On Orientation Control of Functional Redundant Robots

Leon Žlajpah

Abstract— **There are tasks that do not require a controlled motion in all spatial directions. These unused degrees-of-freedom (DOF) make the robot functionally redundant. Traditional methods for redundancy resolution developed for intrinsic redundant robots cannot be used directly for resolving functional redundancy. The functional redundancy is in general not reflected as rows in the Jacobian matrix and therefore, the unused DOFs cannot be mapped into the null-space. In the paper we present a novel approach for the orientation related functional redundancy resolution based on the task space rotation. We define a time-variant task space frame so that one or more frame axes are aligned with the unused directions. This allows that the rows of Jacobian matrix corresponding to functional redundant DOFs are excluded. As the standard quaternion control is not appropriate to control orientation of such functional redundant robots, we propose two control algorithms to control one or two orientation directions. Two examples then show how the proposed control algorithms can be implemented and finally, we present the application of the algorithm on a two-arm robot system.**

I. Introduction

To define the pose of an object in the Cartesian space we need three DOFs for the position and three DOFs for the orientation. Most of today's industrial robots are nonredundant and have six degrees-of-freedom (DOF), which allow them to position and orient their end-effector to any pose in the Cartesian space. However, the six DOFs nonredundant kinematic structure reveals in practice very soon its weakness. Limited joint ranges and kinematic singularities of these robots prevent such robots to reach an arbitrary end-effector pose in the whole workspace. Actually, the workspace, where an arbitrary pose can be reached, is rather limited. Therefore, it is reasonable to augment the robot structure with additional DOFs to overcome the mentioned limitations. Redundant DOFs significantly enhance the capabilities of robots by offering better flexibility and versatility. However, we should be aware that the redundancy is inevitably related to the task in the sense that the robot can be considered redundant if it has more DOFs than needed to accomplish the task.

Tasks are usually defined as a motion of the robot end-effector or of some other point on the robot body. The importance of controlling the motion in the task space directly was recognized very early [1], [2] and since then many control schemes for task-space control have been proposed [3], [4], [5], [6], [7]. When the robot is redundant, then there are infinite joint configurations which result in the desired task-space pose of the end-effector. Therefore, a criterion to select one of the possible solutions is required. The simplest approach to solve the kinematic redundancy is to resolve it on the velocity level [1], where the differential kinematic equation is solved by using a kind of generalized inverse, and the projection of an arbitrary vector onto the null-space of the Jacobian is used to exploit the self-motion of the robot. To find the best solution, different local optimization criteria have been used.

Most of the proposed control schemes for tasks requiring/allowing directional dependent behavior of the robot rely on approach, where the task requirements are transformed into the task space, where the robot kinematics and dynamics are defined [2]. However, such an approach is not suitable when the task does not require a controlled motion in all spatial directions. For example, there are tasks like arc welding or spray painting, where not all six Cartesian dimensions are important to accomplish the task. These unused DOFs can then contribute to the degree-of-redundancy (DOR) of the robot and they can be exploited for the self motion. This type of redundancy has been recognized as the *functional redundancy* [8], [9], [10]. To control robots with functional redundancy, Baron [9] proposed to add a virtual joint to the robot and used the usual redundancy resolution with the extended Jacobian. Later Huo and Baron [10] proposed the orthogonal decomposition of the task-space without considering the null-space of the Jacobian matrix. In [11] the use of the extended Jacobian is proposed for tasks where axis-symmetric tools are used (like spray-painting gun) and the orientation of the tool along one axis is not important. For 5 DOF tasks a redundancy resolution algorithm based on sequential quadratic programming is proposed in [12] and based on convex optimization in [13] .

Common to most of the above control resolution schemes is that they deal only with functional redundancy. This paper aims at contributing to this field by presenting efficient orientation control strategies for functional redundant robots. The main purpose is to improve the capabilities of robots using intrinsic and functional redundancy in the same framework. For that we propose to define a time-variant task frame where the functional redundancy is represented as one or more rows of Jacobian matrix. So, it is possible to exclude the rows related to the functional redundancy from the Jacobian matrix and enlarge the dimensionality of the null-space.

II. Modelling

In robotics the task is usually related to the motion of one or more operational points, which can be anywhere on the

This work was supported by EU Horizon 2020 Programme grant 680431, ReconCell, and Slovenian Research Agency grant J2-7360.

Leon Žlajpah is with Dept. of Automation, Biocybernetics and Robotics, Jozef Stefan Institute, Ljubljana, Slovenia, leon.zlajpah@ijs.si

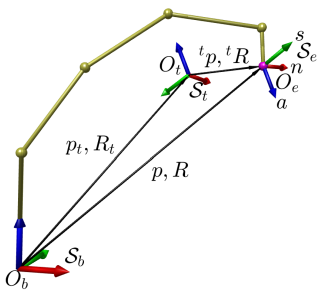

Fig. 1. Representation of task space, world space and corresponding frames

robot structure. Typically they are assigned to the robot end-effectors [2], [5], [6]. Depending on the task, these points are required to move along a specified path, or the robot has to exert a certain force at that point. Without loss of generality we assume in the following that only motion of one operational point has to be controlled.

First we define three spaces:

- the *configuration space* $\mathcal{C}$ is the space where the joint variables $\boldsymbol{q}$ are defined;
- the *operational space* $\mathcal{O}$ is the 6-dimensional Cartesian space, where the positions/orientations of the robot end-effector are defined; and
- the *task space* $\mathcal{T}$ — a Cartesian subspace, where the task is defined, $\mathcal{T} \subseteq \mathcal{O}$.

The pose of the operational point is completely defined by its position and orientation. Let $\mathcal{S}_e$ represents a coordinate frame with the origin O_e attached to the end-effector and aligned with the orientation of the end-effector, and $\mathcal{S}_b$ be a fixed base (or world) coordinate frame with the origin O_b in the base of the robot (see Fig. 1). Then, the location of the frame $\mathcal{S}_e$ in frame $\mathcal{S}_b$ can be represented by a homogeneous matrix

$$\mathbf{T}_e = \begin{bmatrix} \mathbf{R} & \boldsymbol{p} \\ \mathbf{0} & 1 \end{bmatrix} , \tag{1}$$

where $\boldsymbol{p}$ and $\mathbf{R}$ are a 3-dimensional position vector, $\boldsymbol{p} = [p_x, p_y, p_z]^T$, and a 3×3 rotation matrix, $\mathbf{R} = [\boldsymbol{n}, \boldsymbol{s}, \boldsymbol{a}]$, respectively[1]. Note that $\{\boldsymbol{p}, \mathbf{R}\} \in \mathcal{O}$.

We deal with robot manipulators, which can be represented by a serial kinematic chain. Let the configuration of the robot manipulator be represented by the n-dimensional vector $\boldsymbol{q}$ of joint positions, $\boldsymbol{q} \in \mathcal{C}$. Then, the robot end-effector position $\boldsymbol{p}$ and orientation $\mathbf{R}$ can be expressed as a function of joint coordinates using the direct kinematic equations

$$\boldsymbol{p} = \boldsymbol{p}(\boldsymbol{q}) \tag{2}$$
$$\mathbf{R} = \mathbf{R}(\boldsymbol{q}) . \tag{3}$$

In general, the spatial end-effector velocity is expressed as a 6-dimensional vector $\boldsymbol{v}$

$$\boldsymbol{v} = [\dot{\boldsymbol{p}}^T \boldsymbol{\omega}^T]^T , \tag{4}$$

where $\dot{\boldsymbol{p}}$ and $\boldsymbol{\omega}$ are the linear and angular velocity of the end-effector, respectively. They are obtained by differentiating Eqs. (2) and (3). Note that the angular velocity is related to the derivative of $\mathbf{R}$ by the relation [4]

$$\dot{\mathbf{R}} = \mathbf{S}(\boldsymbol{\omega})\mathbf{R} , \tag{5}$$

where $\mathbf{S}(\cdot)$ is a skew-symmetric operator

$$\mathbf{S}(\boldsymbol{a}) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} . \tag{6}$$

The relation between joint velocities $\dot{\boldsymbol{q}}$ and end-effector velocity $\boldsymbol{v}$ is represented by the differential forward kinematics in the form

$$\boldsymbol{v} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}_G(\boldsymbol{q})\dot{\boldsymbol{q}} . \tag{7}$$

where $\mathbf{J}_G(\boldsymbol{q})$ is $6 \times n$ geometric Jacobian matrix [4].

The orientation representation with rotation matrix $\mathbf{R}$ is redundant due to the orthonormality constraints. Using a set of Euler angles $\boldsymbol{\Phi} = [\alpha, \beta, \gamma]^T$ we can obtain a minimal representation of the orientation. Combining elementary rotations about coordinate axes of successive frames $\mathbf{R}_a$, $\mathbf{R}_b$ and $\mathbf{R}_c$, where a, b and c denote one of the 12 possible axis combinations, the orientation matrix $\mathbf{R}$ can be expressed as

$$\mathbf{R}(\boldsymbol{\Phi}) = \mathbf{R}_a(\alpha)\mathbf{R}_b(\beta)\mathbf{R}_c(\gamma) . \tag{8}$$

Now, the end-effector position and orinetation can be represented by the vector

$$\boldsymbol{x}_A = [p_x, p_y, p_z, \alpha, \beta, \gamma]^T \tag{9}$$

and the direct kinematics can be given in the form

$$\boldsymbol{x}_A = \boldsymbol{x}_A(\boldsymbol{q}) \tag{10}$$
$$\dot{\boldsymbol{x}}_A = \mathbf{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{11}$$

where $\mathbf{J}_A$, is the analytical Jacobian matrix

$$\mathbf{J}_A(\boldsymbol{q}) = \frac{\partial \boldsymbol{x}_A(\boldsymbol{q})}{\partial \boldsymbol{q}} . \tag{12}$$

In general angular velocities $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\Phi}}$ are not equal. Assuming that $\dot{\boldsymbol{x}}_A$ is 6-dimensional vector The relation between $\boldsymbol{v}$ and $\dot{\boldsymbol{x}}_A$ is given in the form [4]

$$\boldsymbol{v} = \begin{bmatrix} \mathbf{I}_3 & 0 \\ 0 & \mathbf{A}(\boldsymbol{\Phi}) \end{bmatrix} \dot{\boldsymbol{x}}_A , \tag{13}$$

where $\mathbf{A}(\boldsymbol{\Phi})$ is 3×3 matrix depending on the selection of Euler angles. Matrix $\mathbf{A}(\boldsymbol{\Phi})$ can be obtained from the time derivative of the rotation matrix $\mathbf{R}(\boldsymbol{\Phi})$ in Eq. (8).

The main drawback of Euler angles representation are representation singularities, which occur for some angles (depending on the Euler angles selection). To avoid these singularities the orientation can be represented using Euler parameters (unit quaternions) $\mathcal{Q} = \{\eta, \boldsymbol{\epsilon}\}$. The scalar part of $\mathcal{Q}$ is defined as

$$\eta = \cos(\varphi/2) \tag{14}$$

and the vector part as

$$\boldsymbol{\epsilon} = \sin(\varphi/2)\boldsymbol{r} , \tag{15}$$

[1]Columns $\boldsymbol{n}$, $\boldsymbol{s}$ and $\boldsymbol{a}$ are representing x, y and z axis of the frame $\mathcal{S}_e$

where φ and $\boldsymbol{r}$ are the angle and the vector of the equivalent angle/axis description of the rotation matrix $\mathbf{R}$. The relation between the quaternion time derivative $\dot{\mathcal{Q}}$ and the rotational velocity $\boldsymbol{\omega}$ can be obtained by the quaternion propagation [4]

$$\dot{\mathcal{Q}} = \{\dot{\eta}, \dot{\boldsymbol{\epsilon}}\} = \{-\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\omega}, -\frac{1}{2}\mathbf{E}(\eta, \boldsymbol{\epsilon})\boldsymbol{\omega}\} \,, \tag{16}$$

where

$$\mathbf{E}(\eta, \boldsymbol{\epsilon}) = \eta\mathbf{I} - \mathbf{S}(\boldsymbol{\epsilon}) \,. \tag{17}$$

III. Task space selection

In many cases, it is convenient to describe the location of the end-effector $\mathbf{T}_e$ in a suitably defined task frame, possibly also moving. Fig. 1 shows a general situation where $\mathcal{S}_b$ is the base (world) frame with the origin in O_b and $\mathcal{S}_t$ is the selected task frame with the origin in O_t. The location of frame $\mathcal{S}_t$ in frame $\mathcal{S}_b$ is defined by the translation vector $\boldsymbol{p}_t$ connecting the origins O_b and O_t, and the rotation matrix R_t representing the rotation between O_b and O_t, which yields the following expression for the end-effector location ${}^t\mathbf{T}$ in $\mathcal{S}_t$

$$\begin{aligned} {}^t\mathbf{T} &= \begin{bmatrix} {}^t\mathbf{R} & {}^t\boldsymbol{p} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_t^T & -\mathbf{R}_t^T\boldsymbol{p}_t \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \boldsymbol{p} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_t^T\mathbf{R} & \mathbf{R}_t^T(\boldsymbol{p} - \boldsymbol{p}_t) \\ \mathbf{0} & 1 \end{bmatrix} , \end{aligned} \tag{18}$$

where ${}^t\boldsymbol{p}$ and ${}^t\mathbf{R}$ are the end-effector position vector and the orientation matrix in the task frame $\mathcal{S}_t$, respectively. Hereafter, we use the notation ${}^t(\cdot)$ to denote that a quantity is expressed with respect to the frame $\mathcal{S}_t$. Otherwise the quantity is expressed with respect to the base frame $\mathcal{S}_b$.

How the task frame $\mathcal{S}_t$ is selected depends on the task the robot should perform. Typically, the frame $\mathcal{T}$ is selected to coincide with the base frame $\mathcal{S}_b$ ($\boldsymbol{p}_t = [0\,0\,0]^T$ and $\mathbf{R}_t = \mathbf{I}$). When the end-effector motion is constrained and the constraints restrict the motion in some spatial directions, it is convenient to align the task space with the constraints [2], [14], or when the task requires different behavior in spatial directions. These spatial directions can depend on the task space motion or as shown later on the null-space motion. Therefore, we assume hereafter that the task frame is time-variant, $\boldsymbol{p}_t = \boldsymbol{p}_t(t)$ and $\mathbf{R}_t = \mathbf{R}_t(t)$.

The differential kinematics in task frame $\mathcal{S}_t$ can be obtained by differentiating Eq. (18). Let $\boldsymbol{v}_t = [\dot{\boldsymbol{p}}_t^T \;\; \boldsymbol{\omega}_t^T]^T$ be the velocity of the task frame origin O_t with respect to $\mathcal{S}_b$, and ${}^t\boldsymbol{v} = [{}^t\dot{\boldsymbol{p}}^T \; {}^t\boldsymbol{\omega}^T]^T$ be the operational point velocity with respect to $\mathcal{S}_t$. Then, differentiating Eq. (18) yields

$${}^t\boldsymbol{v} = \widetilde{\mathbf{R}}_t^T\boldsymbol{v} - \widetilde{\mathbf{R}}_t^T\mathbf{J}_t\boldsymbol{v}_t \,, \tag{19}$$

where $\widetilde{\mathbf{R}}_t$ is a 6×6 matrix

$$\widetilde{\mathbf{R}}_t = \begin{bmatrix} \mathbf{R}_t & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{R}_t \end{bmatrix} \tag{20}$$

and $\mathbf{J}_t$ is a 6×6 matrix defined as

$$\mathbf{J}_t = \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{S}(\boldsymbol{p} - \boldsymbol{p}_t) \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} . \tag{21}$$

IV. Kinematic control

Since the beginning of robotics many different approaches to the control of robots have been proposed. As robot manipulators are highly nonlinear dynamical systems, most of the proposed control strategies use some kind of inner-loop inverse dynamic control like computed torque technique [15] or operational space control [2] to compensate nonlinearities. Then, a kinematic controller in the outer-loop provides necessary control signals for the inner-loop from the desired task-space velocities and accelerations, and exploits the redundancy of the system in order to optimize the motion considering some performance criteria [3]. Here, we focus on the redundancy resolution at the velocity level and we assume that the inner-loop controller already compensates the nonlinear dynamics of the robot.

The main idea of the redundancy resolution at the velocity level is to compute the necessary outer-loop control velocity $\dot{\boldsymbol{q}}_c$ by inverting Eq. (7) or (11) [16]

$$\dot{\boldsymbol{q}}_c = \mathbf{J}^{\#}(\boldsymbol{q})\dot{\boldsymbol{x}}_c + \left(\mathbf{I} - \mathbf{J}^{\#}(\boldsymbol{q})\mathbf{J}(\boldsymbol{q})\right)\dot{\boldsymbol{q}}_n \,, \tag{22}$$

where $\dot{\boldsymbol{x}}_c$ represents the desired task-space control velocity, and $\dot{\boldsymbol{q}}_n$ is an arbitrary joint velocity, which is projected into the null-space of $\mathbf{J}$ and can be used to perform some additional lower priority subtasks. Note that $\dot{\boldsymbol{x}}_c$ can represent the velocity in any task space as long as the Jacobian matrix $\mathbf{J}$ relates the robot configuration space and the selected task space.

It is common to select the task-space control velocity $\dot{\boldsymbol{x}}_c$ in (22) as

$$\dot{\boldsymbol{x}}_c = \boldsymbol{v}_d + \mathbf{K}_p\boldsymbol{e} \,, \tag{23}$$

where $\boldsymbol{v}_d$ and $\boldsymbol{e}$, $\boldsymbol{e} = [{\boldsymbol{e}_p}^T \;\; {\boldsymbol{e}_o}^T]^T$, are the desired end-effector position/rotation velocity and the end-effector position/orientation error expressed in the base frame $\mathcal{S}_b$, respectively. Matrix $\mathbf{K}_p$ represents the gains, which define the close-loop behavior.

A. Task space pose error

Let $\boldsymbol{x}_d = \{\boldsymbol{p}_d, \mathbf{R}_d\}$ be the desired pose and $\boldsymbol{x}_e = \{\boldsymbol{p}_e, \mathbf{R}_e\}$ the actual pose of the robot end-effector. Then, the position error in Eq. (23) can be defined as

$$\boldsymbol{e}_p = \boldsymbol{p}_d - \boldsymbol{p}_e \,. \tag{24}$$

The definition of the orientation error is not so easy as it depends on the particular orientation representation used [17], [18], [19], [20]. Although, using rotation matrix $\mathbf{R}$ in control as proposed in [17] is not very convenient, it is worth mentioning that the orientation error is related to the mutual orientation $\mathbf{R}_{de}$ between the desired orientation $\mathbf{R}_d$ and actual orientation $\mathbf{R}_e$, which can be expressed as [18]

$$\mathbf{R}_{de} = \mathbf{R}_d\mathbf{R}_e^T \,. \tag{25}$$

Note that $\mathbf{R}_{de}$ is expressed in the base frame $\mathcal{S}_b$. The control objective is that the pose errors converge to 0, i.e.

$$\lim_{t\to\infty} \boldsymbol{e}_p = \mathbf{0} \quad \text{and} \quad \lim_{t\to\infty} \mathbf{R}_{de} = \mathbf{I} \,. \tag{26}$$

The simplest way and analogous to Eq. (24) is to define the orientation error using Euler angles

$$\boldsymbol{e}_o = \Delta\phi = \boldsymbol{\Phi}_d - \boldsymbol{\Phi}_e \tag{27}$$

where $\boldsymbol{\Phi}_d$ and $\boldsymbol{\Phi}_e$ are the desired and actual Euler angles, respectively. To overcome some representation singularity problems, the orientation error can be defined as the Euler angles of the rotation matrix $\mathbf{R}_{de}$ [19]. However, it is necessary to select such Euler angles combination that the corresponding representation singularity is not at $\boldsymbol{\Phi} = \mathbf{0}$.

For singularity-free control it is more convenient to define the orientation error using Euler parameters. As before, we can define the error using Euler parameters of the desired and actual orientation, $\mathcal{Q}_d$ and $\mathcal{Q}_e$, resulting in [8]

$$\Delta\mathcal{Q} = \{\Delta\eta, \Delta\boldsymbol{\epsilon}\} = \mathcal{Q}_d * \mathcal{Q}_e^{-1} \tag{28}$$

or directly using the mutual orientation matrix

$$\Delta\mathcal{Q} = \mathcal{Q}(\mathbf{R}_{de}) \,. \tag{29}$$

It can be easily seen that (26) implies

$$\lim_{t\to\infty} \Delta\mathcal{Q} = \{1, \mathbf{0}\} \tag{30}$$

and it is a common practice to use in the control algorithm only the vector part of $\Delta\mathcal{Q}$ [18]

$$\boldsymbol{e}_o = \Delta\boldsymbol{\epsilon} \,. \tag{31}$$

B. Functional redundancy

A redundant robot has more DOFs than it is necessary to perform a task. This implies that the redundancy is not only a feature of the robot structure but depends also on the task — there are tasks for which the robot becomes redundant. Executing a typical robot task, which requires that the end-effector follows a trajectory in the 6-dimensional operational space, a robot with more than six joints in a reasonable configuration is *intrinsic redundant*, $\dim(\mathcal{O}) < \dim(\mathcal{C})$. For intrinsic redundant robots the redundancy resolution is based on some kind of generalized inverse the Jacobian matrix $\mathbf{J}_G(\boldsymbol{q})$, which is typically a $6 \times n$ matrix with $n > 6$, assuming that the position and orientation are considered.

However, there are task that do not require that all 6 spatial DOFs are controlled. This means that task space $\mathcal{T}$ is a subspace of the operational space $\mathcal{O}$, $\dim(\mathcal{T}) < \dim(\mathcal{O})$, and the robot becomes *functional redundant*. We assume that functionally redundant robots are able to perform a motion in all 6 spatial DOFs, but the task requires less than 6 DOFs. Further, we assume here also that the functional redundancy is related only to the orientation. To exploit the available functional redundancy it is necessary to find a task frame, where the redundant DOFs are rows of the Jacobian matrix. In most cases, even for 5 DOFs tasks, it is impossible to find such a frame which would be time-invariant and would assure that the redundant DOFs are rows of the Jacobian matrix [21]. Therefore, we propose to define a time-variant task frame in which the functional redundancy are rows of the Jacobian matrix all the time.

First, knowing the redundant spatial directions in frame $\mathcal{S}_b$, we define the task frame $\mathcal{S}_t$ so that one or two of its axes are aligned with the redundant spatial directions. When all three orientation directions are redundant, this is a trivial case and is not considered here.

Next, we map the control given by Eq. (22) into the task space $\mathcal{S}_t$ using Eq. (19). As only orientation DOFs are included in the functional redundancy it is enough to rotate the task space and use $\boldsymbol{p}_t = \boldsymbol{p}$. This simplifies the mapping; the task space rotation can be applied only to the orientation part of the control (22), which yields

$$\widetilde{\mathbf{R}}_t = \begin{bmatrix} \mathbf{I} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{R}_t \end{bmatrix} \tag{32}$$

and $\mathbf{J}_t = \mathbf{I}$. Using this in Eq. (18), the position/orientation error in frame $\mathcal{S}_t$ becomes

$${}^t\boldsymbol{e} = \widetilde{\mathbf{R}}_t^T \boldsymbol{e} \,. \tag{33}$$

and from Eq. (19) we get

$${}^t\boldsymbol{v}_d = \widetilde{\mathbf{R}}_t^T \boldsymbol{v}_d - \widetilde{\mathbf{R}}_t^T \mathbf{J}_t \boldsymbol{v}_t \,. \tag{34}$$

Substituting Eqs. (33) and (34) into Eqs. (22) and (23), and rearranging them, we obtain the kinematic velocity controller in the form

$$\dot{\boldsymbol{q}}_c = (\widetilde{\mathbf{R}}_t^T \mathbf{J})^{\#}\, {}^t\dot{\boldsymbol{x}}_c + (\mathbf{I} - (\widetilde{\mathbf{R}}_t^T \mathbf{J})^{\#} \widetilde{\mathbf{R}}_t^T \mathbf{J}) \dot{\boldsymbol{q}}_n \,, \tag{35}$$

where

$${}^t\dot{\boldsymbol{x}}_c = \widetilde{\mathbf{R}}_t^T \boldsymbol{v}_d + \mathbf{K}_{p,t} \widetilde{\mathbf{R}}_t^T \boldsymbol{e} \,. \tag{36}$$

Here, the term $\widetilde{\mathbf{R}}^T\mathbf{J}$ can be denoted as the *task frame Jacobian*. Comparing Eqs. (22) and (35) we can see that both controllers are similar. The only difference is that in the controller (35) the Jacobian matrix and task velocities are mapped using matrix $\widetilde{\mathbf{R}}_t$.

To consider the functional redundancy in the control (35) we exclude some rows from the Jacobian matrix and also the corresponding components of the desired velocity $\boldsymbol{v}_d$ and the error $\boldsymbol{e}$. It is convenient to do this by replacing the matrix $\widetilde{\mathbf{R}}_t$ in Eqs. (35) and (36) with a modified matrix $\widehat{\mathbf{R}}_t$ where rows corresponding to redundant DOFs are excluded. For example, if the orientation around the task frame vector $\boldsymbol{a}$ (z-axis) should not be part of the task controller, then we use

$$\widehat{\mathbf{R}}_t = [\mathbf{I}_{5\times5} \;\; \mathbf{0}_{5\times1}]\, \widetilde{\mathbf{R}}_t \,. \tag{37}$$

The kinematic velocity controller in the form

$$\dot{\boldsymbol{q}}_c = (\widehat{\mathbf{R}}_t^T \mathbf{J})^{\#}\, {}^t\dot{\boldsymbol{x}}_c + (\mathbf{I} - (\widehat{\mathbf{R}}_t^T \mathbf{J})^{\#} \widehat{\mathbf{R}}_t^T \mathbf{J}) \dot{\boldsymbol{q}}_n \,, \tag{38}$$

and

$${}^t\dot{\boldsymbol{x}}_c = \widehat{\mathbf{R}}_t^T \boldsymbol{v}_d + \mathbf{K}_{p,t} \widehat{\mathbf{R}}_t^T \boldsymbol{e} \,. \tag{39}$$

The basic characteristics of the redundancy resolution control algorithms is that the null-space motion should not influence the task-space motion. If the orientation error used in Eq. (37) is based on the quaternions (Eq. (31)) and some of the error components are not part of the control as in our case, then the term $\widehat{\mathbf{R}}_t^T \boldsymbol{e}$ depends on the rotation of the end-effector in the null-space. This influence can be compensated

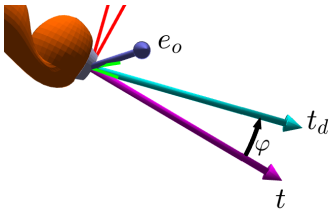

Fig. 2. Desired and actual end-effector frames for tasks with one redundant orientation DOF. The pointing direction is aligned with the end-effector $\boldsymbol{z}$-axis.

if the mutual orientation between the desired and actual orientation $\mathbf{R}_{de}$ is not calculated using Eq. (25) but using

$$\mathbf{R}_{de} = \mathbf{R}_d \left(\mathbf{R}_f(\varphi)\mathbf{R}_e\right)^T , \tag{40}$$

where $\mathbf{R}_f(\varphi)$ is a rotation matrix depending on the orientation angle around the redundant direction and φ the corresponding orientation angle. With $\mathbf{R}_f(\varphi)$ we actually cancel the rotation of the end-effector in the null-space. Here we assume also that no rotation around the redundant DOF is present in $\mathbf{R}_d$.

To avoid this additional computations and make the control faster, we propose another control algorithm for the orientation control, where not all orientation components are controlled. The proposed control algorithms depend on how many orientation components have to be controlled.

When the task requires orientation in two directions, then this orientation can be represented as a unit vector pointing in the desired direction $\boldsymbol{t}_d$. The available null-space motion is the motion around this vector. The control algorithm has to assure that the actual pointing direction $\boldsymbol{t}$ converges to the desired one (see Fig. 2). In other words, the angle φ between $\boldsymbol{t}_d$ and $\boldsymbol{t}$ should converge to 0. For that, we define the orientation error in Eq. (39) as[2]

$$\boldsymbol{e}_o = \varphi \frac{\boldsymbol{t} \times \boldsymbol{t}_d}{\|\boldsymbol{t} \times \boldsymbol{t}_d\|} , \tag{41}$$

where $\varphi = \arccos(\boldsymbol{t}^T \boldsymbol{t}_d)$. This orientation error $\boldsymbol{e}_o$ is not dependent on the motion in the null-space. When the robot is functionally redundant due to the axis-symmetric tool, the direction of $\boldsymbol{t}$ is aligned with the tool axis and $\boldsymbol{t}$ has be selected as the redundant direction instead of $\boldsymbol{t}_d$.

When two orientation components are not part of the task space the situation is complementary to the previous one. The direction of the pointing vector $\boldsymbol{t}$ depends on the null-space motion and the rotation ϕ around $\boldsymbol{t}$ represents the orientation in the task space. In this case, the task orientation ${}^t\dot{x}_c$ is a scalar and the task space controller becomes

$${}^t\dot{x}_c = \dot{\phi}_d + K_{p,t}(\phi_d - \phi) , \tag{42}$$

where ϕ_d and ϕ are the desired and actual rotation in the task space, respectively; $\dot{\phi}_d$ is the desired task rotation velocity.

[2]All direction vectors are assumed to be unit vectors.

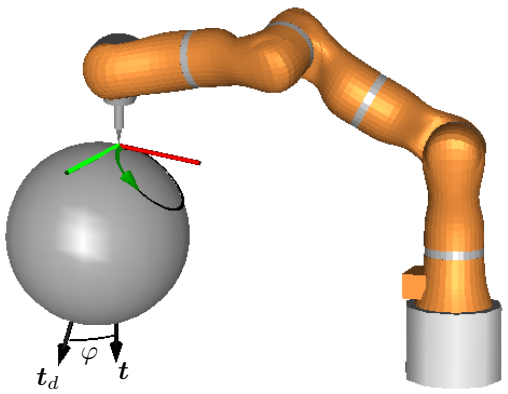

Fig. 3. KUKA LWR robot with an axis-symmetric tool moving perpendicular on a circular path on a sphere surface.

V. Case Studies

Typical examples of robot tasks with one functional redundant DOF are arc-welding, laser-cutting or spray-painting applications, where the tool is axis-symmetric. To illustrate the performance of the proposed control approach we have selected a task, where a 7DOF robot (KUKA LWR) with an axis-symmetric tool has to move with a constant velocity along a circular path on a ball, as it is shown in Fig. 3. The tool should be always perpendicular to the surface. Obviously, the tool orientation around the surface normal is not important for the task. Therefore, the task frame $\mathcal{S}_t$ has been rotated so that one axis, in our case the tool axis $\boldsymbol{a}_{\mathcal{S}_t}$, has been aligned with the surface normal $\boldsymbol{n}_s$. As the task frame $\mathcal{S}_t$ depends on the location of the end-effector on the surface, the orientation of $\mathcal{S}_t$ is changing during the task execution.

The desired motion has been defined so that the end-effector is moving with a constant linear velocity along a circular path on the ball surface. As the rotation around the tool axis has been excluded from the task control, the pointing direction $\boldsymbol{t}_d$ has been selected as $\boldsymbol{t}_d = \boldsymbol{n}_s$, task space has been oriented as the tool frame, $\mathbf{R}_t = \mathbf{R}_e$, and $\widehat{\mathbf{R}}_t$ was defined as in Eq. (37).

The self motion due to the intrinsic and functional redundancy has not been used for any secondary tasks. Rather we have defined $\boldsymbol{q}_d$ in controller (38) so that intensive motion in the null-space has been achieved to show the performance of the proposed task controller. The initial robot configuration has been selected so that the end-effector was not on the path and the system had an initial position and orientation error.

We have compared the task controller (39), where the error $\boldsymbol{e}_o$ has been calculated using Eq. 40 and (41) with gains $\mathbf{K}_{p,t} = 5\mathbf{I}$. The results were identical for both controllers. Therefore, it is more efficient to use the error as defined in Eq. (41) in the controller. The simulation results are shown in the Fig. 4. Additionally, the animation is presented in the multimedia attachment as Example 1. The top plot in Fig. 4 shows the desired and the actual rotational velocities in the task space $\mathcal{S}_t$. We can see that the actual velocities are

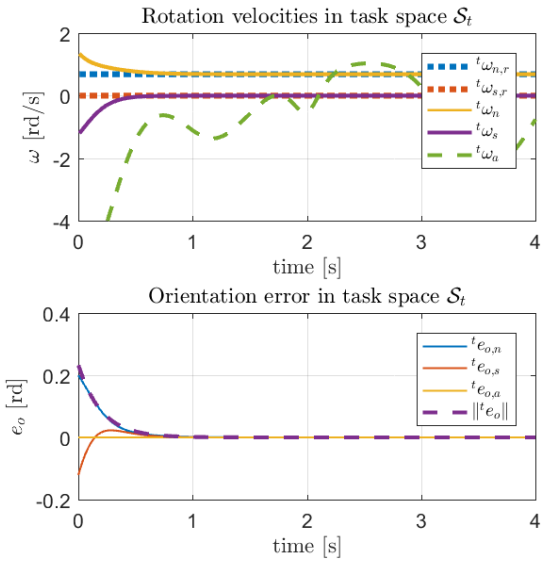

Fig. 4. Simulation results: KUKA LWR robot moving along circular path on a ball and the tool is prependicular to the surface ($\mathbf{K}_{p,t} = 5\mathbf{I}$) - The desired rotation velocity components in task space ${}^t\omega_{n,r}$ and ${}^t\omega_{s,r}$ (dotted line); the actual rotation velocity components in task space ${}^t\omega_n$ and ${}^t\omega_s$ (solid line), and the rotation velocity component due to the null-space motion ${}^t\omega_a$ (dashed line).

converging to the desired one although the null-space motion was changing significantly the end-effector orientation (see ${}^t\omega_a$ signal). The bottom figure shows the orientation error in the task space $\mathcal{S}_t$. As we can see the orientation error $\boldsymbol{e}_o$ is converging to 0.

As an example of an application where only one orientation component is important for the task, we have selected the task where an "L" shaped tool has to touch a wave shaped conical surface (see Fig. 5). For this task, the rotation around main cone axis $\boldsymbol{a}_c$ and the rotation around the tool axis $\boldsymbol{n}$ are not important for the task. Only the orientation component necessary to rotate the tool onto the surface is important and has to be included in the task controller. Therefore, the pointing direction was defined as

$$\boldsymbol{t} = \frac{\boldsymbol{a}_c \times \boldsymbol{n}}{\|\boldsymbol{a}_c \times \boldsymbol{n}\|} \tag{43}$$

and the task space motion is represented by the rotation ϕ around $\boldsymbol{t}$ with the value corresponding to the angle between $\boldsymbol{a}_c$ and $\boldsymbol{n}$

$$\phi = \arccos(\boldsymbol{a}_c^T \boldsymbol{n}) \tag{44}$$

as it is shown on Fig. 5. For this task, the task frame rotation matrix $\mathbf{R}_t$ was selected as

$$\mathbf{R}_t = \left[\frac{\boldsymbol{t} \times \boldsymbol{a}_c}{\|\boldsymbol{t} \times \boldsymbol{a}_c\|} \quad \boldsymbol{t} \quad \boldsymbol{a}_c \right] . \tag{45}$$

In our case, where the surface is wave shaped, the desired task space orientation ϕ_d depends on the location of the tool on the surface

$$\phi_d = (-\frac{\pi}{24})(1 - cos(6 * \varphi)) + \frac{\pi}{2} . \tag{46}$$

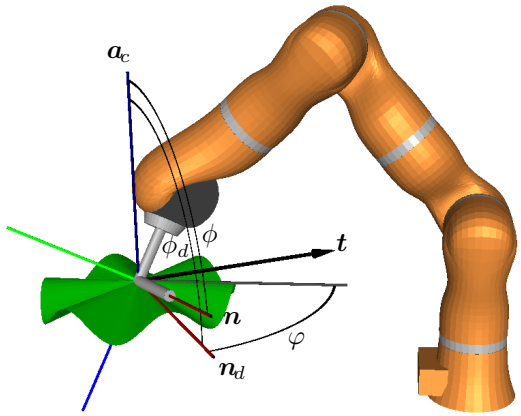

Fig. 5. KUKA LWR robot with as "L" shaped tool and the tool has to touch the surface.

Note that the orientation angle φ on the surface depends only on the null-space motion. We have used the task space controller (42) with gains $\mathbf{K}_{p,t} = 5$.

As before, the self motion due to the intrinsic and functional redundancy has not been used for any secondary tasks, but we have defined $\boldsymbol{q}_d$ so that intensive smooth motion in the null-space has been achieved. In the initial configuration the tool has been above the surface so that the system had an initial task position and orientation error.

The simulation results are shown in Fig. 6 and the animation is presented in the multimedia attachment as Example 2. The top plot in Fig. 6 shows the desired and actual rotational velocities in the task space $\mathcal{S}_t$. We can see that the actual velocity ϕ is converging to ϕ_d although the null-space motion was changing significantly the end-effector orientation (see ${}^t\omega_n$ and ${}^t\omega_a$ signals). Note that the desired task rotation ϕ_d had to follow the changes in φ caused by the null-space motion so that the tool was all the time on the surface. The bottom figure shows the orientation error ${}^te_o = \phi_d - \phi$ in the task space $\mathcal{S}_t$ and we can see that the error is converging to 0.

VI. Experimental results

The proposed control strategy has been implemented on two KUKA LWR robot arms operating as a two-arm robot system as shown in Fig. 7. As the task we have selected the "buzz wire" task, i.e. to move the ring, which is in our case attached to the left LWR robot arm, along the wire, which is held by the right LWR robot arm. For convenience, the wire has been in a circular form (diameter 0.25m). During the motion the wire has to be in the middle of the ring (the ring and the wire should not touch) and the ring should be perpendicular to the wire (see Fig. 8). Note that the orientation of the ring around the wire is not important for the task. Therefore, the ring can be freely rotated around the wire and this rotation represents the functional redundancy (see Fig. 9). Our dual-arm experimental system has 14 DOF and considering the functional redundancy the system has altogether nine redundant DOF.

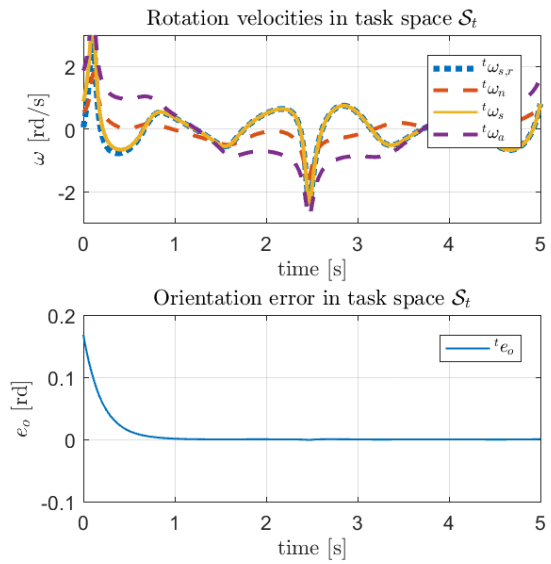

Fig. 6. Simulation results: KUKA LWR robot with "L" shaped tool moving on a surface ($\mathbf{K}_{p,t} = 5\mathbf{I}$) - the desired rotation velocity component in task space ${}^t\omega_{s,r}$ (dotted line); the actual rotation velocity component in task space ${}^t\omega_s$ (solid line), and the rotation velocity components due to the null-space motion ${}^t\omega_n$ and ${}^t\omega_a$ (dashed line).

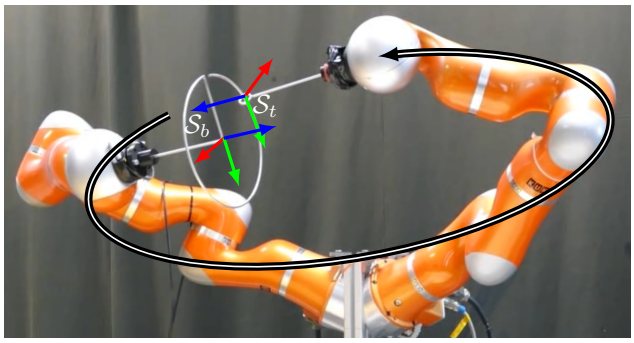

Fig. 7. Two-arm KUKA LWR robot system

To do the task, the motion of both robot arms has to be coordinated. As for the "buzz wire" task the relative motion between robot end-effectors is important, we have modelled both arms as one kinematic chain starting at the end-effector of the right arm ending at the end-effector of the left arm as indicated with an arrow in Fig. 7. So, the right arm end-effector frame represents the base frame $\mathcal{S}_b$ of the robot system and the desired motion is defined in $\mathcal{S}_b$ as

$$\boldsymbol{p}_d = 0.125[\cos(\varphi(t)) \;\; \sin(\varphi(t)) \;\; 0]^T \tag{47}$$

and

$$\mathbf{R}_d = \mathbf{R}_z(-\varphi(t))\mathbf{R}_x(\pi)\,, \tag{48}$$

where $\varphi(t)$ represents the path parameter. Matrices $\mathbf{R}_z$ and $\mathbf{R}_x$ are the rotation matrices representing the rotations around z and x axis of the frame $\mathcal{S}_b$, respectively. For this task we have used the control strategy as explained in Fig. 2; vector av was in the direction of the path and the task

Fig. 8. "Buzz wire" task

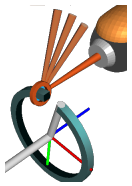

Fig. 9. "Buzz wire" self motion

frame $\mathcal{S}_t$ was oriented so that frame verctor $\boldsymbol{s}$ (y-axis) has been aligned with $\boldsymbol{t}$ (see Fig. 7). We have used the controller (38) and (39), where the components corresponding to the rotation around y-axis have been removed from $\widetilde{\mathbf{R}}_t$

$$\widehat{\mathbf{R}}_t = diag(1\,1\,1\,1\,0\,1])\widetilde{\mathbf{R}}_t \tag{49}$$

To control two LWR robot arms as one system we have developed a special MATLAB/Simulink xPC target server which communicates with Fast Research Interface (FRI) provided by KUKA [22]. The joint motion generated by the controller (38) has been then used as the desired joint motion of each LWR robot using FRI joint position control mode.

The redundant DOFs offered by the system have been used to perform a lower priority subtasks. For demonstration we have implemented an algorithm, which allows a human to reconfigure both arms without disturbing the main task simply by pushing any link of the robots. The avoiding motion has been calculated using the information about the external forces action on the body of the robot

$$\dot{\boldsymbol{q}}_n = -\mathbf{K_n}\,\mathbf{J}_G^T(\boldsymbol{q})\boldsymbol{F}_{ext} \tag{50}$$

Gains $\mathbf{K_n}$ are selected to achieve a suitable compliance in the null-space. As KUKA LWR controller provides internal joint torque sensors due to external forces, the above controller can be simplified into

$$\dot{\boldsymbol{q}}_n = -\mathbf{K_n}\,\boldsymbol{\tau}_{ext}(\boldsymbol{F}_{ext})\,, \tag{51}$$

where $\boldsymbol{\tau}_{ext}$ are external joint torques due to the external forces $\boldsymbol{F}_{ext}$.

Fig. 10 shows a motion sequence of two LWR robots performing the "buzz wire" task while a human is re-configuring the robot. As it can be seen, the ring is not touching the wire although the arms are reconfiguring into the pose to which a human pushes the links. Note that the rotation of the ring around the wire is also utilized for the self motion. The experimental video is presented in the multimedia attachment.

VII. Conclusions

We have shown how to augment the null-space motion of redundant robots with additional DOFs offered by functional redundancy subject to unused end-effector rotational DOFs. As standard quaternion control is not suitable when one or two rotations are excluded from the task control, we have proposed modified orientation control algorithms at velocity level where the task frame is rotated so that the non-controlled rotations can be excluded from the task space

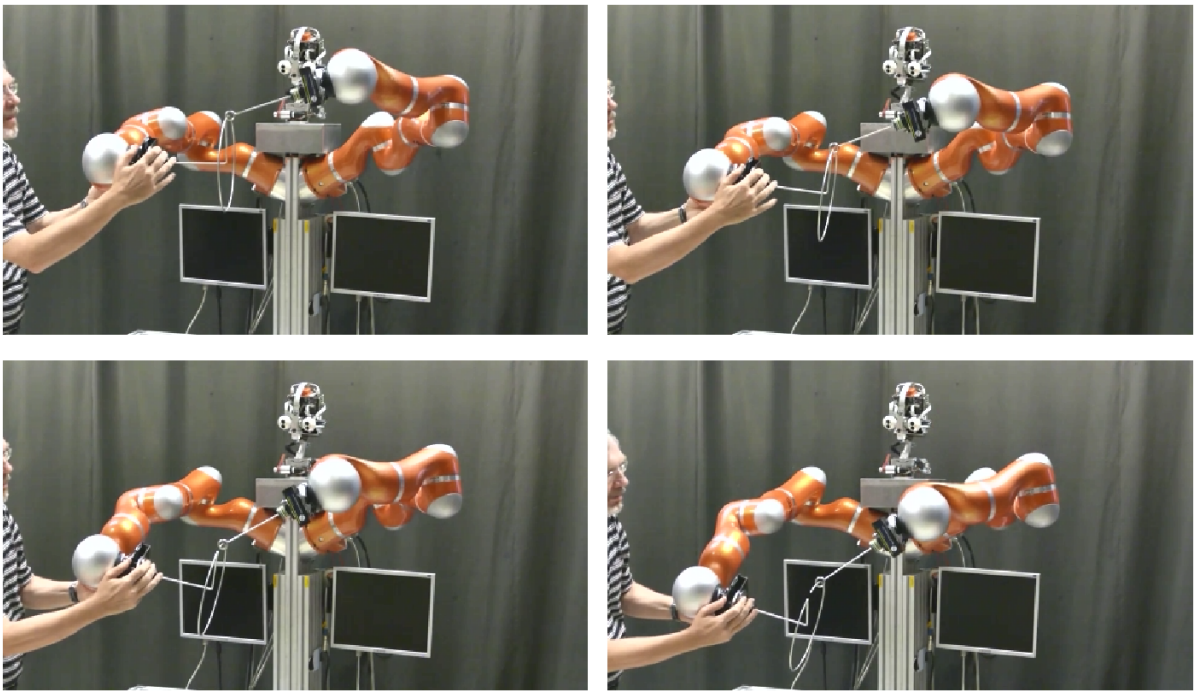

Fig. 10. Experimantal results: Dual-arm LWR robots performing the "ring" task and interacting with a human (Time interval between frames is 1s)

and used for self motion. The proposed control can be easily extended to the acceleration level.

REFERENCES

[1] D. E. Whitney, "Resolved motion rate control of manipulators and human protheses," *IEEE Trans. Man-Machine Syst.*, vol. MMS-10, pp. 47–53, 1969.

[2] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[3] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational Space Control: A Theoretical and Empirical Comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, jun 2008.

[4] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics - Modelling, Planning and Control*. Springer-Verlag London, 2009.

[5] J. Russakow, O. Khatib, and S. Rock, "Extended operational space formulation for serial-to-parallel chain (branching) manipulators," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1995, pp. 1056–1061.

[6] O. Brock, J. Kuffner, and J. Xiao, "Motion for Manipulation Tasks," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, ch. 26, pp. 615–645.

[7] F. Caccavale and M. Uchiyama, "Cooperative Manipulators," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ch. 29, pp. 701–718.

[8] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed., ser. Advanced textbooks in control and signal processing. Springer, London, 2000.

[9] L. Baron, "A Joint-Limits Avoidance Strategy for Arc-Welding Robots," in *Int. Conf. on Integrated Design and Manufacturing in Mech. Eng.*, Montreal, Canada, 2000.

[10] L. Huo and L. Baron, "Kinematic inversion of functionally-redundant serial manipulators: Application to arc-welding," *Transactions of the Canadian Society for Mechanical Engineering*, vol. 29, no. 4, pp. 679–690, 2005.

[11] A. M. Zanchettin and P. Rocco, "On the use of functional redundancy in industrial robotic manipulators for optimal spray painting," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 18, no. PART 1, 2011, pp. 11 495–11 500.

[12] J. Leger and J. Angeles, "A Redundancy-resolution Algorithm for Five-degree-of-freedom Tasks via Sequential Quadratic Programming," in *TrC-IFToMM Symposium on Theory of Machines and Mechanisms,*, Izmir, 2015.

[13] P. J. From and J. T. Gravdahl, "A real-time algorithm for determining the optimal paint gun orientation in spray paint applications," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 4, pp. 803–816, 2010.

[14] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[15] P. Hsu, J. Hauser, and S. Sastry, "Dynamic Control of Redundant Manipulators," *J. of Robotic Systems*, vol. 6, no. 2, pp. 133–148, 1989.

[16] L. Žlajpah and T. Petrič, *Serial and parallel robot manipulators - kinematics, dynamics, control and optimization*. Rijeka: InTech, 2012, ch. Obstacle avoidance for redundant manipulators as control problem, pp. 203 – 230.

[17] J. Luh, M. Walker, and R. Paul, "Resolved-Acceleration Control of Mechanical Manipulators," *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 468–474, 1980.

[18] J. S. C. Yuan, "Closed-Loop Manipulator Control Using Quaternion Feedback." *IEEE journal of robotics and automation*, vol. 4, no. 4, pp. 434–440, 1988.

[19] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Resolved-acceleration control of robot manipulators : A critical review with experiments," *Robotica*, vol. 16, no. September 2000, pp. 565–573, 1998.

[20] R. Campa and H. D. Torre, "Pose Control of Robot Manipulators Using Dierent Orientation Representations: A Comparative Review," in *Proceedings of the 45th IEEE Conference on Decision and Control*, no. 1, 2009, pp. 2855–2860.

[21] L. Huo and L. Baron, "The joint-limits and singularity avoidance in robotic welding," *Industrial Robot*, vol. 35, no. 5, pp. 456–464, 2008.

[22] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the kuka lightweight robot," in *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications(ICRA 2010)*, 2010, pp. 15–21.