ICAS-92-7.3.3

RESOLVED MOTION RATE CONTROLLER FOR REDUNDANT ROBOTS WITH LOCAL OPTIMIZATION

Frank Melzer
Institute B of Mechanics
University of Stuttgart
Federal Republic of Germany

Abstract

A manipulator typically consists of the same number of joints as the number of task space coordinates, whereas if the number of degrees of freedom is greater one speaks of redundant manipulators. These manipulators can perform sophisticated tasks, but kinematical redundancy results in a more complicated control problem. This paper is based on the known approach of resolved motion rate control, where the linearized relation between world and joint velocities is utilized to solve the inverse kinematic problem. Kinematic control is performed in position and orientation, with the error-feedback of the orientation vector based on computationally efficient Euler parameters. External constraints, as obstacles in the work space or joint limitations, are incorporated through objective functions. Describing them explicitly by appropriate weighting matrices results in a gain of efficiency. Inverse kinematics is performed by a weighted generalized inverse of the Jacobian, computed by the efficient direct calculation allowing better control close to singular configurations. Two case studies illustrate the performance of this approach. The model used for the simulation is based on an already existing mobile robot with 9 revolute joints. It was shown that the disadvantage of redundancy, a conservative motion in task space does not necessarily lead to a conservative motion in joint space, can be resolved with the use of precomputed configurations. Using this approach of precomputed configurations in the workspace even a globally optimal path may be reached.

Introduction

The demand of manipulators with high mobility in their workspace led to the design of redundant robots. Compared to nonredundant robots these systems may meet external constraints, like obstacle avoidance or limited joint range, and may optimize performance criteria such as minimization of energy. Furthermore a redundant system can overcome the limitations caused by singular regions in the workspace.

At the institute for manufacturing engineering and automation (IPA) of the Fraunhofer research establishment in Stuttgart (FRG) a large-scale manipulator was developed. This robot, mounted on a truck, has a maximum radius of 24 meters and has a load-bearing of up to 1500 kilograms, see fig. 1. The application spectrum of this mobile robot covers interesting fields like maintenance work or cleaning of large aircrafts and ships. It may also serve as a highly flexible crane in construction or manufacturing industry. Nevertheless, the scope of applications for redundant robots is not limited to earth-based operations. Future space missions, like the assembling of very large space structures, may require such flexible manipulators as well.

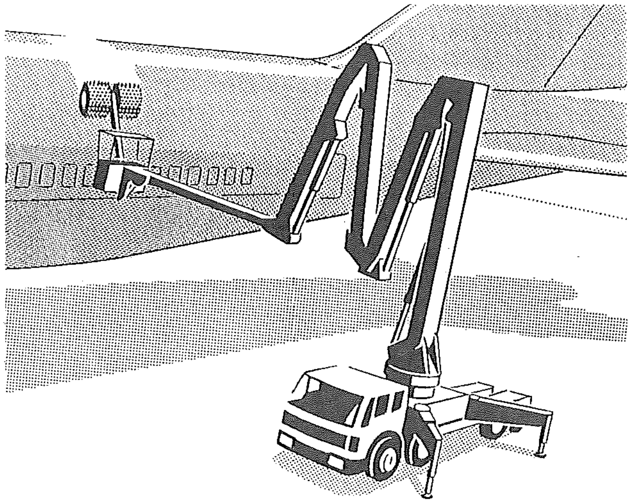


Figure 1. Prospected task of the IPA manipulator: Cleaning large passenger aircraft. *(Putzmeister)*

A manipulator is considered to be *kinematically redundant*, if the number of task space coordinates m are less than the number of degrees of freedom (DOF) n. In cartesian space the position and the orientation of an object is described by six independent coordinates. Therefore, a robot with more than 6 DOF is considered to be redundant. The *degree of redundancy*, defined by the difference $m - n$ between the number of task variables and joint variables, is actually not constant. Singular configurations or physical limitations of the joint range decrease the degree of redundancy.

The crucial point in the analysis of redundant manipulators is the mapping of the joint coordinates $\boldsymbol{\beta} \in \mathrm{R}^n$ to the task or world variables $w \in \mathrm{R}^m$

$$w = f^{-1}(\boldsymbol{\beta}), \tag{1}$$

which is named the *inverse kinematic problem.* Eq. (1) is an *under-determined system of nonlinear equations* and it is clear that there is no unique solution rather than a solution space. The various approaches to resolve redundancy are divided into two groups:[7]

- methods utilizing *local optimization*
- methods utilizing *global optimization*

Various investigations were done at the IPA in order to identify the best solution strategy for a realtime implementation of the inverse kinematic problem. It was found that the local optimization technique, i.e. linearization of eq. (1), is best suited for a fast computation. However this technique has certain drawbacks:

- Global optimization is difficult to fulfil.
- The motion of the robot may be nonconservative, i.e. a closed path in task space does not necessarily lead to a closed path in joint space.(4)
- Optimization criterias are in general difficult to set up and are expensive in terms of computing time.

The goal of this paper is to show an efficient and fast implementation of a local optimization technique which minimizes the above mentioned drawbacks and is suited for realtime implementation on a robot control system which was originally based on a 80286 processor.

Concept of the controller

As stated above, the main requirement for the controller is to perform its task in realtime. Furthermore it is desirable to have a modular approach in the overall design.(9) Both requirements were realized in a layer model, see fig. 2. Each layer has a different time slice and only the lowest level, the kinematic controller, runs at realtime.

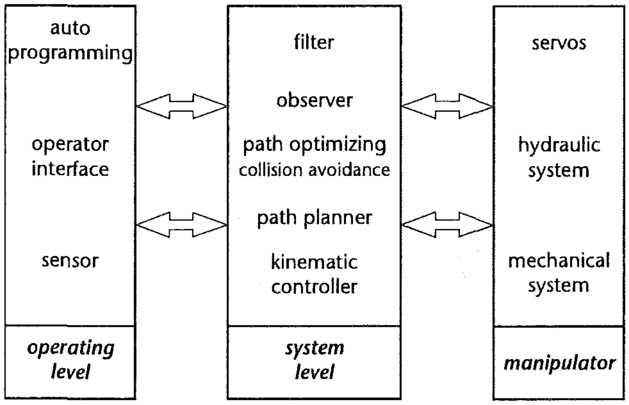


Figure 2. Block diagram of overall control structure

The other layers perform their particular task whenever the processor is scheduling a time slice for them. The *path planner* is responsible for the calculation of the world coordinates of the trajectory, the *observer* feeds back measured data to the kinematic controller, and the *path optimizing* layer handles the obstacle avoidance. This module is further responsible to detect singular regions in the workspace and other singular forms, like fully stretched or folded configurations. Information from the optimizing module will be passed to the *kinematic controller* in form of weighting matrices. This paper emphasises on the design of the lowest level: the kinematic controller which is responsible for the transformation of the world coordinates of the trajectory onto the joint coordinates and vice versa.

Kinematic Controller

The duty of a manipulator on-line control is to move the endeffector (tool centre point) along a prescribed time discretized path in cartesian space, which is given by a sequence of desired positions $w^i = w(t^i)$ in task space R^m. For this purpose *one* solution of eq. (1) is required. Of course this solution should consider external constraints like obstacles. Resolving eq. (1) may be done by utilizing linearized inverse kinematics at velocity level,(7) the so called *resolved motion rate control* (RMRC), or by utilizing a formulation based on inverse kinematics on acceleration level: the *resolved acceleration rate control.*(7) It is clear that RMRC can be used only for a fairly slow motion of the endeffector (quasistatic motion).

Scheme of the solution algorithm

The solution algorithm for a resolved motion rate controller can be divided into the following four steps:

1 *Calculation of the actual position and orientation*

For the time t^i the actual endeffector position in world or task coordinates w^i is calculated from the joint coordinates $\boldsymbol{\beta}^i$

$$w^i = f(\boldsymbol{\beta}^i)\,. \tag{2}$$

This unique transformation is known as *forward kinematics.*

2 *Determination of the position and orientation error*

The path planner calculates for the next time t^{i+1} the following position $w^{i+1} = w(t^{i+1})$ and the difference between actual and desired world coordinates, which can be treated as an error in world coordinates, is calculated. This vector dz consists of a translational vector $\boldsymbol{dr}$ and an orientational vector $\boldsymbol{dp}$

$$dz = \begin{bmatrix} dr \\ dp \end{bmatrix}. \tag{3}$$

3 *Calculation of the new joint coordinates*

The nonunique transformation of the world coordinate space onto the joint coordinate space is known as *inverse kinematic.* Having determined the $\boldsymbol{\beta}^{i+1}$, the new position

$$\bar{w}^{i+1} = f(\boldsymbol{\beta}^{i+1}) \tag{4}$$

of the endeffector may be calculated easily using forward kinematics.

The error $\boldsymbol{\varepsilon}$ of the actual endeffector position may be tolerated as long as it is within acceptable limits. The merit of this strategy is a constant number of operations for each time step, an important feature for realtime control. Not tolerating the error $\boldsymbol{\varepsilon}$ would require a Newton type iteration till the exact position w^{i+1} on the trajectory is reached.

4 *Determination of the new position and orientation*

For the next time increment the position $\bar{w}^{i+1}$ is considered to be exact and the world coordinate error dz for the time step Δt is calculated as the difference between this actual position and the next desired position on the trajectory, see Fig. 3. This strategy forces the endeffector always towards the desired trajectory and the global position and orientation error is minimized.

Forward kinematics

A manipulator consists of a series of bodies, interconnected by joints. Typically the topology of manipulators has a chain structure. The position of a body $j+1$ within such a multibody

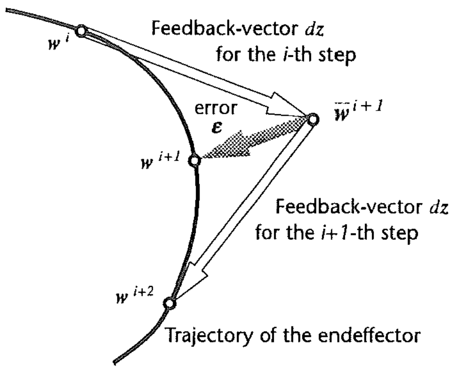


Figure 3.First order approximation of a given trajectory

system relative to a body j is described by a vector ${}^j\boldsymbol{r}_{j,j+1}$. The superscript j denotes here, that this vector is resolved in the coordinate frame attached to body j. Omitting the left superscript denotes the resolution in the inertia frame 0. A transformation matrix ${}^j\boldsymbol{T}_{j+1}$ describes the orientation of body $j+1$ relative to a preceding body j. Position and orientation with respect to the inertia frame 0 are described by

$$\boldsymbol{T}_{j+1} = \boldsymbol{T}_j\,{}^j\boldsymbol{T}_{j+1}, \tag{5}$$

$$\boldsymbol{r}_{0,j+1} = \boldsymbol{r}_{0,j} + \boldsymbol{T}_j\,{}^j\boldsymbol{r}_{j,j+1}. \tag{6}$$

Using homogeneous coordinates, *forward kinematics* described by eqs. (5) and (6) may be simplified as

$$\boldsymbol{D}_{j+1} = \boldsymbol{D}_j\,{}^j\boldsymbol{D}_{j+1}, \tag{7}$$

where the 4x4 matrix $\boldsymbol{D}$ can be written as

$${}^j\boldsymbol{D}_{j+1} = \left[\begin{array}{c|c} {}^j\boldsymbol{T}_{j+1} & {}^j\boldsymbol{r}_{j+1} \\ \hline \boldsymbol{0} & 1 \end{array}\right]. \tag{8}$$

For a manipulator with n joints the world coordinates $\boldsymbol{w}$ are equal to the position and orientation of the endeffector frame $n+1$. Using Denavit-Hartenberg notation(2) for the geometric description of the joints, the matrix $\boldsymbol{D}_{n+1}$ is a function of the joint coordinates $\boldsymbol{\beta}$. Therefore, the world coordinates $\boldsymbol{w}$ are the unique solution of eq. (7) for a given set of joint coordinates $\boldsymbol{\beta}$.

Position and orientation error

For the kinematic controller the difference between the actual and the desired world coordinates are used as the error feedback, see Fig. 3. This error vector dz, eq. (3), consists of a translational and an orientational component. Especially the latter component is not trivial to compute.

Orientation of a coordinate frame in cartesian space may be represented by either a transformation matrix, or by a set of three angles (either Euler or Bryant angles), or by a 4-parametric representation, known as Euler parameters.(8) The latter representation is free of singularities and is computationally efficient. Mathematically speaking Euler parameters are normalized quaternions and are defined as(3)

$$\mathrm{P} = \begin{bmatrix} p_0 \\ \boldsymbol{p} \end{bmatrix}. \tag{9}$$

The components of P can be interpreted as a scalar p_0 and an imaginary vector $\boldsymbol{p}$. Quaternion algebra has significant advantages over vector algebra, since quaternions form a group. Using quaternions the relative orientation of a frame k with respect to a frame i is described by

$$\mathrm{P}_{ik} = \mathrm{P}_{0k}\,\mathrm{P}_{i0} \tag{10}$$

The quaternion P_{0k} represents the absolute orientation of frame k with respect to the inertia frame 0. Introducing the complex conjugate quaternion

$$\bar{\mathrm{P}} = \begin{bmatrix} p_0 \\ -\boldsymbol{p} \end{bmatrix} \tag{11}$$

the orientation of the inertia frame relative to a frame i is expressed by

$$\mathrm{P}_{0i} = \bar{\mathrm{P}}_{i0} \tag{12}$$

From eqs. (9), (11) and (12) the relative difference between actual orientation k and desired orientation i can be computed as

$$\mathrm{P}_{ik} = \begin{bmatrix} \mathrm{P}_{0k}^T\,\mathrm{P}_{0i} \\ -\boldsymbol{L}_{0k}\mathrm{P}_{0i} \end{bmatrix} = \begin{bmatrix} p_{0ik} \\ \boldsymbol{p}_{ik} \end{bmatrix}, \tag{13}$$

where the 3x4 matrix $\boldsymbol{L}$ is given by(8)

$$L = [-\boldsymbol{p}, -\tilde{\boldsymbol{p}} + p_0\boldsymbol{I}\ ]. \tag{14}$$

Here the tilde operator ~ denotes a tensor representing the vector product and $\boldsymbol{I}$ denotes an unit matrix.

For the feedback of the orientational error a 3-parametric representation of eq. (13) is desirable, and it is obvious, that the vector component $\boldsymbol{p}_{ik}$ of the quaternion P_{ik} is a suitable expression for the relative orientation error. Denoting this orientational error resolved in fame i as ${}^i\boldsymbol{dp}$ and substituting eq. (14) in (13) yields

$${}^i\boldsymbol{dp} = p_{0i}\boldsymbol{p}_k - p_{0k}\boldsymbol{p}_i - \tilde{\boldsymbol{p}}_i\boldsymbol{p}_k. \tag{15}$$

Transforming the components of ${}^i\boldsymbol{dp}$ back to the inertia frame 0, ensures that all vectors are resolved in this coordinate frame.

Global asymptotic convergence of this orientation error is shown in(10).

Less cumbersome is the derivation of the translational error $\boldsymbol{dr}$, which is the difference of the endeffector position vector $\boldsymbol{r}_{n+1}$ at time i and at time $i+1$

$$\boldsymbol{dr} = \boldsymbol{r}_{n+1}(t^i) - \boldsymbol{r}_{n+1}(t^{i+1}) = \boldsymbol{r}_{n+1}(\boldsymbol{\beta}^i) - \boldsymbol{r}_{n+1}(\boldsymbol{\beta}^{i+1}) \tag{16}$$

With eqs. (15), (16) the components of the error-vector of the world coordinates dz, eq. (3), are defined.

Calculation of the Jacobian

The mxn Jacobian matrix $\boldsymbol{J}$ defines the relation between the world velocity $\dot{z}$ and the joint velocity $\dot{\boldsymbol{\beta}}$

$$\dot{z} = \begin{bmatrix} v \\ \boldsymbol{\omega} \end{bmatrix} = \boldsymbol{J}(\boldsymbol{\beta})\,\dot{\boldsymbol{\beta}}, \qquad (17)$$

where $\boldsymbol{v}$ and $\boldsymbol{\omega}$ are the desired translational and angular endeffector velocities, and

$$\boldsymbol{J}(\boldsymbol{\beta}) = \frac{\partial f}{\partial \boldsymbol{\beta}} \qquad (18)$$

is the Jacobian matrix. An effective calculation of $\boldsymbol{J}$ for a system consisting of a topological chain interconnected with revolute joints is given by[3]

$$\boldsymbol{J} = \begin{bmatrix} \tilde{\boldsymbol{u}}_1 \boldsymbol{r}_{1,n+1} & \tilde{\boldsymbol{u}}_2 \boldsymbol{r}_{2,n+1} & \dots & \tilde{\boldsymbol{u}}_n \boldsymbol{r}_{n,n+1} \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n \end{bmatrix} \qquad (19)$$

Here $\boldsymbol{u}_j$ denotes the unit vector along the joint axis j. The relative vectors $\boldsymbol{r}_{j,n+1}$, $j=1(1)n$, are defined as the vectors from the joint coordinate frame j to the endeffector frame $n+1$. The rank k of the Jacobian in cartesian space $m=6$ is

$$k = \text{rank}\,(\boldsymbol{J}) \le 6. \qquad (20)$$

Inverse kinematic

For the projection of the change in world coordinates dz onto the change of joint coordinates $d\boldsymbol{\beta}$ the relationship between world velocities $\dot{z}$ and joint velocities $\dot{\boldsymbol{\beta}}$ is used. The first order linearization of eq. (2) yields the desired relation between dz and $d\boldsymbol{\beta}$:

$$dz = \boldsymbol{J}d\boldsymbol{\beta}. \qquad (21)$$

Solving the under-determined linear system of eq. (21) yields the inverse kinematic problem

$$d\boldsymbol{\beta} = \boldsymbol{J}^{+}dz, \qquad (22)$$

where the superscript + denotes a generalized inverse. The nxm matrix $\boldsymbol{J}^{+}$ is called a Moore Penrose inverse if the four Moore Penrose conditions[6] are satisfied. For the particular row regular case ($n \ge k=6$) in eq. (21), $\boldsymbol{J}^{+}$ can be expressed as

$$\boldsymbol{J}^{+} = \boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{J}^{T})^{-1}. \qquad (23)$$

The general solution of eq. (21) is given by an inhomogeneous solution $d\boldsymbol{\beta}_p$ and an homogeneous solution $d\boldsymbol{\beta}_h$, which is obtained by a projection of an arbitrary vector $\boldsymbol{x}_0 \in \mathrm{R}^n$ onto the nullspace of $\boldsymbol{J}$

$$d\boldsymbol{\beta} = \underbrace{\boldsymbol{J}^{+}dz}_{d\boldsymbol{\beta}_p} + \underbrace{(\boldsymbol{I}_n - \boldsymbol{J}^{+}\boldsymbol{J})\,\boldsymbol{x}_0}_{d\boldsymbol{\beta}_h}. \qquad (24)$$

Using only $d\boldsymbol{\beta}_p$ as the solution, eq. (22), a least-squares solution is obtained, whereas the general solution $d\boldsymbol{\beta}_p + d\boldsymbol{\beta}_h$ can be utilized to optimize external constraints.

Local Optimization

Determination of a manipulator configuration at time t^{i+1}, whilst considering external constraints, is done by local optimization around the known configuration at time t^{i}. The system of linear equations, eq. (21), is formulated as an optimization problem with a local objective function $F(d\boldsymbol{\beta})$

$$F(d\boldsymbol{\beta}) = \frac{1}{2}d\boldsymbol{\beta}^{T}\boldsymbol{Q}d\boldsymbol{\beta} + \boldsymbol{p}^{T}d\boldsymbol{\beta}, \qquad (25)$$

with $\boldsymbol{Q}$ as an appropriate nxn diagonal matrix and $\boldsymbol{p}$ as a $nx1$ vector:

$$\boldsymbol{Q} = \text{diag}\,[q_1, \dots, q_n], \qquad (26)$$

$$\boldsymbol{p} = [p_1, \dots, p_n]^{T}. \qquad (27)$$

Rearranging eq. (21) yields an implicit constraint on the optimization problem of eq. (25):

$$\boldsymbol{G}(d\boldsymbol{\beta}) = \boldsymbol{J}d\boldsymbol{\beta} - dz = \boldsymbol{0} \qquad (28)$$

Using the Lagrange-multiplier approach, the extended optimization function to be minimized can be stated as

$$\Phi(d\boldsymbol{\beta}) = F(d\boldsymbol{\beta}) + \boldsymbol{\lambda}^{T}\boldsymbol{G}(d\boldsymbol{\beta}) = \text{Min}. \qquad (29)$$

Minimizing eq. (29) subject to eq. (28) for $d\boldsymbol{\beta}$ yields

$$d\boldsymbol{\beta} = \boldsymbol{J}_Q^{+}dz + (\boldsymbol{I}_n - \boldsymbol{J}_Q^{+}\boldsymbol{J})\,(-\boldsymbol{Q}^{-1}\boldsymbol{p}), \qquad (30)$$

where $\boldsymbol{J}_Q^{+}$ is the so-called[7] *weighted pseudo inverse*

$$\boldsymbol{J}_Q^{+} = \boldsymbol{Q}^{-1}\boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{Q}^{-1}\boldsymbol{J}^{T})^{-1}. \qquad (31)$$

Examples of objective functions

From the computational point of view a Taylor expansion $F(\boldsymbol{\beta} + d\boldsymbol{\beta})$ of a global objective function $F(\boldsymbol{\beta})$ is very expensive, since partial derivatives of $F(\boldsymbol{\beta})$ have to be formed. For this reason an explicit objective function in the form of eq. (25) was set up. It was found, that this approach leads to an optimized movement of the manipulator and is computationally very efficient.

The first term of eq. (25), $^1\!/_2\, d\boldsymbol{\beta}^{T}\boldsymbol{Q}\,d\boldsymbol{\beta}$, weights the joint velocities, and the factors q_k may interpreted as damping coefficients. A possible choice for the selection of the coefficients is the distance between joint j and the endeffector, leading to a damping of the arm joint velocities.

The second term of eq. (25), $\boldsymbol{p}^{T}d\boldsymbol{\beta}$, weights the configuration of the manipulator. Through appropriate selection of the p_k the robot is forced to move towards previously calculated configurations. Physically, this term may be described as a spring which weights the difference between actual values $\boldsymbol{\beta}_a$ and desired values $\boldsymbol{\beta}_d$ of the joint coordinates. For this case the weighting vector $\boldsymbol{p}$ may be described as the normalized difference in joint coordinates

$$\boldsymbol{p} = \frac{(\boldsymbol{\beta}_a - \boldsymbol{\beta}_d)}{|\boldsymbol{\beta}_a - \boldsymbol{\beta}_d| + \varepsilon}. \qquad (32)$$

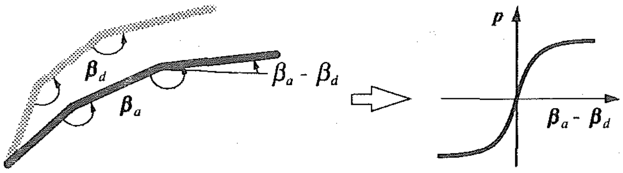


Figure 4. Example for weighting a desired configuration

In the kinematic controller the vector p performs tasks like collision avoidance and achieves a global optimized movement by using certain precomputed optimal configurations in workspace. This approach can further be utilized to achieve a conservative motion in joint space and to consider joint limitations. The latter constraint can be expressed by the vector p in form of penalty terms. The closer the actual joint coordinate to its actual limit is, the higher is the appropriate penalty coefficient in p.

Design of the kinematic controller

The scheme of the kinematic controller is depicted in fig. 5. World coordinates are calculated in the path planner and the orientation and position errors are calculated according to eqs. (15) and (16) respectively and may be scaled by gain matrices. The heart of the controller is the inverse kinematic transformation. The Jacobian is calculated according to eq. (19) and with the precomputed matrices Q and p the general solution $d\beta$ is determined according to eq. (24). Integrating the increments $d\beta$ yields the actual joint coordinates β^{i+1}, which are passed through a filter to the manipulator servos. Actual position and orientation is being computed by forward kinematics and both quantities are fed back separately to compare them with the new world coordinates w^{i+1}.

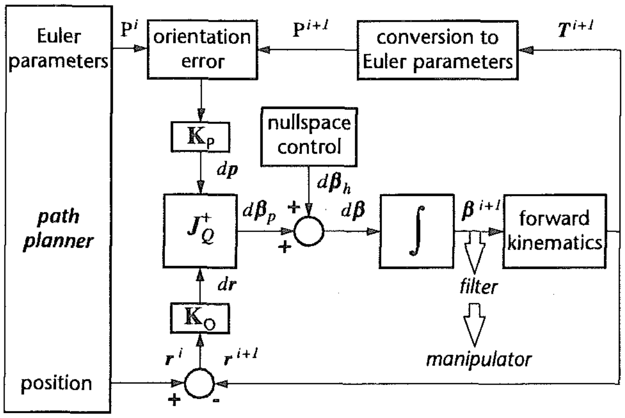


Figure 5. Closed loop kinematic controller

Numerics

The crucial point in realtime control, using eq. (24) as a general solution, is the calculation of the generalized inverse J^+. A particular problematic case arises when the rank of the mxn Jacobian J is less than m. The actual manipulator configuration in this case is either in a singular position or is close to such a singularity (numeric loss of rank).

If a matrix $J+E$, where E is small, has a rank greater than rank(J), then the generalized inverse differs significantly. In fact, at a singular configuration the generalized inverse is discontinuous, which again results in a discontinuous solution $d\beta$, eq. (24). A discontinuity in the joint coordinates can not be accepted, whereas an error in the computation of $J^+$and, by this, an error in the world coordinates can be tolerated, since the feedback loop will compensate this error.

As a conclusion the following rules for numerical methods for the computation of generalized inverses at realtime for control applications can be formulated:

- The *rank k* of the Jacobian has to be explicitly determined and has to be monitored to detect ill-conditioned Jacobians, which represent regions in the workspace close to a singular configuration.
- *Accuracy* of the solution is not that important in a feedback loop system.
- *Efficiency* of the computation algorithm, in terms of numbers of operations and required storage space, is a major requirement.

Having derived some guidelines to judge computational methods, we have now a closer look to the algorithms.

Keeping in mind that singularities in the workspace of redundant robots are rare and that for this reason the Jacobian has normally full rank $k = 6$ and is well-conditioned, a straight forward procedure is to use the

- direct calculation of the generalized inverse according to eq. (23).

Most computer algorithms solving linear systems are based on a Gauss elimination scheme. Methods based on the L-U factorization extend this to mxn matrices of rank $k \leq m$:

- Transformation on hermite normal form(6)
- Transformation on normal form(11)

Other algorithms are based on:(6)(11)

- Householder transformation
- Modified Gram-Schmidt orthogonalization
- Singular-value decomposition

Further have to be mentioned:

- Greville algorithm(1)
- Iterative algorithms
- An algorithm minimizing the residual error(5)

The last group of algorithms are very different from the decomposition or transformation techniques. The Greville algorithm is a recursive method with poor accuracy, iterative algorithms do not compute the rank explicitly, and the very last method introduces a damping factor which is difficult to choose(7) and leads to a computationally inefficient solution.

This list of algorithms is roughly sorted according to the number of operations required, except for the recursive and iterative methods. From the point of view of efficiency the direct calculation of J^+ is the most attractive one. Looking closer at this method, it is clear that the pivot-elements of the matrix inversion may be used to monitor the rank and that the term J^TJ of eq. (23) worsens the condition. This results in an error in the $d\boldsymbol{\beta}$, eq. (24). According to the above considerations about the computation of a generalized inverse for realtime control, this error can be tolerated as long as the discontinuity of the solution is minimized. Exactly this behaviour was found by comparing a solution $d\boldsymbol{\beta}$ computed by the direct calculation with a solution found by the singular value decomposition. For configurations close to a singular one, the direct solution deviates from the exact solution and converges towards the solution at the singular configuration. Furthermore, the discontinuity nearly vanishes. However, the direct calculation method fails for solving eq. (23) in case of rank $k < m$. An exception handling has to be implemented, when the direct solution method is used for computation of the generalized inverse. A possible strategy for the exception handling is the use of the generalized inverses calculated at time t^{i-1}.

Case studies

Two different case studies will be presented to demonstrate the behaviour of the presented kinematic controller. Both cases use a model of the mobile robot with 9 revolute joints. The first joint of the robot is perpendicular to the ground and denoted as joint 1. The following 5 joints of the robot-arm are parallel to the ground and the 3 last joints form a central wrist.

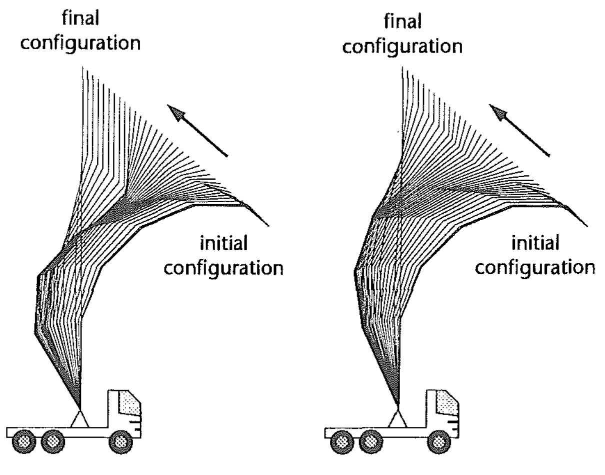
Control using only a pseudoinverse

The task for this case study was to demonstrate the different behaviour of the manipulator by using

(i) a pseudoinverse, eq. (23),

(ii) a weighted pseudoinverse, eq. (31),

in the homogeneous solution. The nullspace was not utilized.



(a) - case i *(b) - case ii*

Figure 6. Driving into a fully stretched configuration - pseudoinverse control

Fig. 6(a) shows the movement of the robot from a bow configuration into a fully stretched (singular) position using only the inhomogeneous solution with a pseudoinverse. It can be clearly seen that the pseudoinverse solution results in a movement where all joints are actuated; a solution of a least squares type is obtained. Fig. 6(b) demonstrates the same task using a weighted pseudoinverse instead. The weighting matrix $\boldsymbol{Q}$ was selected as

$$\boldsymbol{Q} = [25, 25, 18, 14, 10, 6, 2, 2, 2]^T.$$

Comparing the figs. 7 and 8, the scaling of the joint velocities by the weighting matrix $\boldsymbol{Q}$ is significantly.

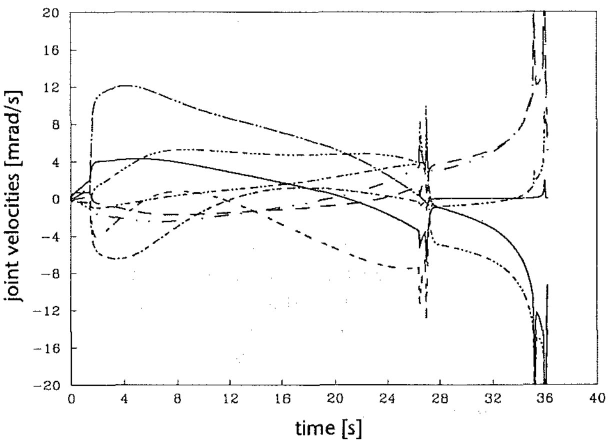


Figure 7. Joint velocities – case i

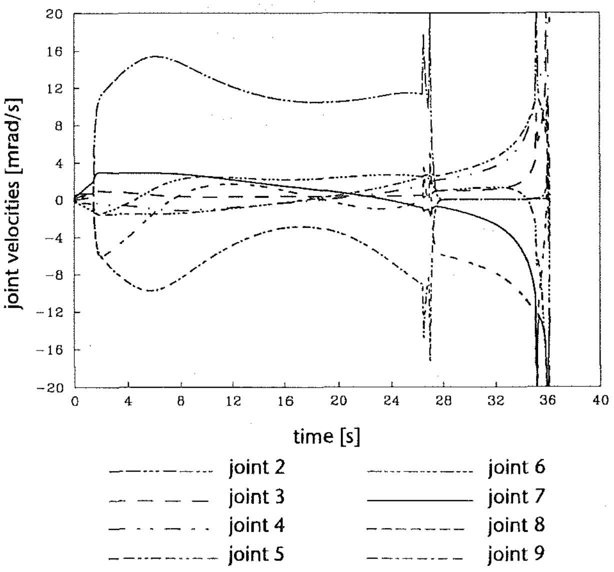


Figure 8. Joint velocities - case ii

Tracking a closed triangular trajectory

In this case study the influence of the nullspace solution is demonstrated. Two different controller were compared:

(i) pseudoinverse – no utilization of the nullspace

(ii) weighted pseudoinverse and utilization of the nullspace

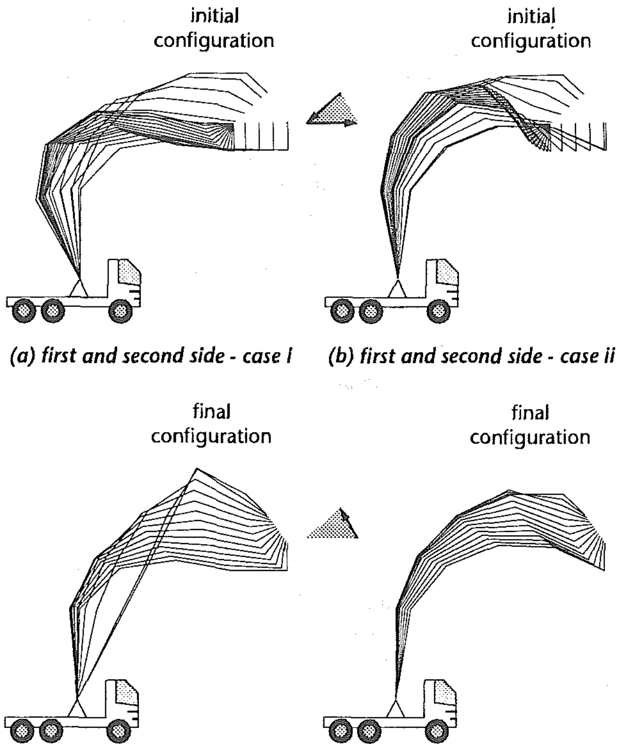
The $\boldsymbol{Q}$ matrix and the $\boldsymbol{p}$ vector of case (ii) were set up as:

$$\boldsymbol{Q} = [25, 25, 18, 14, 10, 6, 2, 2, 2\,]^{T},$$

$$\boldsymbol{p} = \frac{\boldsymbol{\beta} - \boldsymbol{\beta}_0}{|\boldsymbol{\beta} - \boldsymbol{\beta}_0| + 0.1},$$

where $\boldsymbol{\beta}_0$ are the joint coordinates of the initial configuration. The fig. 9(a) shows the tracking of the first two sides of the triangle with a pseudoinverse control, whereas the fig. 9(b) shows the same task utilizing the nullspace as well. The tracking of the last side of the triangle is depicted in figs. 9(c) and (d). Fig. 9(d) demonstrates the effect of the nullspace control. The manipulator reaches again the initial configuration, whereas with a solution based solely on pseudoinverse control a configuration far away from the initial one is reached, fig. 9(c). Furthermore this configuration can not be determined in advance.

Using the nullspace control of case (ii) with an appropriate choice of the weighting vector $\boldsymbol{p}$, the task of tracking a closed path in world coordinate space led to a conservative motion in joint space. Extending this approach to various desired configurations along the trajectory in the workspace, which are precomputed in the path optimizing module, an approximation of the global optimum may be achieved as well as collision avoidance.



(a) first and second side - case i *(b) first and second side - case ii*

(c) third side - case i *(d) third side - case ii*

Figure 9. Tracking a closed triangle - effect of nullspace control

Conclusion

This paper has presented an implementation of the resolved motion rate control for a redundant robot developed at the IPA research establishment. Special emphasis has been laid out on aspects concerning realtime computation of the generalized inverse. Between various methods the direct calculation has been identified as the most efficient and as the most suitable one for a feedback loop. Utilizing this method the discontinuity of the solution at a singular configuration can be minimized. The orientation error, a feedback in the control loop, has been described in 3-parametric form based on quaternions. External constraints, like obstacles or joint limitations, have been considered by an explicit, easy to set up, objective function using interpretative measures for the weighting matrices. Two case studies, using a model of the IPA robot with 9 joints, have been demonstrated the performance of this approach. The simulation of tracking a closed triangular path in task space has showed, that utilizing an appropriate objective function led to a conservative motion in joint space. The robot reached its initial configuration again, whereas with an inhomogeneous solution the final configuration could not be determined in advance.

Acknowledgment

The support of the graduate college *Discretization and Modelling for Continua and Flows* at the University of Stuttgart is gratefully acknowledged. The author would also like to thank C. Woernle and K. D. Rupp for many helpful discussions.

References

[1] Ben-Israel, A.; Greville, T. N. E.: *Generalized Inverses: Theory and Applications*, John Wiley & Sons, New York, 1972.

[2] Denavit, J.; Hartenberg, R. S.: *A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices*, ASME J. Appl. Mech. 22, 1955, p. 215 - 221.

[3] Hiller, M.; Kecskemethy, A.; Woernle, C.: *Computergestützte Kinematik und Dynamik für Fahrzeuge, Roboter und Mechanismen*, Fachgebiet Mechanik, Universität Duisburg, Institut A für Mechanik, Universität Stuttgart, 1989.

[4] Klein, C. A.; Huang, C. S.: *Review of Pseudoinverse Control for use with Kinematically Redundant Manipulators*, IEEE Trans. Systems, Man. and Cybernetics, 13, p 245 - 250, 1983.

[5] Kuhnert, F.: *Pseudoinverse Matrizen und die Methode der Regularisierung*, BSB B. G. Teubner Verlagsgesellschaft, Leipzig, 1976.

[6] Nash, Z. M. (Ed.): *Generalized Inverses and Applications*, Proceedings of an Advanced Seminar, Academic Press, New York, 1976.

[7] Nenchev, D. N.: *Redundancy Resolution through Local Optimization: A Review*, Journal of Robotic Systems, John Wiley & Sons, 1989, p 769 -798.

[8] Nikravesh, P. E.: *Computer Aided Analysis of Mechanical Systems*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[9] Wanner, M. C.; Engeln, W.; Rupp, K. D.: *Enabling Technologies for Large Manipulators*, Proceedings of the 8th international Symposium on Automation and Robotics in Construction, IPA, Stuttgart, 1991.

[10] Yuan, J. S. C.: *Closed Loop Manipulator Control Using Quaternion Feedback*, IEEE Journal of Robotics and Automation, Vol. 4, No. 4., 1988, p 434 - 440.

[11] Zielke, G.: *Verallgemeinerte inverse Matrizen*, Jahrbuch Überblicke Mathematik 1983, Bibliographisches Institut AG, 1983, S. 95 - 116.