Smooth continuous transition between tasks on a kinematic control level: Obstacle avoidance as a control problem

Tadej Petrič *, Leon Žlajpah
Department of Automation, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

HIGHLIGHTS

• A new simple formulation for the null-space projection is proposed.
• Undisturbed motion of the secondary task if the primary task is not active.
• Avoiding obstacles without changing initial trajectory.
• Smooth transition between primary and secondary tasks.

ARTICLE INFO

Article history:
Received 6 September 2012
Received in revised form
19 April 2013
Accepted 22 April 2013
Available online 16 May 2013

Keywords:
Obstacle avoidance
Redundant robots
Velocity control

ABSTRACT

Kinematically redundant robots allow simultaneous execution of several tasks with different priorities. Beside the main task, obstacle avoidance is one commonly used subtask. The ability to avoid obstacles is especially important when the robot is working in a human environment. In this paper, we propose a novel control method for kinematically redundant robots, where we focus on a smooth, continuous transition between different tasks. The method is based on a new and very simple null-space formulation. Sufficient conditions for the tasks design are given using the Lyapunov-based stability discussion. The effectiveness of the proposed control method is demonstrated by simulation and on a real robot. Pros and cons of the proposed method and the comparison with other control methods are also discussed.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

The kinematic redundancy is one of the important properties of the new generation of service and humanoid robots [1,2]. For example, most robot arms mounted on a mobile platform, as well as humanoid robots, are kinematically redundant. The kinematic redundancy is characterized by additional degrees-of-freedom (DOFs) with respect to DOF required by a given task [3]. Due to the redundant DOF the robot has the ability to move the end-effector along the same trajectory using different configurations of the mechanical structure. This provides means for solving sophisticated motion tasks, such as avoiding obstacles [4, 5], avoiding singularities [6], optimizing manipulability [7], minimizing joint torques [8], etc. As a consequence, the dexterity of the system is significantly increased. This is important to accomplish complex tasks. Additionally, the redundancy can also have an important influence on the dynamic behavior of the robotic system.

For redundant systems three main groups of control schemes have been proposed: velocity-, acceleration- and torque-based. All of them allow execution of two or more prioritized tasks. Usually, the secondary task is executed in the null-space of the primary task [9–11,1]. Khatib deeply investigated the use of the second-order inverse kinematics on torque and acceleration level, starting from robot manipulators [12] to recent task-prioritized humanoid applications [13,14]. It has been established that certain acceleration-based control schemes exhibit instabilities, i.e., instantaneous torque minimization redundancy-resolution schemes as proposed in [15,16], which were further mathematically analyzed by O’Neil [17]. An alternative approach is to use the augmented Jacobian, as introduced in [18], where the secondary task is added to the primary task to obtain a square Jacobian matrix that can be inverted. The main drawbacks of this technique are the so-called algorithmic singularities. They occur when the secondary task causes a conflict with the primary task. The basic algorithm for the velocity-based control scheme was given by Maciejewski et al. [10]. The extension of this algorithm to a large number of tasks was given by Siciliano and Slotine [19] and further improved by Baerlocher and Boulic [20]. An experimental comparison between velocity-, acceleration- and torque-level approaches
was given in [21] and a unified framework for the control of redundant robots was presented in [22].

In the literature possible conflicts between the tasks were seldom analytically investigated or it has been avoided by resorting to conservative constraints. In the task priority context, a singularity robust solution for two tasks was proposed in [23]. However, as noted in [24,25] its generalization to a generic number of tasks is not trivial. Recently, Antonelli [26,2] has discussed a general stability analysis by following the Lyapunov-based approach for the null-space-based behavior control. In his work [2], a sufficient condition that allows the determination of minimum bounds for the control gains of some common task-priority closed-loop inverse kinematic algorithms was given. Moreover, he also gave a simple condition to verify, if the tasks are properly designed.

In general, the task-priority closed-loop inverse kinematic algorithms do not always allow simple transitions between the tasks or changing the task priority [3,27,28]. For example, if the task is to move the end-effector from point A to point B and the obstacle is on the desired path, then the robot must properly adapt its motion to avoid the obstacle and then to move to the point B. This can be done either by changing the desired trajectory [29–31], or by applying algorithms that can change the task priorities. Recently, Mansard et al. [13] have introduced a new inversion operator for computing the appropriate null-spaces. Their approach is generic and could be applied to various sets of tasks, constraints, and robots. However, it leads to very complex formulations which are difficult to use. A similar approach was proposed by Sugiuara et al. [5], where a self collision avoidance system that superposes trajectories was proposed. They used a concept based on virtual forces between close segments of the robot. The final motion uses the avoidance movements blended with a whole body motion control. However, no proof of the stability or the convergence of the proposed control algorithm was provided. In [32] a dynamic hierarchy was proposed, where obstacle avoidance is realized in the null-space of the primary task. If null-space of the primary task is not sufficient for avoiding obstacles, then the obstacle avoidance task is given a higher priority. To smooth the transition, Lee et al. used modified null-space techniques. An approach addressed in a dynamic domain, was proposed by Dietrich et al. [33], where a singular value decomposition was used for shaping a continuous null-space projection.

In this paper, we propose a very simple novel formulation for the null-space projection. The proposed null-space formulation allows undisturbed motion of the secondary task if the primary task is not active. The primary task becomes active only if a selected criterion approaches a predefined threshold. Upon approaching the threshold, the primary task smoothly takes over and the secondary task motion is more and more constrained to the null-space of the primary task.

The stability of the proposed algorithm was analyzed, following the work of [2]. To demonstrate the effectiveness of the proposed algorithm we selected an obstacle avoidance task. The preliminary results of this work were published at IEEE-ROBIO conference [34], where an experiment of a self collision avoidance for two Kuka LWR robots was shown. In this paper, we have extended the obtained results to different simulation scenarios to demonstrate different control system properties. The proposed control method was also compared to other known control methods, which were used in [1].

The rest of the paper is organized as follows: In Section 2 we propose the algorithm for continuous transition between the tasks. In Section 3 the stability of the proposed control algorithm is discussed. In Section 4 we evaluate the proposed approach by using it for the obstacle avoidance on different robots. The obtained results are then compared with the previous published results. The discussion is given in Section 5. Conclusions and the summary are given in Section 6.

2. Kinematics

The robotic systems under study are serial manipulators. We consider only redundant systems, which have more DOF than needed to accomplish the desired task, i.e., the dimension of the joint space \(n\) exceeds the dimension of the task space \(m\), \(n > m\). The difference between \(n\) and \(m\) will be denoted as the degree-of-redundancy (DOR). Note that by this definition the redundancy is not only a characteristic of the manipulator itself but also of the task, meaning that for a certain task a non-redundant manipulator may also become redundant. By defining \(\mathbf{q}\) as an \(n\)-dimensional vector of joint positions and \(\mathbf{x}\) as an \(m\)-dimensional end-effector positions, then the following equation describes the direct kinematic of the manipulator:

\[
\mathbf{x} = f(\mathbf{q}).
\]

where \(f\) is an \(m\)-dimensional vector function, which maps the joint-space variables into the task space variables. The corresponding relationship between joint and end-effector velocities is given by the following well-known expression

\[
\dot{\mathbf{x}} = \frac{\partial f}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J} \dot{\mathbf{q}}.
\]

where \(\mathbf{J}\) is the \(m \times n\) Jacobian matrix. The solution of the above equation for \(\dot{\mathbf{q}}\) by pursue minimal-norm velocity leads to the least squares solution

\[
\dot{\mathbf{q}} = \mathbf{J}^{\dagger} \mathbf{x} + N \dot{\mathbf{q}}_r.
\]

Here subscript \(u\) denotes the commanded control input and \(\mathbf{J}^{\dagger}\) is a Moore–Penrose pseudo-inverse of \(\mathbf{J}\) using the identity matrix as the weight

\[
\mathbf{J}^{\dagger} = \mathbf{J}^{\top} (\mathbf{J}^{\top} \mathbf{J})^{-1},
\]

and \(N\) is the \(n \times n\) matrix representing the projection into the null-space of \(\mathbf{J}\)

\[
\mathbf{N} = \mathbf{I} - \mathbf{J}^{\dagger} \mathbf{J}.
\]

where \(\mathbf{I}\) is the \(n \times n\) identity matrix and \(\mathbf{q}_r\) is an arbitrary \(n\)-dimensional vector of joint velocities. This formulation is well established in close-loop kinematic control algorithms. To obtain a dynamically consistent null-space projection the mass matrix must be used in the pseudo-inverse of \(\mathbf{J}\); for more details see [12].

A redundant manipulator can execute more tasks simultaneously but they should be arranged by priority. For the sake of simplicity, we consider two tasks, \(T_a\) and \(T_b\), which are defined as

\[
x_a = f_a(\mathbf{q}),
\]

\[
x_b = f_b(\mathbf{q}),
\]

and we assume that \(T_a\) is the primary task. The corresponding Jacobian matrices are \(\mathbf{J}_a\) and \(\mathbf{J}_b\), and their null-space projections \(\mathbf{N}_a\) and \(\mathbf{N}_b\). Using the task prioritized formulation in (3) yields

\[
\dot{\mathbf{q}}_a = \mathbf{J}_a^{\dagger} \mathbf{x}_a + \mathbf{N}_a \dot{\mathbf{q}}_a\quad \text{and}\quad \dot{\mathbf{q}}_b = \mathbf{J}_b^{\dagger} \mathbf{x}_b + \mathbf{N}_b \dot{\mathbf{q}}_b.
\]

When more than two tasks should be executed then a possible generalization of the singularity-robust task priority inverse kinematic solution as proposed in [19,23] leads to

\[
\dot{\mathbf{q}}_i = \mathbf{J}_i^{\dagger} \mathbf{x}_i + \sum_{i=b}^{k} \mathbf{N}_j \dot{\mathbf{q}}_j, \quad \mathbf{N}_i = \mathbf{N}_a \mathbf{N}_b \ldots \mathbf{N}_{i-1}
\]

where \(k\) denotes the task with the lowest priority in the task sequence \(T_a, T_b, \ldots, T_k\).

The velocities associated with the lower-priority task are projected into the null-space of the next higher priority task. This formulation commonly realized in most whole-body control
schemes and it is also known as a successive projection method. Compared to other augmented projection methods, it is easier to implement it, it has less problems with singularities in the redundancy resolution and it has lower computational complexity. However, the successive projection method does not ensure the compulsorily strict compliance with the priority order compared to the augmented projection method. The detailed comparison between the successive and the augmented projection method is given in [2].

In many cases it would be of benefit to have the possibility to change the priority of particular subtasks. Using the formulation (9) this cannot be done in a smooth way. Therefore, we propose a new control algorithm, where the velocity \( \dot{q} \) is defined as

\[
\dot{q} = J^\top_e \dot{x}_{a,e} + \sum_{i=b}^{a} \hat{N}_i J^\top_e \dot{x}_{a,e}, \quad \hat{N}_i = N_i N_{i+1} \ldots N_{a-1} \tag{10}
\]

where the matrix \( N_i \) is given by

\[
N_i = I - \lambda(T_i) J_i J_i^\top. \tag{11}
\]

Here \( \lambda(T_i) \) is a scalar measure of how “active” is the task \( T_i \), \( \lambda(T_i) \in [0, 1] \). Note that the variable \( T_i \) stands for a general task. When the task \( T_i \) is active, i.e., some motion is required by the task \( T_i \) then \( \lambda(T_i) = 1 \). When the task \( T_i \) is not active, then \( \lambda(T_i) = 0 \). In the transition phase when \( T_i \) is becoming active or when \( T_i \) needs no more actions, \( \lambda(T_i) \) has a value between 0 and 1. Consequently, if the primary task is denoted with \( T_a \) and secondary task with \( T_b \), then when \( \lambda(T_a) = 1, N = N \) and the motion for \( T_b \) is bounded to the null-space of the \( T_a \). When \( \lambda(T_a) = 0, N = I \) and motion \( T_b \) is not constrained by \( T_a \). Note that if tasks \( T_a \) and \( T_b \) are in the conflict then the task \( T_b \) will not be executed completely.

In general, scalar measure \( \lambda(T_i) \) used in (11) can be any continuous function of the task \( T_i \) with monotonic first derivative defined on the interval \([0, 1]\). Furthermore, its selection corresponds to the particular task. For example, for obstacle avoidance tasks Dietrich et al. [33] proposed a third order polynomial defined on an interval between 0 and 1. Another approach is given in [35], where task activation weights are calculated based on predefined task dynamics. In [36] an approach based on the exponential function for activating the stability task was proposed. A similar approach is used later in this paper in Section 4.

3. Stability analysis

To analyze the stability of the proposed control algorithm (10) we use the Lyapunov method similar as the stability analysis was done for some of the task-priority closed-loop inverse kinematics algorithms in [2,26].

First we have to check the relation between the task \( T_a \) and the task \( T_b \). Tasks \( T_a \) and \( T_b \) are independent if

\[
\text{rank}(J_e^\top J_e) + \text{rank}(J_b^\top J_b) = \text{rank}(J_e^\top J_b^\top J_e), \tag{12}
\]

and they are dependent if

\[
\text{rank}(J_e^\top J_e) + \text{rank}(J_b^\top J_b) > \text{rank}(J_e^\top J_b^\top J_e). \tag{13}
\]

Next, they are orthogonal if

\[
J_e^\top J_b = O. \tag{14}
\]

where \( O \) is an \( m_a \times m_b \) null matrix. The orthogonal condition implies the independency but not vice versa [2]. These three conditions can also be verified by resorting to the transpose of the corresponding Jacobian instead of the pseudo-inverse, because \( J^\top \) and \( J^\top \) share the same span.

We selected \( x_{a,u} \) in (10) as simple \( P \) controller

\[
\dot{x}_{a,u} = K_a(x_{a,d} - x_a) = K_a e_a, \tag{15}
\]

where \( x_{a,d} \) is the desired end-effector trajectory, \( K_a \) is an \( m \times m \) diagonal positive definite matrix, and \( e_a \) is the position error. Next, by assuming that the desired end-effector velocity is 0, \( \dot{x}_{a,d} = 0 \), yields

\[
\dot{e}_a = -\dot{x}_a = -J_a \dot{q}. \tag{16}
\]

Similar is valid also for task \( T_b \)

\[
\dot{e}_b = -\dot{x}_b = -J_b \dot{q}. \tag{17}
\]

By stacking both task errors as

\[
e = [e_a^T e_b^T]^T, \tag{18}
\]

the (18) results in

\[
\dot{e} = -[e_a e_b] = -[J_a J_b] \dot{q}. \tag{19}
\]

By resorting to the Lyapunov stability theorem, the error \( e \) tends asymptotically to zero, if a strictly positive continuously differentiable function \( V(e) \) is defined whose time derivative \( \dot{V}(e) \) is negative definite. For a possible Lyapunov function candidate we select

\[
V(e) = \frac{1}{2} e^T e. \tag{20}
\]

The time derivative of \( V(e) \) equals

\[
\dot{V} = e^T \dot{e} = -e^T [J_a J_b] \dot{q}. \tag{21}
\]

By considering (10) in (21) yields

\[
\dot{V} = -e^T \begin{bmatrix} J_a & J_b \end{bmatrix} \left( J_a^\top K_a e_a + J_b^\top N_a J_a^\top K_b e_b \right). \tag{22}
\]

and rearranging it yields

\[
\dot{V} = -e^T \begin{bmatrix} J_a & J_b \end{bmatrix} \left[ \begin{bmatrix} J_a J_a^\top K_a & J_a J_b^\top K_b \\ J_b J_b^\top K_a & J_b J_b^\top K_b \end{bmatrix} \right] e. \tag{23}
\]

\[
= -e^T M e. \tag{24}
\]

\[
M = \begin{bmatrix} M_{11} & M_{12} \\
M_{21} & M_{22} \end{bmatrix}. \tag{25}
\]

Here, the matrix \( M \) is decomposed into sub-matrices \( M_{ij} \) of proper dimensions. To apply the Lyapunov stability theorem, it is necessary to investigate the definiteness of \( M \). This can be done by independently investigating the definiteness of sub-matrices \( M_{ij} \).

A necessary condition for \( M \) to be positive definite is that all sub-matrices on the diagonal are positive definite. Remarkably, from (23) and (25) it follows that

\[
M_{11} = J_a J_a^\top K_a = K_a
\]

is positive definite as long as the gain matrix \( K_a \) is a positive definite matrix. Next, if the tasks \( T_a \) and \( T_b \) are orthogonal, i.e. the condition (14) holds, then

\[
M_{22} = J_b J_b^\top K_b = J_b (1 - \lambda(e_a) J_a J_a^\top) J_b^\top K_b
\]

\[
= J_b J_b^\top K_b - \lambda(e_a) J_a J_b^\top J_a J_b^\top K_b = K_b.
\]

Hence, \( M_{22} \) is a positive definite matrix if the gain matrix \( K_b \) is a positive definite matrix.
A sufficient condition for \( M \) to be a positive definite matrix is given by its eventual lower triangular form. Again, if condition (14) holds, \( M_{12} = 0 \), then

\[
M_{12} = J_N [\alpha] J_0 = J_0 (I - [\lambda (e_0)] J_0^T J_0) J_0^T K_0
= J_0^T K_0 - [\lambda (e_0)] J_0^T J_0) J_0^T K_0 = 0.
\]

The sign of the sub-matrix holding the lower triangle is not decisive for the overall identification of \( M \). However, it is worth noticing that if the tasks \( T_0 \) and \( T_\alpha \) are orthogonal, \( M_{12} \) is \( O \), otherwise it is not determined in sign.

Summarizing, if \( T_0 \) and \( T_\alpha \) are orthogonal and matrices \( K_0 \) and \( K_\alpha \) are positive definite then also \( M \) is positive definite and the proposed Lyapunov function \( V(e) \) (20) is a strictly negative definite function. Under given assumptions the control algorithm (10) with proportional controllers (15) and (18), and the null desired velocity \( \dot{x}_0 = 0 \) ensure the stability of the system.

4. Obstacle avoidance task

The proposed algorithm can be implemented as a part of a low level control for different scenarios, e.g., obstacle avoidance, constraining task or joint space, reflexive behavior of humanoid robot for achieving stability, etc. In this paper, we show how to apply the proposed approach for obstacle avoidance.

In general, the obstacle-avoidance consists in identifying the points on the robotic mechanism, which are near obstacles and to assign to them a motion component that moves these points away from the obstacles. Generally, to solve the obstacle-avoidance (or collision-avoidance) problem two classes of strategies can be used: 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. Then, the collision-free path is found in the unoccupied portion of the configuration space [37–39]. 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 the use of global strategies in practice 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 the obstacle avoidance as a control problem. Their aim is not to replace the higher-level, collision-free path planning but to exploit the capabilities of the low-level control, e.g., they can use the sensor information to change the path if the obstacle appears in the workspace or if the obstacle 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.

For this purpose, methods based on different null-space criteria have been proposed [1,11,10]. Seto et al. [37,38] proposed a collision avoidance system for the interaction between humans and the robot using virtual reaction force. Sugiuara et al. [5] proposed a method that blends the joint velocity vector for collision avoidance and the target reaching motions depending on the distance between the closest segment pair. Their method is based on a blending coefficient, which determines how much a particular velocity vector will affect the resulting motion. However, no stability proof or stability conditions were given by them.

In this section, we apply our framework (Section 2) to obstacle avoidance so that the motion of the robot is modified, if at least one part of the robot is within the critical neighborhood of an obstacle, i.e., the distance between the robot and the obstacle is less than the predefined threshold. We denote this obstacle with the term the “active obstacle” and the corresponding closest point on the body of the robot mechanism as the “critical point”, in Fig. 1 marked with \( A_0 \).

In our case we assume that the motion of the end-effector can be disturbed by any obstacle, which was not the case in examples presented in [1]. If such a situation occurs when using any of the approach proposed in [1], usually the task execution has to be interrupted and the 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 successfully use the control scheme as proposed in (10). In this case, the obstacle avoidance is the primary task \( T_0 \) and the end-effector motion is the secondary task \( T_\alpha \). The velocities \( \dot{x}_\alpha \) are defined by the relative position between the robot and the obstacle.

In a situation, when an obstacle is on the end-effector path, the task \( T_\alpha \) becomes active and the robot has to move off the desired path to avoid the collision. For that, the obstacle avoidance requires the motion of the critical point in the direction away from the closest point on the obstacle. In general, this is a one-dimensional constraint and only one DOR is needed. Let \( d_0 \) be the vector connecting the closest points on the obstacle and the robot, and let the operational space in \( A_0 \) be defined as one-dimensional space in the direction \( d_0 \), as in [1]. Then, the Jacobian \( J_0 \), which relates to the joint space velocities \( \dot{q} \) and the velocity in the direction of \( d_0 \), is calculated as

\[
J_0 = n_0^T J_0.
\]

Here, \( n_0 \) is the unit vector in the direction \( d_0 \), given by

\[
n_0 = \frac{d_0}{\|d_0\|}\tag{27}
\]

and \( J_0 \) is the Jacobian in point \( A_0 \) defined in the Cartesian space as

\[
J_0 \dot{q} = \dot{x}_0,
\]

where \( \dot{x}_0 \) is the Cartesian velocity that moves the point \( A_0 \) away from the obstacle (see also Fig. 1). Note that the dimension of matrix \( J_0 \) is \( 1 \times n \) and only one DOR is required for the obstacle avoidance.

Eq. (27) only considers one active obstacle acting on the robot. For the derivation of Jacobian \( J_0 \) where arbitrary number of obstacles is considered please see [40]. Note that the dimension...
of the Jacobian matrix $J_0$ is still $1 \times n$ if the approximate solution as defined in [1] is used.

An important issue in the control of redundant robotic manipulators is singular configurations where the associated Jacobian matrix loses its rank. Usually, only the configuration of the whole manipulator is of interest, but in obstacle avoidance we have to consider also the singularities of the substructures defined by the critical point $A_0$, i.e., the part from the base to the point $A_0$ and the part from $A_0$ to the end-effector. Although we can assume that the Jacobian $J$ associated with the desired path is not singular, this is not always true for the Jacobian $J_0$. As the robot is supposed to move in an unstructured environment, it is impossible to know in advance when $J_0$ will become singular. Therefore, a very important advantage of the Jacobian $J_0$ compared to $J_0$ is that $J_0$ has significantly less singularities (for details see [1]).

As already mentioned, the primary task $T_p$ is associated with the obstacle-avoidance motion, i.e., the motion in the direction of $d_0$, and the task $T_b$ is associated with the end-effector motion. This yields

$$
J_0 = J_0,
$$

$$
J_b = J.
$$

Next, let the avoiding velocity $\dot{x}_d$ to be defined as

$$
\dot{x}_d = \lambda(T_0) n_0^T \dot{x}_0 = \lambda(T_0) v_0,
$$

where $v_0$ is scalar representing the nominal avoiding velocity. If only one obstacle is considered $\lambda(T_0)$ is defined as

$$
\lambda(T_0) = \left\{ \begin{array}{ll}
\frac{d_m}{\|d_0\|} & \|d_0\| \geq d_m \\
1 & \|d_0\| < d_m
\end{array} \right.
$$

where $n$ is positive number, $n = 1, 2, 3, \ldots$, and $d_m$ is the critical distance to the obstacle (see also Fig. 2). Note that if there are no obstacles present, $\|d_0\| = \infty$ and $\lambda(T_0) = 0$. If not stated otherwise we use in the following $n = 7$. Now, (10) can be rewritten in the form

$$
\dot{q}_{IP} = J_0^T \dot{x}_d + N_0^T \dot{x}_c,
$$

where subscript IP denotes the controller where the task priority is changing, $\dot{x}_c$ is the task controller for the end-effector tracking and matrix $N_0$ is defined as

$$
N_0 = I - \lambda(T_0) J_0^T J_0.
$$

Formulation (33) allows unconstrained joint movements required by the task $T_b$, if $\lambda(T_0)$ is close to zero $\lambda(T_0) \approx 0$. Thus, while no obstacle is on the path the robot end-effector is able to track the desired path undisturbed. Namely, using (31) and (34) in (33) and considering that in this case $\lambda(T_0) \approx 0$ gives

$$
\dot{q}_{IP} = J_0^T (T_0) v_0 + \left( I - \lambda(T_0) J_0^T J_0 \right) \dot{x}_c = J \dot{x}_c.
$$

On the other hand, when the robot is close to an obstacle ($\lambda(T_0) \approx 1$), the null space matrix $N_0$ in (34) becomes $N_0 \approx N_0 = I - J_0^T J_0$, and the motion of the task $T_b$ is possible only in the null space of the task $T_b$, i.e. not disturbing the avoiding motion. In this case, we can still move the end effector, but the tracking performance is decreased due to the obstacle avoidance. As in this case $\lambda(T_0) \approx 1$, the (33) now derives in

$$
\dot{q}_{IP} = J_0^T v_0 + \left( I - J_0^T J_0 \right) \dot{x}_c.
$$

A special attention has to be given to the selection of the nominal avoiding velocity $v_0$. Large values of $v_0$ would cause unnecessary high velocities and consequently move the manipulator far away from the obstacle. If there are many obstacles in the neighborhood of the manipulator, such a motion may cause problems. Namely the manipulator may bounce between the obstacles [1]. On the contrary, too small values of $v_0$ would not move the critical point of the manipulator far enough away from the obstacle.

The proposed algorithm considers the obstacle avoidance problem at the kinematic level and we compared it with the algorithms given in [1]. For better understanding, a short recap of the approaches in [1] is given. In [1] the authors denote the term “exact solution (EX)” with the movement defined by

$$
\dot{q}_{EX} = J \dot{x}_c + c_0 (J_0 n_0^T (\dot{x}_0 - J_0 \dot{x}_c)),
$$

where $c_0$ is the obstacle avoidance gain and $\dot{x}_c$ is the task velocity. In this case, the closest point to the obstacle on the robot $A_0$ is assigned to move in the direction of the vector $d_0$. In the next case, denoted with the term “approximate solution (AP)”, the closest point on the robot $A_0$ is assigned just to move away from the obstacle, and it is not necessary to move exactly in the direction of the vector $d_0$. Here, the velocity $\dot{q}$ is given by

$$
\dot{q}_{AP} = J \dot{x}_c + N_0^T \dot{x}_d.
$$

Detailed explanations and derivations of these approaches are given in [1].

### 4.1. Simulation examples

The following simulation examples illustrate the behavior of an n-R planar manipulator when it is moving in an unstructured environment with an obstacle. The position of the obstacle is not known in advance. The simulations were done in Matlab/Simulink using the Planar Manipulator Toolbox [39,41]. The proposed algorithm (TP) based on (33) is compared with two kinematic controllers based on strategies introduced in [1]: the exact velocity controller (EX), defined with (37), and the approximate velocity controller (AP), defined with (38). The main difference between EX and AP approaches and the TP approach is that in case of EX and AP the primary task is the path-tracking and the obstacle-avoidance is the secondary task. On the other hand, in the case of TP the priority of the tasks is opposite:

**EX, AP:** $T_p \rightarrow \dot{x}_c$, $T_b \rightarrow \dot{x}_0$  
**TP:** $T_p \rightarrow \dot{x}_b$, $T_b \rightarrow \dot{x}_c$.

If not stated otherwise, EX, AP and TP approaches have the same task space controller given by

$$
\dot{x}_c = K(x_d - x).
$$
where $K$ is the proportional gain ($K = 10$). To avoid the obstacles the critical distance has been selected as $d_m = 0.15$ and nominal avoiding velocity as $v_0 = 1$ for EX, AP and TP.

In the following, three different scenarios are shown: avoiding obstacles if the desired task space path is free, avoiding obstacle if the obstacle is in the desired path, and moving into a narrow labyrinth. The term task space path denotes the path of the end-effector in Cartesian space.

Figs. 3 and 4 show the simulation results of the obstacle avoidance if the obstacle is not on the task space path. In Fig. 3 we can see that the behavior of all three controllers is similar. However, by observing the top plot in Fig. 4, where the minimal distance $d_0$ between the robot and the obstacle is shown, we can see that for the EX and AP methods, the robot comes closer to the obstacle compared to the TP method.

The bottom plot in Fig. 4, shows that the task space tracking error $e_2$ defined as a square difference between the desired robot position and the actual robot position $e_2 = (x_d - x)^2$. As we can see, $e_2$ is not constant when the TP method is used. This is because the effective position gain depends on $N'$ and is changing during the motion. On the other hand, the tracking error in the EX and AP cases is constant as the effective position gain for EX and AP is constant during the motion.

The next scenario comprehends obstacle avoidance if the obstacle is on the task-space path. As expected, methods EX and AP are not able to avoid the obstacle. Hence, a higher-level planer must be employed. Using method TP the robot can avoid the obstacle as shown in Fig. 5. Moreover, the task for obstacle avoidance smoothly takes over when necessary.

The top plot in Fig. 6 shows the minimal distance to the obstacle. We can see that, when using the TP approach, the robot is able to maintain minimal distance to the obstacle defined by $d_m = 0.15$. The bottom plot in Fig. 6 shows the tracking error of the desired task position. We can see that the tracking error is significant when the robot is avoiding the obstacle. This means that the proposed approach can only be used if tracking accuracy of the end-effector motion is not so important examples of such tasks is a reaching task, where the end-effector has to move from the point A to the point B and it is not so crucial if the robot is not always exactly on the prescribed path.

Fig. 7 shows the activation factor $\lambda(T_a)$, the joint velocities $\dot{q}_0$ for the primary task (obstacle avoidance) and the joint velocities $\dot{q}_c$ for the secondary task (tracking desired end-effector path). We can see that the velocities $\dot{q}_0$ for the obstacle avoidance shown in middle plot, appear when robot approaches the predefined threshold for the obstacle distance and thus “activates” the factor $\lambda(T_a)$, given with (32). We can also see, that velocities $\dot{q}_0$ are all time smooth and consistent. This is also true for the velocities $\dot{q}_c$ for the end-effector tracking task. Even during the transition phase ($0 \ll \lambda(T_a) \ll 1$), where the velocities are more and more constrained by the null-space projection of the primary task, we can observe a smooth motion.
**Fig. 6.** The top plot shows the closest distance between the robot and the obstacle and the bottom plot shows the task space tracking error using the TP approach. Note that the dotted line in the top plot shows the minimal distance $d_m = 0.15$.

**Fig. 7.** The top plot shows the activation $\lambda(T_a)$, the middle plot shows the joint velocities $\dot{q}_0 = J_0^T \dot{x}_0$ for the primary task (obstacle avoidance) and the bottom plot shows the joint velocities $\dot{q}_c = N_0' J_0^T \dot{x}_c$ for the secondary task (tracking desired task space position).

From Fig. 7 we can see that when the primary task becomes active $\dot{q}_0$, the controller for secondary task $\dot{q}_c$ tries to compensate the tracking error (the obstacle is in the desired end-effector path) but as the motion is mapped into the null-space of the primary task, the necessary compensating motion cannot be realized completely.

In the next example, the task is to move into a narrow labyrinth. The desired end-effector path is in the middle of the corridor and there are no obstacles on the desired path. To perform the task a robot with 10 DOF was used. To analyze the behavior in a narrow free space, the distances between the walls were less than the manipulator link lengths and the critical distance was $d_m = 0.1$.

**Fig. 8.** Traces for obstacle avoidance when moving into a small labyrinth using different approaches, i.e. EX, AP, TP and TP + AP, from left to right hand-side respectively. The green line is the initial configuration, the red line is the final configuration, the black solid line is the desired end-effector trajectory and the black dotted line is the actual end-effector trajectory. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Fig. 9.** Comparison of obstacle avoidance for moving into a small labyrinth. The top plot shows the closest distance between the robot and the obstacle and the bottom plot shows the task space tracking error.

In the next example, the task is to move into a narrow labyrinth. The desired end-effector path is in the middle of the corridor and there are no obstacles on the desired path. To perform the task a robot with 10 DOF was used. To analyze the behavior in a narrow free space, the distances between the walls were less than the manipulator link lengths and the critical distance was $d_m = 0.1$.

Figs. 8 and 9 show the simulation results. In the case of the EX method all critical points arising simultaneously are considered with smoothly changing weights as in [1]. As mentioned previously, the AP approach is a simplification of the EX approach. Although the direction of the avoiding motion is not optimal due to the approximate null-space mapping (see [1] for details), the obtained results are satisfactory. Next, we can see that for the TP case the robot is unable to reach the end of specified path. The reason is that the obstacle avoidance task has the highest priority and the robot is always close to the obstacles (the wall). Consequently, the end-effector motion, which has lower priority, is bounded to the null space of the primary task and not enough DOF are available for this motion. Since no optimization is applied, the robot does not know how to correct the pose, in order to track the path. However, if we augment the TP approach with the AP method, the robot is able to get into the labyrinth. The joint velocity $\dot{q}$ for the combined...
Fig. 10. Experimental setup for bimanual movement imitation.

Fig. 11. Closest distance between robots, where the left robot is commanded to track the desired position in Cartesian space, and the right robot is moved by a human. The bottom plot shows the squared tracking error $e_2$ for the left robot. Note that the dotted line in the top plot shows the minimal distance $d_m = 0.15$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The approach, denoted with TP + AP, is given by

$$\dot{q}_{TP+AP} = J_{\dot{x}_0}^{D_A \dot{x}_d} + N_0 \dot{q}_{AP},$$  

and

$$= J_{\dot{x}_0}^{D_A \dot{x}_d} + N_0 \left( J^{x_c} \dot{x}_c + NJ_{\dot{x}_d}^{D_A \dot{x}_d} \right).$$

By comparing (41) to (33), we can see that the difference is that Eq. (33) includes also the null-space of the task Jacobian. This shows that we can apply different optimization algorithms with the proposed approach: to optimize the robot pose, to do the minimal velocity optimization or even to do different tasks in the null-space of the other tasks. The last plot in Fig. 8 shows the behavior of the two robots together with the combined controller (TP + AP) given by (41).

Fig. 9 shows the minimal distance between the robot and the obstacle and the tracking error of the desired task position. We can see that when the TP approach is used, task space error is significant. On the other hand, if the TP approach is augmented with the AP approach as shown in (41), task space error is neglectable. Summarizing, the behavior of TP + AP is similar to the behavior of EX, except that in case of TP + AP the robot would still be able to avoid the obstacle on the desired task space path. However, using TP + AP method only makes sense if the robot with many DOFs is used and if it is likely to collide with many possible obstacles simultaneously.

As presented, the proposed controller TP for the obstacle avoidance task can be successfully embedded into the control level. The control algorithms are fast enough to work in real-time and there is no need to replan the desired task space path if an obstacle appears on the path.

4.2. Experimental results

We applied our TP algorithm to two Kuka LWR robots as shown in Fig. 10. The TP algorithm is used as a low-level control to prevent the self collision, i.e. the collision between the robots’ links.

In the first example, the right robot arm is teleoperated by a human and the left robot arm is supposed to track a predefined path. We show how the left robot smoothly moves away from the right arm when it comes close to the left arm. In the second example, we present how to prevent a self collision when the robot arms are in the master–slave operation mode. In the third example, we show how to imitate the movements shown by a human demonstrator in real-time using the Microsoft Kinect sensor for human arm motion acquisition.

Fig. 11 shows the results of the self collision avoidance for the first experiment where the task for the left robot was to stay at the desired end-effector position in the Cartesian space and the right robot, was guided by the human. In the top plot of Fig. 11 we can see that the closest distance $d_0$ between both robots is never below the desired threshold $d_m = 0.15$. For details on how to calculate closest distance between two Kuka LWR robots see [42]. However, in the bottom plot we can see that because the robots are close to each other, the square tracking error $e^2$ is in some parts significant. This behavior is expected as the left robot arm has to prevent the collision with the right arm even if it cannot preserve the desired end-effector position. The behavior of the robot is also shown in Fig. 12 and in the first part of the attached video Self_Collision_Avoidance_LWR.avi.

The second scenario shows how both robots avoid each other when they operate in the master–slave mode. The results are shown in Fig. 13. In the top plot we can see that the closest distance between the robots was always above the desired threshold. Other plots show the joint angles for the first four joints for both robots, respectively. Comparing them with the top plot, we can see that
Fig. 13. Closest distance between the robots in the top plot when they are in master–slave configuration. Remaining plots show the first four actual joint angles. The dashed line is the master robot and the solid line is the slave robot. The green shaded area indicates when the distance between the robots is $d_0 < 0.3$ m.

the left robot arm (the slave) is able to track the right robot arm (the master) while they are not close to each other. The motion of the robots is also shown in Fig. 14 and in the second part of the attached video Self_Collision_Avoidance_LWR.avi.

In the third example, we show how to imitate the human motion with the robotic system in real-time. The human motion is captured using the Microsoft Kinect sensor. Microsoft Kinect is based on range camera developed by PrimeSense, which interprets 3D scene information from a continuously-projected infrared structured light. By processing the depth image, the PrimeSense API enables the tracking of human arms movements in real-time.

In the experimental results, where both robots arms were controlled by a human demonstrator using the Microsoft Kinect.

Imitating the motion of the human arm requires some basic understanding of the human physiology. The posture of each arm may be described by four angles—three angles in the shoulder joint and one in the elbow. Shoulder joints enable the following motions [43]: arm flexion, arm abduction and external rotation. These angles are calculated from the data obtained with Microsoft Kinect [42].

Fig. 15 shows the experimental results, where both robots arms were controlled by a human demonstrator using the Microsoft Kinect.

Fig. 14. A sequence of still photos with a frame rate of 3 pictures per second shows the movement of two Kuka LWR robots in master–slave configuration.

Fig. 15. Closest distance between two robots in the top plot. The robots are controlled by a human demonstrator in real time, using Microsoft Kinect. Remaining plots show the desired value for first four joint angles (solid line) and actual values (dashed line) for the left robot in the left-hand side and the right robot in the right-hand side.
Kinect sensor. In the top plot, we can see that the distance between the robots is above the selected threshold. Other plots show the desired and the actual angles for the robot joints. The real and the desired angles are close to each other when the robot arms are not close together. When the arms are close, both robots properly adapt their motion to prevent collisions. Fig. 16 shows a sequence of a successful self collision avoidance (see also the last part of the attached video Self_Collision_Avoidance_LWR.avi).

5. Discussion

In Section 3 we prove the stability of the proposed control system if the tasks are orthogonal. When the tasks are not orthogonal the stability cannot be proven for the tasks separately because the motion of one task influences the motion of the other task and vice-versa. However, orthogonal tasks are not usual in robotics but more a rare exceptions. Usually, the tasks are not orthogonal and this may in some sense relativize the stability proof. As the simultaneous execution of tasks, which are not orthogonal, is common in robotics, we show by examples that the proposed control method can successfully cope with non-orthogonal tasks as well. Therefore, we applied the proposed method to the obstacle-avoidance problem where one task is tracking of the end-effector position and the other task is to avoid collisions with obstacles.

To evaluate the performance of the proposed method system we compared it with two common methods for the online obstacle-avoidance: exact method EX and approximate method AP. For both methods (EX and AP), the obstacle avoidance task was the secondary task projected into the null space of the primary tasks which was the tracking the end-effector position. When the proposed method (TP) was used, the primary task was the obstacle-avoidance and the secondary task was tracking the desired trajectory. By changing priority of these two tasks, we ensured that the robot will be able to avoid obstacles even if they appear on the end-effector path (this was clearly shown in Fig. 5). Remarkably, if EX and AP approaches were used in such a case, the robot would collide with the obstacle if no additional control strategy would be used.

However, there are some special cases where EX or AP would clearly outperform the TP approach. One of such examples is when a highly redundant robot with many DOFs is moving among many obstacles which are not on the end-effector path. In this case, it is clear that with EX and AP methods the end-effector will successfully track the desired path since path tracking is the primary task. As in the TP case the tracking is the secondary task and the corresponding motion is projected into the null-space of the obstacle avoidance task, and as the robot is in this particular case always in the proximity of the obstacles, the primary task is always active and constrains the performance of the secondary task. In such a case, a possible solution is to improve the behavior by augmenting the control algorithm. One possibility is presented in the paper where the TP approach was combined with the AP approach (TP + AP case). Of course, the TP method can easily be extended by any other method. By applying additional methods the TP method can outperform the EX and AP methods since it can avoid obstacles if they appear on the end-effector path without using any additional path planers or strategies.

6. Conclusion

In this paper, we have shown how to modify the prioritized task control at the velocity level to implement smooth transitions between tasks with different priorities. Each task needs “some” DOF and when the tasks have different priorities, the lower priority task can use for its motion only the DOFs, which are not necessary for the higher priority task. This fact can reduce the dexterity of the robot significantly. The basic idea of our approach is to “free” some DOFs used by higher priority task when the task is not using them. This is done by activating and deactivating this task. The proposed control method does this in a smooth way. The higher priority task will only be active when the desired criterion is met and otherwise the higher priority task is smoothly deactivated.

The effectiveness of the proposed approach was shown by the simulation of planar robots and by comparison with other control algorithms. We have also shown how to apply the proposed algorithm to two Kuka LWR robot arms to prevent collisions between them. Using the Lyapunov based approach we have shown that the proposed control ensures a stable motion of the robot under given conditions. Sufficient conditions for the asymptotic stability have been provided. The proposed control approach is general and can be applied to any number of arbitrary tasks which satisfy the given condition.

Appendix. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2013.04.019.

References


Tadej Petrič received the B.Sc. degree in automatics from the Faculty of electrical engineering and computer science at the University of Maribor in 2009. His B.Sc. covered modeling and robotic control of underactuated dynamical system. For his work, he received the Bedjané award in 2008. In 2013 he received the Ph.D. degree in robotics from the Faculty of Electrical Engineering at the University of Ljubljana. He performed a part of his doctoral research at the Department of Robotic Systems for Dynamic Control of Legged Humanoid Robots at German Aerospace Center (DLR) in Berlin, Germany. He is currently a Postdoctoral Assistant with the Department of Automatics, Biocybernetics, and Robotics, Jozef Stefan Institute, Ljubljana. His research interests include human-like robotic movement and mostly control of dynamical systems.

Leon Žlajpah received the B.Sc. degree in automatics from the University of Ljubljana, Ljubljana, Slovenia, in 1982. From the same University he received in 1985 the M.Sc. degree and in 1989 the Ph.D. degree in electrical science. Since 1996 he has been a researcher at the “Jozef Stefan” Institute in Ljubljana, working in different fields of robotics. Since 1998 he is a Senior Researcher at the “Jozef Stefan” Institute and since 2005 he is the Head of the Department of Biocybernetics and Robotics at the “Jozef Stefan” Institute.

His fields of interest include advanced control, especially analysis and synthesis of control systems for robots, redundant robots, learning by observation, modeling
and simulation of robot systems, applications of robots, sensor systems, computer-integrated manufacturing and recently control of robots interacting with humans.

In 1982 he received The Prešeren Award for his work Analyses and Synthesis of Control Systems for Industrial Robots With Microcomputers, in 1987 he was awarded together with collaborators by National award for research excellence (Boris Kidrič Fund Award) for research work in the field of robotics, in 1989 he received The Bedjanič Award for his dissertation, in 1990 he received Boris Kidrič award for patents and improvements for a robotized work-cell for cold reforming and in 2002 he received an Iskra Baterije Zmaj Company Award for the project Automation of manufacturing of alkaline batteries.

Dr. Žlajpah is vice-president of Slovenian Society for Simulation and Modelling, and member of Automatics Society of Slovenia.