generates a velocity component that is in a plane defined by the vectors $\mathbf{o}-\mathbf{y}$ and $\dot{\mathbf{y}}$. It is also orthogonal to the line $\mathbf{o}-\mathbf{y}$, which is connecting the tip of the robot and the obstacle.

We can ensure that the tip of the robot, i.e., the end-effector, avoids the obstacle by adding the coupling term $\mathbf{C}_t$ to Eq. (45)

$$\tau\dot{\mathbf{z}} = \alpha_z(\beta_z(\mathbf{g}-\mathbf{y})-\mathbf{z}) + \mathbf{f}(x) + \mathbf{C}_t \tag{54}$$

The resulting behavior is shown in Fig. 17. Note that in this way we can only ensure that the robot tip avoids the obstacle. However, the rest of the robot could still collide with it. Effectively, such an implementation of obstacle avoidance treats the problem of the end-effector collision as the primary task. Given that the DMP encodes a task-space trajectory, the actual joint trajectories are calculated using IK algorithms. Null-space obstacle avoidance such as discussed in Sect. 4 can be employed for the obstacle avoidance of separate segments of the robot.

6.3 Experimental Results

To show the applicability of the dynamic system for trajectory generation we applied it to two Kuka LWR robots. The task was a bimanual cooperative manipulation while avoiding obstacles. The obstacles in this example were detected using the

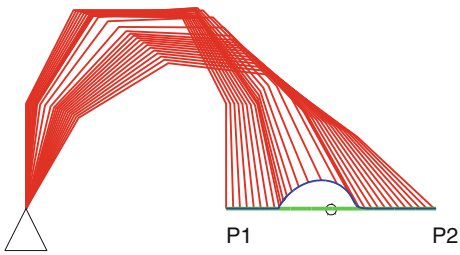

Fig. 17 The obstacle is the *black sphere*, which is directly in the path of the robot, denoted by *green*. When the obstacle-avoidance term is introduced, the robot takes the *blue* trajectory

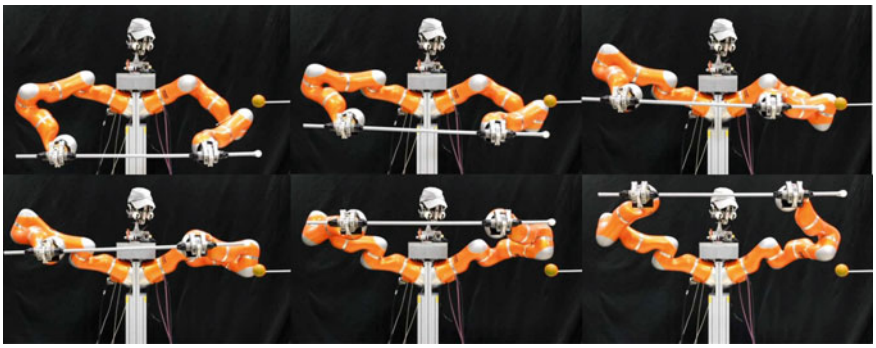

Fig. 18 The image sequence shows a bimanual task, controlled with dynamical systems

stereo-vision cameras. The results are shown in Fig. 19 and as an image sequence in Fig. 18. As we can see one of the arms encounters an obstacle, given by the orange ball, and has to adapt its predefined trajectory (straight line) similar to that shown in the example given in Fig. 17. The control of the other arm is adapted as well in order to maintain a constant distance between them.

7 Conclusion

The presented approaches for on-line obstacle avoidance for redundant manipulators are based on redundancy resolution at the velocity level. For the first presented methods, 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 velocity component in a direction that is away from the obstacle. We have shown that it is 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 pseudo-inverse of the Jacobian matrix $\mathbf{J}_o$ is simpler as it includes a 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.

Next, the control algorithms are presented, where the tasks' priorities can be altered during the execution of the motion. In the context of obstacle avoidance this means that the obstacle can also appear on the desired end-effector trajectory. For changing the priorities of the task we first show how to modify the prioritized task-control algorithm at the velocity level to implement smooth transitions between tasks with different priorities. The higher-priority task will only be active when the desired

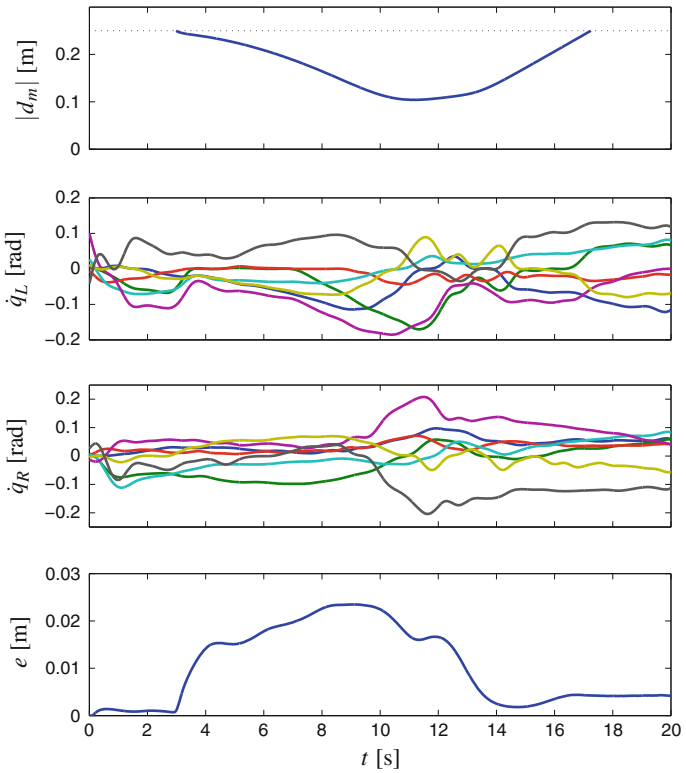

Fig. 19 *Top plot* shows the distance between the obstacle and the nearest link. Note that obstacle avoidance is active only under 0.25 m and that it only acts while the velocity is towards the obstacle. Once the robot is past the obstacle, the perturbation-rejection properties of DMPs ensure smooth return to the original trajectory. *Second* and *third plot* shows the joint velocities for left and right robot respectively, which are continuous and derivable. *Bottom plot* shows the task error

criterion is met and otherwise the higher-priority task is smoothly deactivated. This characteristic to separate tasks and to activate them only when necessary, improves the performance of the robot significantly. Furthermore, the presented method does this activation/deactivation of tasks in a smooth way. We also explain how to find the necessary motion of the robot for all the tasks simultaneously using the extended Jacobian. As such an approach does not always give a feasible solution we propose to use a priority weighted damped least-squares Jacobian for arranging the tasks by priority. In this way the best solution can be found for the particular situation. With some examples we show how the priority-based damping factors influence the motion generation for particular tasks. With a proper choice of these factors it is possible to get such joint velocities which ensure the desired behavior in the best possible way.

Finally, we show how a dynamical system for trajectory generation can be modified to be suitable for online control. Since the dynamical system can only avoid obstacles that appear in the trajectory path, it is necessary to use a control method that can modify the robot null-space configuration if needed. The combination of both dynamical systems for trajectory generation and control with obstacle avoidance is a powerful framework that can easily be used in different applications.

References

1. Chiaverini S (1997) Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Trans Robot Autom 13(3):398–410. doi:10.1109/70.585902
2. Egeland O (1987) Task-space tracking with redundant manipulators. IEEE J Robot Autom 3(5):471–475. doi:10.1109/JRA.1987.1087118
3. Lenarcic J, Stanisic M (2003) A humanoid shoulder complex and the humeral pointing kinematics. IEEE Trans Robot Autom 19(3):499–506
4. Nakamura Y, Hanafusa H, Yoshikawa T (1987) Task-priority based redundancy control of robot manipulators. Int J Robot Res 6(2):3–15
5. Kuffner JJ, Lavalle SM (2000) RRT-connect: an efficient approach to single-query path planning, April, pp 995–1001
6. Lozano-Perez T (1983) Spatial planning: a configuration space approach. IEEE Trans Comput 100(2):108–120
7. Lozano-Pérez T, Wesley MA (1979) An algorithm for planning collision-free paths among polyhedral obstacles. Commun ACM 22:560–570
8. Burns B (2005) Toward optimal configuration space sampling. In: Proceedings of robotics: science and systems, pp 1–6
9. Diankov R, Kuffner J (2007) Randomized statistical path planning. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 1–6. doi:10.1109/IROS.2007.4399557. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4399557
10. Toussaint M (2009) Robot trajectory optimization using approximate inference. In: Proceedings of the 26th annual international conference on machine learning—ICML'09. ACM Press, New York, pp 1049–1056. doi:10.1145/1553374.1553508. http://portal.acm.org/citation.cfm?doid=1553374.1553508
11. Brock O, Khatib O, Viji S (2002) Task-consistent obstacle avoidance and motion behavior for mobile manipulation. In: Proceedings of IEEE international conference on robotics and automation, ICRA'02, vol 1, pp 388–393. doi:10.1109/ROBOT.2002.1013391
12. Colbaugh R, Seraji H, Glass K (1989) Obstacle avoidance for redundant robots using configuration control. J Robot Syst 6(6):721–744
13. Glass K, Colbaugh R, Lim D, Seraji H (1995) Real-time collision avoidance for redundant manipulators. IEEE Trans Robot Autom 11(3):448–457
14. Guo Z, Hsia T (1993) Joint trajectory generation for redundant robots in an environment with obstacles. J Robot Syst 10(2):199–215
15. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Robot Res 5(1):90–98. doi:10.1177/027836498600500106
16. Kim JO, Khosla PK (1992) Real-time obstacle avoidance using harmonic potential functions. IEEE Trans Robot Autom 8(3):338–349
17. Maciejewski AA, Klein CA (1985) Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. Int J Robot Res 4(3):109–117. doi:10.1177/027836498500400308

18. McLean A, Cameron S (1996) The virtual springs method: path planning and collision avoidance for redundant manipulators. Int J Robot Res 15(4):300–319
19. Seraji H, Bon B (1999) Real-time collision avoidance for position-controlled manipulators. IEEE Trans Robot Autom 15(4):670–677
20. Volpe R, Khosla P (1993) A theoretical and experimental investigation of impact control for manipulators. Int J Robot Res 12(4):351–365
21. Feder HJS, Slotine JJE (1997) Real-time path planning using harmonic potentials in dynamic environments. In: Proceedings of IEEE international conference on robotics and automation, April, pp 874–881
22. Iossifidis I, Sch G (2006) Dynamical systems approach for the autonomous avoidance of obstacles and joint-limits for an redundant robot arm. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems, pp 580–585
23. Khansari-Zadeh SM, Billard A (2012) A dynamical system approach to realtime obstacle avoidance. Auton Robot 32(4):433–454. doi:10.1007/s10514-012-9287-y. http://link.springer.com/10.1007/s10514-012-9287-y
24. Park DHPDH, Hoffmann H, Pastor P, Schaal S (2008) Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: Proceedings of 8th IEEE-RAS international conference on humanoid robots, humanoids 2008. IEEE, vol 121, pp 91–98. doi:10.1109/ICHR.2008.4755937. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4755937
25. Sprunk C, Lau B, Pfaff P (2011) Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In: Proceedings of IEEE international conference on robotics and automation, pp 72–77
26. Newman WS (1989) Automatic obstacle avoidance at high speeds via reflex control. In: Proceedings of IEEE international conference on robotics and automation. IEEE, pp 1104–1109
27. Xie F, Qu Z, Garfinkel A (1998) Dynamics of reentry around a circular obstacle in cardiac tissue. Phys Rev E 58(5):6355
28. O'Neil K (2002) Divergence of linear acceleration-based redundancy resolution schemes. IEEE Trans Robot Autom 18(4):625–631. doi:10.1109/TRA.2002.801046
29. Khatib O (1987) A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE J Robot Autom 3(1):43–53. doi:10.1109/JRA.1987.1087068
30. Mansard N, Khatib O, Kheddar A (2009) A unified approach to integrate unilateral constraints in the stack of tasks. IEEE Trans Robot 25(3):670–685. doi:10.1109/TRO.2009.2020345
31. Sentis L, Park J, Khatib O (2010) Compliant control of multicontact and center-of-mass behaviors in humanoid robots. IEEE Trans Robot 26(3):483–501. doi:10.1109/TRO.2010.2043757
32. Stasse O, Escande A, Mansard N, Miossec S, Evrard P, Kheddar A (2008) Real-time (self)-collision avoidance task on a HRP-2 humanoid robot. In: IEEE international conference on robotics and automation, pp 3200–3205
33. Žlajpah L, Petrič T (2012) Serial and parallel robot manipulators—kinematics, dynamics, control and optimization, chap obstacle avoidance for redundant manipulators as control problem. InTech, pp 203–230
34. Sciavicco L, Siciliano B (2005) Modelling and control of robot manipulators, 2nd edn., Advanced textbooks in control and signal processingSpringer, London
35. Petrič T, Žlajpah L (2013) Smooth continuous transition between tasks on a kinematic control level: obstacle avoidance as a control problem. Robot Auton Syst 61(9):948–959
36. Petrič T, Gams A, Babič J, Žlajpah L (2013) Reflexive stability control framework for humanoid robots. Auton Robot 34(4):347–361. doi:10.1007/s10514-013-9329-0
37. Petrič T, Žlajpah L (2011) Smooth transition between tasks on a kinematic control level: application to self collision avoidance for two kuka lwr robots. In: 2011 IEEE international conference on robotics and biomimetics, pp 162–167
38. Sugiura H, Gienger M, Janssen H, Goerick C (2007) Real-time collision avoidance with whole body motion control for humanoid robots. In: IEEE/RSJ international conference on intelligent robots and systems, IROS 2007, pp 2053–2058. doi:10.1109/IROS.2007.4399062

39. Žlajpah L, Nemec B (2002) Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In: IEEE/RSJ international conference on intelligent robots and systems, vol 2, pp 1898–1903. doi:10.1109/IRDS.2002.1044033
40. Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of two 3-d point sets. IEEE Trans Pattern Anal Mach Intell 5:698–700
41. Khatib O, Brock O, Chang KS, Ruspini D, Sentis L, Viji S (2004) Human-centered robotics and interactive haptic simulation. Int J Robot Res 23(2):167–178
42. Konietschke R, Hirzinger G (2009) Inverse kinematics with closed form solutions for highly redundant robotic systems. In: Proceedings of IEEE international conference on robotics and automation. IEEE, pp 2945–2950
43. Santis AD, Siciliano B (2008) Inverse kinematics of robot manipulators with multiple moving control points. In: Lenarčič J, Wenger P (eds) Advances in robot kinematics: analysis and design. Springer, New York, pp 429–438
44. Antonelli G (2009) Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. IEEE Trans Robot 25(5):985–994. doi:10.1109/TRO.2009.2017135
45. Chiaverini S, Oriolo G, Walker ID (2008) Kinematically redundant manipulators. In: Siciliano B, Khatib O (eds) Springer handbook of robotics, chap 11. Springer, Berlin, pp 245–268
46. Park J, Choi YJ, Chung WK, Youm Y (2001) Multiple tasks kinematics using weighted pseudo-inverse for kinematically redundant manipulators. In: Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. No.01CH37164), vol 4. IEEE, pp 4041–4047
47. Baerlocher P, Boulic R (1998) Task-priority formulations for the kinematic control of highly redundant articulated structures. In: Proceedings of IIEEE/RSJ international conference on intelligent robots and systems, vol 1, October, pp 323–329
48. Chiaverini S, Siciliano B, Egeland O (1994) Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. IEEE Trans Control Syst Technol 2(2):123–134
49. Sciavicco L, Siciliano B (1986) Solving the inverse kinematic problem for robotic manipulators. In: Morecki A, Bianchi G, Kdzior K (eds) Proceedings of the 6th CISM-IFToMM symposium on theory and practice of robots and manipulators. Springer, Krakow, pp 107–114
50. Egeland O, Sagli J, Spangelo I, Chiaverini S (1991) A damped least-squares solution to redundancy resolution. In: Proceedings 1991 IEEE international conference on robotics and automation. IEEE Computer Society Press, pp 945–950
51. Buss SR, Kim JS (2004) Selectively damped least squares for inverse kinematics. J Graph Tools 10:37–49
52. Deo A, Walker I (1992) Robot subtask performance with singularity robustness using optimal damped least-squares. In: Proceedings 1992 IEEE international conference on robotics and automation. IEEE Computer Society Press, pp 434–441
53. Maciejewski A, Klein C (1988) Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. J Robot Syst 5(6):527–552
54. Nakamura Y, Hanafusa H (1986) Inverse kinematics solutions with singularity robustness for robot manipulator control. Trans ASME J Dyn Syst Meas Control 108(3):163–171
55. Žlajpah L (2013) Multi-task control for redundant robots using prioritized damped least-squares inverse kinematics. In: 22nd international workshop on robotics in Alpe-Adria-Danube region, Portorož, Slovenia, 11–13 September 2013
56. Likar N, Nemec B, Žlajpah L (2012) Virtual mechanism approach for dual-arm manipulation. Robotica 1:1–16
57. Ijspeert A, Nakanishi J, Pastor P, Hoffmann H, Schaal S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. Neural Comput 25(2):328–373
58. Ijspeert A, Nakanishi J, Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: IEEE international conference on robotics and automation (ICRA), vol 2. Washington, DC, pp 1398–1403

59. Gams A, Ijspeert AJ, Schaal S, Lenarčič J (2009) On-line learning and modulation of periodic movements with nonlinear dynamical systems. Auton Robot 27(1):3–23
60. Ijspeert AJ, Nakanishi J, Schaal S (2002) Learning rhythmic movements by demonstration using nonlinear oscillators. In: Proceedings of IEEE/RSJ international conference intelligent robots and systems. Lausanne, pp 958–963
61. Petrič T, Gams A, Ijspeert AJ, Žlajpah L (2011) On-line frequency adaptation and movement imitation for rhythmic robotic tasks. Int J Robot Res 30(14):1775–1788
62. Ude A, Gams A, Asfour T, Morimoto J (2010) Task-specific generalization of discrete and periodic dynamic movement primitives. IEEE Trans Robot 26(5):800–815