

AN INVERSE KINEMATIC SOLUTION ALGORITHM FOR ROBOTS WITH TWO-BY-TWO INTERSECTING AXES AT THE END EFFECTOR

Lorenzo Sciavicco and Bruno Siciliano*

Universita' di Napoli, Dipartimento di Informatica e Sistemistica,
Via Claudio 21, 80125 Napoli, Italy.

ABSTRACT

One of the most important features of an advanced control system for articulated robots is the capability of transforming the work space coordinates, which naturally characterize any robot task, into the joint coordinates, on which control actions are developed (Inverse Kinematic Problem). While simple kinematical structures allow for closed form solutions, there is a class of robots for which this is not true. If the three axes of revolution at the end effector intersect two-by-two an exact solution seems not to exist.

The goal of the paper is to establish a fairly different solution algorithm, as compared to the trigonometric approach, which yields solutions in the above case. The algorithm is shown to be convergent along any trajectory. It proves very fast since it is based only on direct kinematics. Numerical examples are finally developed.

*Presently Visiting Scholar at Georgia Institute of Technology, George W. Woodruff School of Mechanical Engineering, Atlanta, Georgia 30332.

INTRODUCTION

The basis for all advanced robot control is the relationship between the Cartesian coordinates of the end effector, which naturally identify each task, and the joint coordinates of the manipulator. As a rule, the direct (joint-to-Cartesian space) relationship is unique, whereas the inverse (Cartesian-to-joint) is not. Actually, while there is only one end effector state for a given set of joint coordinates, there are a number of different joint configurations which all place the end effector in the same position and orientation. In many cases only one solution corresponding to a given kinematic configuration is desired, rather than the entire set of solutions. Usually the solution is to be implemented in real-time as it constitutes the servo loop reference; a minimum number of mathematical computations is then to be performed.

Typical six-degree-of-freedom nonredundant kinematical structures have three revolute joints at the end effector; it is the geometric parameters of such joints which determine the spatial configuration of the terminal axes of motion. Most of today's structures have a spherical wrist, i.e. three intersecting revolute joint axes; the application of the well-known trigonometric method, first proposed in [1], allows for closed form solu-

tions only for these simple structures [2]. In other cases, such as either two-by-two intersecting axes or nonconverging at all axes, an explicit solution seems not to be attainable in closed form. An iterative solution technique has been proposed in [3] for the class of two-by-two intersecting revolute joint axes, but involves an order of magnitude and more computations than a closed form solution, even if the trigonometric approach has still been used. The existence of an explicit solution to the kinematic equations for this class of manipulators is of great importance in evaluating the robot's suitability for computer control.

The goal of this paper is to demonstrate how a fairly different approach [4], as compared to the iterative-trigonometric one adopted in [3], can provide solutions for those structures with two-by-two intersecting revolute joint axes at the end effector [5]. The convergence of the resultant algorithm is proved by means of the Lyapunov direct method. Effectiveness of the proposed solution technique mainly lies in the fact that it only makes use of direct kinematics of the manipulator and the extra number of mathematical operations required is small, resulting thus in a contained computational burden. This issue favors the use of this inverse kinematic algorithm along any trajectory assigned in the task space, since the solution sample rates can be the same as those of the servo loops. In addition it will be shown how joint velocities can be directly generated at servo rate without any further computations; this is of a great deal of utility for those advanced control techniques, such as [6] for instance, which require reference joint velocities as well as joint variables. The same robot prototype as in [3] is taken as a reference in order to analyze the computational burden and develop a case study.

KINEMATICS

The kinematics of a serial link manipulator can be specified on the basis of the notations given in [7]. In order to identify uniquely the position and the orientation of the end effector, six degrees of freedom are needed; typical kinematically nonredundant structures have three revolute joints $(\theta_1, \theta_2, \theta_3)$, whereas the first three joints can be either all revolute, such as the PUMA arm [8], or two revolute and one prismatic, such as the Stanford arm [9]; hence in the following these joint variables will be denoted by (q_1, q_2, q_3) .

As far as the three revolute joint axes at the

end effector are concerned, three basic configurations are illustrated in fig. 1. The case a) is of particular interest since it becomes possible to decouple the position of the end effector from its orientation, that may be useful for control purposes. The cases b) and c) may also occur in practical robot designs. All such structures can be conveniently identified through the following constraints on the geometric parameters of the last three joints. More specifically, the lengths a_i and the distances d_n [7] are respectively in the three cases:

- a) concurrent axes: $a_4=a_5=d_5=0$ (fig. 1a),
- b) two-by-two intersecting axes: $a_4=a_5=0, d_5 \neq 0$ (fig. 1b),
- c) nonconverging axes: $a_4 \neq 0, a_5 \neq 0$ (fig. 1c).

As a consequence the knowledge of the geometric parameters of a kinematical structure allows for its classification in terms of the three structures illustrated above.

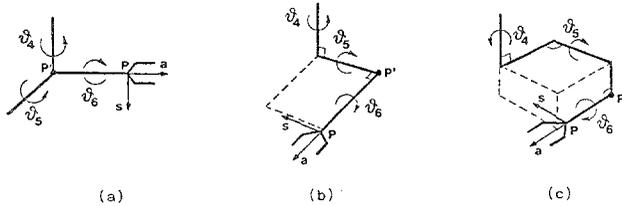


Fig. 1. The last three revolute axes at the end effector:

- a) concurrent, b) two-by-two intersecting, c) nonconverging.

On the other hand a robot task is naturally specified in terms of end effector Cartesian coordinates $(p_x, p_y, p_z, \alpha, \beta, \gamma)$ with respect to the base frame; p_x, p_y, p_z are the components of the end effector position vector \underline{p} , and α, β, γ are the Euler angles which define its orientation (roll, pitch and yaw angles can be adopted as well). The orientation, however, can be conveniently described through a unit approach vector \underline{a} , a unit sliding vector \underline{s} and a unit normal vector \underline{n} . The orientation frame $(\underline{s}, \underline{a}, \underline{n})$ defined with reference to the base frame of the manipulator, can be easily determined starting from the Euler angles [10]. Such frame will be referred to in the following since it allows for a unique definition of the orientation in terms of direct relationship with the joint variables. Under these assumptions, for any robot kinematical structure with known geometric parameters, the direct kinematics can be written as

$$\underline{p} = \underline{f}_p(\underline{q}), \quad \underline{s} = \underline{f}_s(\underline{q}), \quad \underline{a} = \underline{f}_a(\underline{q}) \quad (1)$$

where \underline{q} is the (6x1) vector of joint coordinates, and $\underline{f}_p, \underline{f}_s, \underline{f}_a$ are nonlinear vector functions which are always unique; $\underline{n} = \underline{f}_n(\underline{q})$ is redundant since it can be determined as the vector product $\underline{s} \times \underline{a}$.

Usually the end effector position vector \underline{p} and the approach unit vector \underline{a} are independent of the last rotation θ_6 . Hence as

$$\underline{p}' = \underline{p} - d_6 \underline{a}, \quad (2)$$

\underline{p}' can be assumed as position vector. Furthermore, depending upon the particular structure, it follows that such position vector \underline{p}' is only dependent on

- a) the first three joint variables (q_1, q_2, q_3) ,
- b) the first four joint variables $(q_1, q_2, q_3, \theta_4)$,
- c) the first five joint variables $(q_1, q_2, q_3, \theta_4, \theta_5)$

respectively in the three cases of fig. 1. The first case easily allows the proper definition of a "wrist" whose position depends on the first three d.o.f.'s, and a "hand" whose orientation depends on all d.o.f.'s. In the second case, which is of interest in this paper, four d.o.f.'s are disposable to position the vector \underline{p}' , since also θ_4 concurs to its determination. It is then possible to position \underline{p}' by ∞^1 values of $(q_1, q_2, q_3, \theta_4)$, unless one constraint is added in order to obtain a unique solution. In the last case there are five d.o.f.'s at disposal which involve ∞^2 values of $(q_1, q_2, q_3, \theta_4, \theta_5)$ to position \underline{p}' , except when two constraints are introduced.

THE SOLUTION ALGORITHM

The inverse kinematic problem is reconceived as a dynamical one in order to achieve a solution algorithm which only involves the computation of direct kinematics (1), [11], [13]. With reference to the kinematic notations previously introduced, a typical robot task in the Cartesian space can be assigned through the vectors $(\underline{\hat{p}}, \underline{\hat{s}}, \underline{\hat{a}})$. Let $\hat{\underline{q}}$ denote a solution of (1) relative to these vectors, and \underline{q} the algorithm current state variables. Accounting for the kinematic structures of fig. 1 leads to decompose the inverse kinematic problem into two stages; in particular, by working back from point P through the structure, it results convenient to partition the problem at that point P', dependent on a reduced number of joint variables, which can be still expressed in terms of the Cartesian coordinates of the assigned task. Correspondingly the vector \underline{q} can be partitioned as

$$\underline{q}^T = (\underline{q}_p^T \mid \underline{q}_h^T) \quad (3)$$

where $\underline{q}_p \in \mathbb{R}^3, \underline{q}_h \in \mathbb{R}^3$ in case of spherical wrist, $\underline{q}_p \in \mathbb{R}^4, \underline{q}_h \in \mathbb{R}^2$ if the axes intersect two-by-two, and $\underline{q}_p \in \mathbb{R}^5, \underline{q}_h \in \mathbb{R}$ if the axes do not converge at all. Such doing better copes with the actual kinematic structure and proves very useful as regards robot control. The case of spherical wrist has been widely treated in [4]. The aim of the work now is to establish a two-stage convergent algorithm to solve (1) in case of two-by-two intersecting axes at the end effector. For such structures direct kinematics (1) becomes

$$\underline{p}' = \underline{f}_p(\underline{q}_p), \quad \underline{s} = \underline{f}_s(\underline{q}), \quad \underline{a} = \underline{f}_a(\underline{q}_p, \theta_5). \quad (4)$$

First stage

Since the position vector \underline{p}' is determined through four joint variables, a constraint must be introduced in order to achieve a unique solution for \underline{q}_p . To this purpose, once the task has been assigned, i.e. $(\underline{\hat{p}}, \underline{\hat{s}}, \underline{\hat{a}})$, the first stage of the algorithm must guarantee not only that \underline{p}' coincides

with \hat{p}' but also that the fifth link is oriented in such a way to form an angle with the sixth link which coincides with the twist angle between the two links, α_5 , so as to be sure that the actual configuration is a feasible one as regards its constant geometric parameters. Such a constraint can be kinematically expressed by

$$\hat{a}^T z_4 = \cos \alpha_5 \quad (5)$$

where $\hat{a} = \hat{z}_5$ is given in the task space, and z_4 must be determined through $(q_1, q_2, q_3, \theta_4)$ in order to satisfy the above constraint (5).

Setting out a dynamic algorithm allows the definition of the following errors:

$$\begin{aligned} e_p &= \hat{p}' - f_p(q_p) \\ e_{z_4} &= \cos \alpha_5 - \hat{a}^T f_{z_4}(q_p), \quad q_p^T = (q_1 \ q_2 \ q_3 \ \theta_4) \end{aligned} \quad (6)$$

where $f_p(q_p)$ is the direct kinematic function which relates q_p to p . In order to assure the convergence of the state variables q_p to the desired ones \hat{q}_p , error dynamics is involved, i.e. in compact form

$$\begin{bmatrix} \dot{e}_p \\ \dot{e}_{z_4} \end{bmatrix} = \begin{bmatrix} \dot{\hat{p}}' \\ -\hat{a}^T f_{z_4} \end{bmatrix} - \begin{bmatrix} J_p \\ \hat{a}^T J_{z_4} \end{bmatrix} \dot{q}_p \quad (7)$$

where J_p is the (3x4) Jacobian matrix $\partial f_p / \partial q_p$ and similarly J_{z_4} is the (3x4) Jacobian matrix $\partial f_{z_4} / \partial q_p$. The point then is to relate \dot{q}_p to e_p, e_{z_4} so as to guarantee that such errors go asymptotically to zero, and consequently $q_p \rightarrow \hat{q}_p$. Let

$$V_p = .5(e_p^T e_p + e_{z_4}^2) \quad (8)$$

be a positive Lyapunov function associated with the above errors. Differentiating with respect to time and accounting for (7) yield

$$\dot{V}_p = (e_p^T \ e_{z_4}) \begin{bmatrix} \dot{\hat{p}}' \\ -\hat{a}^T f_{z_4} \end{bmatrix} - (e_p^T \ e_{z_4}) \begin{bmatrix} J_p \\ \hat{a}^T J_{z_4} \end{bmatrix} \dot{q}_p \quad (9)$$

It is to underline that the matrix premultiplied to \dot{q}_p in (9) has rank 4 almost everywhere, since 4 d.o.f.'s are needed to position the point P' and lay the fifth link on a surface. To be more specific (fig. 2), the first stage of the algorithm must assure not only that $p' = \hat{p}'$ but also that the fifth link, individuated by d_5 , must lie on the cone Γ of axis \hat{a} , angle α_5 and vertex P' ; in this way, however, a and \hat{a} do not have necessarily the same direction, since a is only guaranteed to lie on the cone Γ' of axis z_4 , same angle α_5 and vertex P' , so as evidenced in fig. 2. Nevertheless, in the second stage of the algorithm θ_5 will provide to align a with \hat{a} and θ_6 , which does not modify a , will take s over \hat{s} .

So far a first suitable choice for \dot{q}_p is

$$\dot{q}_p = \gamma_p \text{sgn}(J_p^T e_p + J_{z_4}^T \hat{a} e_{z_4}) \quad (10)$$

$$\gamma_p > (||\dot{\hat{p}}'||_{\max} + ||\dot{\hat{a}}||_{\max}) \left(\lambda \begin{bmatrix} J_p \\ \hat{a}^T J_{z_4} \end{bmatrix} \right)^{-1}$$

which assures that V_p be negative definite; $\text{sgnw} = (\text{sgnw}_1 \dots \text{sgnw}_n)$, with $w \in \mathbb{R}^n$, $\lambda(A)$ denotes the minimum eigenvalue of matrix A .

Provided that $e_p(t=0)=0$, $e_{z_4}(t=0)=0$, (10) always guarantees null tracking position errors but naturally introduces, in the neighborhood of $e_p=0$, $e_{z_4}=0$, an equivalent gain which tends to ∞ . This issue leads to the generation of a q_p rich of harmonics whose effect on q_p , however, is cut off by the filtering nature of the integrators which generate \dot{q}_p (see also fig. 3).

On the other hand, if the kinematical structure of the manipulator is not so complex as to be able to evaluate on-line the inverse of the matrix premultiplied to \dot{q}_p in a closed form, without excessive time expenditure, a rather different choice is

$$\dot{q}_p = \begin{bmatrix} J_p \\ \hat{a}^T J_{z_4} \end{bmatrix}^{-1} \begin{bmatrix} \dot{\hat{p}}' + M_p e_p \\ -\hat{a}^T f_{z_4} + m_{z_4} e_{z_4} \end{bmatrix} \quad (11)$$

which reduces \dot{V}_p to a negative quadratic form; the positive definite matrix M_p and the positive scalar m_{z_4} affect the convergence rate and are at designer's disposal. The choice (11), though it involves more computations, presents the inherent advantage of providing continuous joint velocities, as compared with the choice (10).

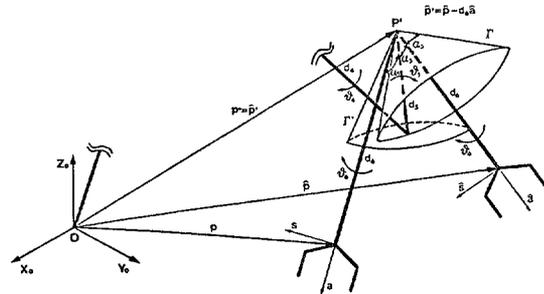


Fig. 2. Geometric task requirements for the first stage of the algorithm.

Second stage

In order to align a with \hat{a} and s with \hat{s} , the second stage of the algorithm must be able to determine θ_5 and θ_6 . Since both s and a are involved, it is natural to define the following errors:

$$e_s = \hat{s} - f_s(q) \quad (12)$$

$$e_a = \hat{a} - f_a(q_p, \theta_5),$$

where \hat{s}, \hat{a} are assigned in the task space and s, a are those in (4). Progressing as for the first stage gives error dynamics

$$\begin{bmatrix} \dot{e}_s \\ \dot{e}_a \end{bmatrix} = \begin{bmatrix} \dot{\hat{s}} \\ \dot{\hat{a}} \end{bmatrix} - \begin{bmatrix} J_{sp} \\ J_{ap} \end{bmatrix} \dot{q}_p - \begin{bmatrix} J_s \\ J_a \end{bmatrix} \dot{q}_h, \quad q_h^T = (\theta_5 \ \theta_6) \quad (13)$$

where J_{sp} and J_{ap} are the (3x4) Jacobian matrices $\partial f_s / \partial q_p$ and $\partial f_a / \partial q_p$, respectively, and similarly J_s and J_a are the (3x2) Jacobian matrices $\partial f_s / \partial q_h$ and

$\partial f / \partial q_h$ respectively. Thus \dot{q}_h must be related to \dot{e}_s, \dot{e}_a so as to assure that such errors go asymptotically to zero ($\underline{s}=\hat{\underline{s}}, \underline{a}=\hat{\underline{a}}$, and obviously $\underline{n}=\hat{\underline{n}}$) and consequently $\underline{q}_h = \hat{\underline{q}}_h$, where $\hat{\underline{q}}_h$ are the desired joint variable solutions, together with $\hat{\underline{q}}_p$, of (4). To this end let

$$V_h = .5(e_{s-s}^T e_{s-s} + e_{a-a}^T e_{a-a}) \quad (14)$$

be a positive Lyapunov function associated with the above errors. Differentiating with respect to time and accounting for (13) give

$$\dot{V}_h = -\underline{s}^T \dot{\hat{\underline{s}}} - \underline{a}^T \dot{\hat{\underline{a}}} - (\hat{\underline{s}}^T J_{sp} + \hat{\underline{a}}^T J_{ap}) \dot{q}_p + \quad (15)$$

$$-(\hat{\underline{s}}^T J_s + \hat{\underline{a}}^T J_a) \dot{q}_h$$

At this extent one must recall the following kinematic properties concerning with the unit vectors $\underline{s}, \underline{a}$ [4]:

- i) $\text{rank}(J_s) = \text{rank}(J_a) = 2 \quad \forall \underline{q}$ (16)
- ii) $N(J_s) = \text{span}(\underline{s}), N(J_a) = \text{span}(\underline{a}) \quad \forall \underline{q}$ (17)
- iii) given $\underline{x}, \underline{y} \in R^3, J_s^T \underline{x} + J_a^T \underline{y} = \underline{0}$ if $\underline{x} \in \text{span}(\underline{s}), \underline{y} \in \text{span}(\underline{a})$ (18)

where $N(A)$ denotes the null space of matrix A. By accounting for such properties and observing that other orientation singularities of (18) are of no interest for a convergent algorithm, in [4] it has been already proved that a suitable choice for \dot{q}_h results

$$\dot{q}_h = \gamma_h \text{sgn}(J_s^T \hat{\underline{s}} + J_a^T \hat{\underline{a}}), \quad (19)$$

$$\gamma_h > (\|\hat{\underline{s}}\|_{\max} + \|\hat{\underline{a}}\|_{\max} + \|\dot{\underline{q}}_p\|_{\max}) \cdot$$

$$\cdot [\Lambda (J_{sp}^T J_{sp} + J_{ap}^T J_{ap})]^{.5} [\lambda (J_s^T J_s + J_a^T J_a)]^{-.5}$$

where $\Lambda(A)$ denotes the maximum eigenvalue of matrix A. It is to underline that (19) suffers from the same problems concerned with sgn type laws in (10).

Remarks

In sum, realizing the above two stage algorithm always assures the state variables \underline{q} converge to $\hat{\underline{q}}$, thus performing the inverse kinematics required. Since the number of mathematical computations is easily seen to be contained, the application of such an algorithm along a prespecified trajectory in the task space looks attractive; from the implementation standpoint the solution sample rates can be conveniently increased up to the same values of joint servo sample rates. A digital implementation by means of a single dedicated microprocessor system, even if for the spherical wrist case, has already been realized in lab, and fully described in [12].

Nevertheless, as the algorithm provides at each step a solution which is adjacent to the preceding one, uniqueness of the solution is automatically assured. Starting with the same initial conditions $\underline{q}(0)=\hat{\underline{q}}(0)$, moreover, avoids any problem of indeterminacy concerned with $\cos \alpha_5 =$

$\cos(-\alpha_5)$ and the orientation singularities of (18), [4].

It must be emphasized also that the algorithm can directly generate joint velocities \dot{q} corresponding to the assigned task space velocities $\dot{\underline{p}}, \dot{\underline{s}}, \dot{\underline{a}}$, so as required by advanced control techniques, such as [6] for instance. To this goal, in order to avoid joint velocities be rich of harmonics, as previously illustrated, one must give up null tracking errors and accept reasonably small errors by replacing the sgn type laws in (10) and (19) respectively by the proportional type laws

$$\dot{q}_p = \gamma_p (J_{p-p}^T e_p + J_{z4}^T \hat{a} e_{z4}), \quad (20)$$

$$\dot{q}_h = \gamma_h (J_s^T \hat{s} + J_a^T \hat{a}). \quad (21)$$

With such choices, whose block diagram schemes are shown in figg. 3a and 3b respectively, V_p and V_h result negative only outside a region in the proper error spaces which contains the origin, thus assuring limited tracking errors. The maximum tracking errors will depend directly on task velocities and inversely on feedback gains; it must be underscored that the steady-state ($\dot{\underline{p}}=0, \dot{\underline{s}}=0, \dot{\underline{a}}=0$) errors are identically zero. It is these laws (20) and (21) which will be adopted in the case study.

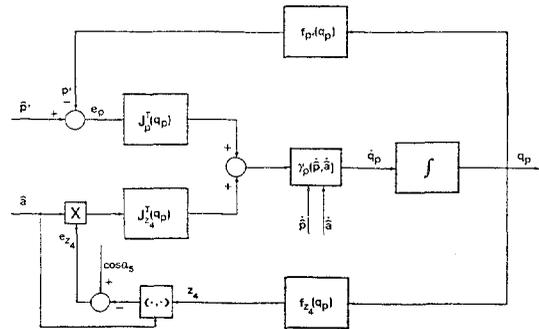


Fig. 3a. Inverse Kinematic Scheme: first stage.

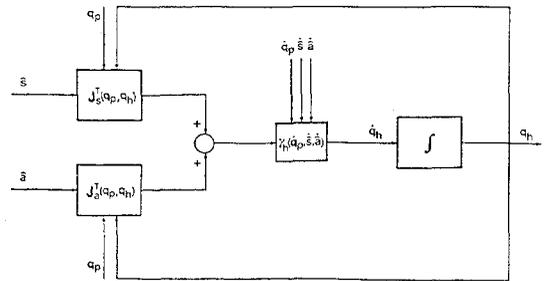


Fig. 3b. Inverse Kinematic Scheme: second stage.

A CASE STUDY

The robot prototype of fig. 4, [3], has been selected as a reference to develop a case study for the inverse kinematic algorithm set forth in this paper, as compared to the iterative procedure proposed in [3].

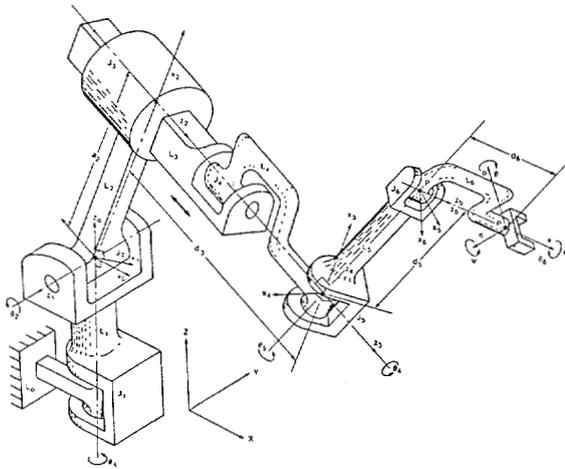


Fig. 4. The prototype arm reported in [3].

Digital implementation

Kinematics of the manipulator of fig. 4, as regards the direct functions which are to be evaluated at each step of the algorithm just presented, are not reported here for brevity and can be found in [14].

The iterative procedure proposed in [3] is said to converge, only from the experimental standpoint, within four to five iterations, under the assumptions to admit average orientation errors be low $.1^\circ$ and to be at a sufficient distance from a point of degeneracy for the kinematical structure. Adopting such solution requires 8 transcendental function calls, 41 floating point multiplies, 24 additions, 3 square roots, along with 7 two argument arctangent function calls, for each iteration. Hence the average number of mathematical computations required makes this technique impractical for performing, at the same servo control rate, the inverse kinematics along a trajectory given in the task space, unless interpolation between a certain number of via points, obtained at a lower solution sample rate, is provided. Nevertheless, since the number of iterations cannot be a priori fixed, the worst case solution period must be chosen as regards control purposes; the farther the via points in the joint space are each other, however, the larger the errors with respect to the exact joint variables are. If joint velocities are needed too, an even more conspicuous number of mathematical computations are likely to be involved.

On the other hand the algorithm described in this paper overtakes most of the above drawbacks: a moderate number of computations are required (10 transcendental function calls, 112 floating point multiplies, 58 additions), no problem of solution nonuniqueness arises since the algorithm starts with the same initial conditions on the state variables q and progresses with continuity along a trajectory, tracking errors are very small while steady-state errors are practically zero, as it will be shown later.

A further remark is to be made about the task trajectory planning. If the desired trajectory passes in the proximity of a singular point, the algorithm obviously involves large tracking errors around such point, even if it converges at steady-state; this is not surprising since the actual robot trajectory involves very high velocities. The attempt of increasing the feedback gains in (20), (21) so as to counteract the terms related to the above velocities, indeed, can present some difficulties, as far as the digital implementation of the algorithm is concerned, unless such trajectory is slow enough. On the other hand, if the planned trajectory crosses a point of singularity, the algorithm (20), (21) always works, since it does not require any function inversion (Jacobian, etc.).

It is also remarkable that the tracking errors illustrated in the following numerical examples are obtained with one iteration of the algorithm so as to save computation time; in case of trajectories which pass by a singular point, a sufficient number of iterations would allow lower tracking errors.

Last but not least, besides joint coordinates, even joint velocities are directly generated by the algorithm, without requiring any further computation, that is a good optional for tracking control.

Numerical examples

In order to show the effectiveness of the proposed algorithm, two numerical examples have been simulated for the robot prototype of fig. 4. The desired trajectories to track in the Cartesian space are illustrated in fig. 5.

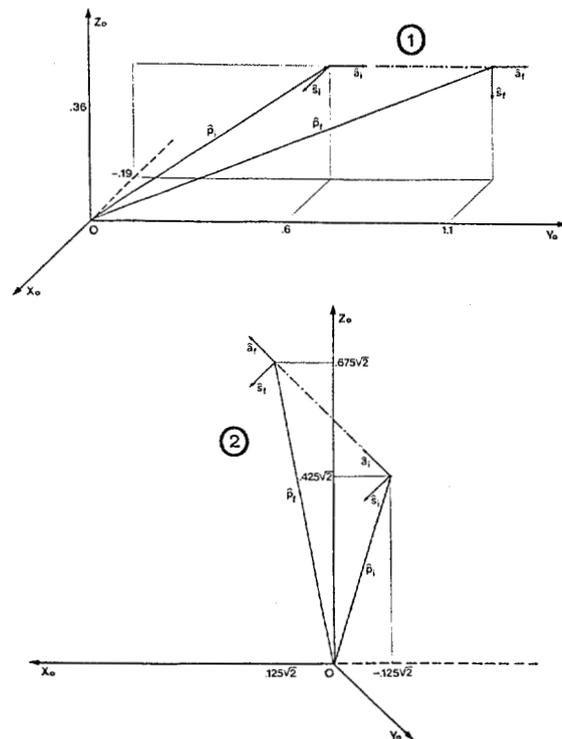


Fig. 5. Task trajectories to be tracked.

Trapezoidal velocity profiles have been imposed both for the position vector and for the Euler angles of orientation of the end effector. Tab. 1 shows the maximum velocities and time intervals for each trajectory.

Tab. 1. Parameters for the velocity profiles.

Trajectory	t_f [s]	$\ \dot{\hat{p}}\ _{\max}$ [m/s]	$\dot{\theta}_{\max}$ [°/s]	$\dot{\phi}_{\max}$ [°/s]	$\dot{\psi}_{\max}$ [°/s]
2	1	.75	0	0	0
	5	.15	0	0	0
1	1	.75	0	135	0
	5	.15	0	27	0

The proportional type solutions (20), (21) have been adopted, with the inherent advantage of directly generating joint velocities. The computational burden has suggested a solution sample period of 2 ms., on condition to use a single dedicated microprocessor with floating point unit, [12]. The gains in (20), (21) at designer's disposal have been set up at: $\gamma = \gamma_1 = 500$.

Due to lack of space tracking errors are not shown here, but they can be found in [14]. Position errors have resulted below 1 mm. and orientation errors below .15° for both trajectories. At steady-state, however, they vanish in virtue of the closed loop structure of the developed inverse kinematic algorithm. For the second trajectory, in particular, simulation results showed that no problem arises at the singularity ($\hat{p}' = \hat{p}' = 0$) so as it had been anticipated. Tracking errors, finally, increase as velocities increase; this issue actually matches with practice, since robot tracking performance is likely to result higher in working tasks than in handling tasks.

CONCLUSIONS

This paper has presented a solution algorithm for the inverse kinematic problem for robotic manipulators whose three end effector revolute joint axes intersect two-by-two. By working back from the end effector through the intermediate links, the algorithm has been partitioned at an opportune point whose position, dependent on a reduced number of joint variables, can be expressed in terms of the Cartesian position and orientation coordinates of the required task. In this way a first stage provides the joint coordinates which determine the position of the above point along with a feasible direction of the fifth link, while the second stage yields the remaining joint coordinates which align the orientation unit vectors. It must be underlined that such technique can also be applied to the case of nonconverging axes, see [5] for further details, still by working back and accounting for adequate mechanical and geometrical constraints.

The occurrence of singular solutions can be prevented by starting with initial conditions on the joint variables congruent with the initial Cartesian position and orientation of the assigned task trajectories; as the numerical algorithm implemented works with continuity along the trajectory, adjacent solutions in the joint space are assu-

red. The computational burden is contained, allowing for solution sample rates equal to those of joint servos, thus avoiding interpolation. In short the algorithm presented seems to be preferable to iterative-trigonometric techniques as far as computation time, occurrence of singularities and need for joint velocities are concerned.

Future developments are devoted to extend this dynamical approach for the inverse kinematics to kinematically redundant manipulators which seem to show potential advantages over current robot designs.

REFERENCES

- [1] R.P. Paul, B. Shimano and G.E. Mayer, "Kinematic control equations for simple manipulators," *IEEE SMC*, vol.11, no.6, June 1981.
- [2] D.L. Pieper, "The kinematics of manipulators under computer control," Stanford A.I. Project, Memo. AIM-72, Oct. 1968.
- [3] V.I. Lumelsky, "Iterative coordinate transformation procedure for one class of robots," *IEEE SMC*, vol.14, no.3, May/June 1984.
- [4] A. Balestrino, G. De Maria, L. Sciavicco and B. Siciliano, "A novel approach to coordinate transformation for robotic manipulators," subm. to *IEEE SMC*, 1985.
- [5] G. De Maria, L. Sciavicco and B. Siciliano, "A general solution algorithm to coordinate transformation for robotic manipulators," *Proc. '85 ICAR*, Tokyo, Japan, Sept. 1985.
- [6] A. Balestrino, G. De Maria and L. Sciavicco, "An adaptive model following control for robotic manipulators," *ASME DSC*, vol.105, Sept. 1983.
- [7] J. Denavit and R.S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, June 1955.
- [8] C.S.G. Lee, "Robot arm kinematics, dynamics, and control," *IEEE Computer*, vol.15, no.12, Oct. 1982.
- [9] V.D. Scheinman, "Design of a computer manipulator," Stanford A.I. Lab, Memo. AIM-92, 1969.
- [10] J.Y.S. Luh, "An anatomy of industrial robots and their controls," *IEEE AC*, vol.28, Feb. 1983.
- [11] A. Balestrino, G. De Maria and L. Sciavicco, "Robust control of robotic manipulators," *Proc. 9th IFAC World Congress*, Budapest, Hungary, July 1984.
- [12] G. De Maria and R. Marino, "A discrete algorithm for solving the inverse kinematic problem for robotic manipulators," *Proc. '85 ICAR*, Tokyo, Japan, Sept. 1985.
- [13] W.A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," *Proc. 23rd IEEE CDC*, Las Vegas NV, Dec. 1984.
- [14] L. Sciavicco and B. Siciliano, "Coordinate transformation: a solution algorithm for one class of robots," subm. to *IEEE SMC*, 1985.