Task-priority formulations for the kinematic control of highly redundant articulated structures

Conference Paper · November 1998

DOI: 10.1109/IROS.1998.724639 · Source: IEEE Xplore

2 authors:

P. Baerlocher

10 PUBLICATIONS **627** CITATIONS

SEE PROFILE

Ronan Boulic

École Polytechnique Fédérale de Lausanne

241 PUBLICATIONS **4,346** CITATIONS

SEE PROFILE

Task-Priority Formulations for the Kinematic Control of Highly Redundant Articulated Structures

P. Baerlocher, R. Boulic
EPFL - DI - LIG

{baerloch | boulic}@lig.di.epfl.ch

Abstract

In this paper we compare two formulations for the kinematic control of redundant manipulators, based on task prioritization. We also propose an incremental method to speed up the evaluation of the solution. This is especially interesting when dealing with multiple task-priority levels and with highly redundant structures. Finally, the effectiveness of the approach in the context of posture control of human-like articulated figures is illustrated.

1. Introduction

The complexity of the human kinematic control - with its large number of DOFs, its tree structure and its intrinsic unstable equilibrium - requires an efficient approach for multiple tasks prioritization. Numerous end-effectors and the center of mass generally need to be controlled simultaneously to achieve even the simplest goal of reaching an object in space. Their control need a clear prioritization as the associated tasks have different safety levels (e.g. : keeping balance is more important than reaching an object). In the present paper we address the problem of kinematic control with prioritization for an articulated structure and we illustrate it for the human case.

The plan is as follows : first, the general problem is stated in Section 2, then the state of the art is reviewed in Section 3, and useful mathematical properties are recalled in Section 4. Two existing formulations and their extensions to multiple tasks are presented in Sections 5 and 6. An incremental method is proposed in Section 5 to speed up the evaluation, especially when handling many tasks. We compare the behavior and computational cost of both formulations in Section 7, and present an application of task-priority control to human posture synthesis in Section 8. We finally summarize the results in Section 9.

2. Statement of the problem

Let q be the joint coordinate vector of an n-DOFs articulated structure. The kinematic behavior can be completely specified through *Inverse Kinematics* with :

- an arbitrary number, say t, of tasks (T_1...T_t), with an order of priority (T_i has priority over T_{i+1}). A task controls location and/or orientation of one or more end-effectors simultaneously, and is defined by an m_i-dimensional velocity vector in task-space $\dot{x}_i$ as well as an $m_i$xn Jacobian matrix J_i (relating unit joint velocities to the end-effector(s) task velocity);
- an optimization vector in joint space, called $\dot{q}_{opt}$; by setting $\dot{q}_{opt} = \alpha \nabla C(q)$, it can be used to optimize a desired criterion expressed by a scalar function $C(q)$.

The problem is to find a joint velocity vector $\dot{q}$ such that every kinematically achievable task T_i is satisfied, unless it conflicts with a higher-priority task T_j (j < i); in this case, or if T_i is not kinematically achievable, its error has to be minimized. The joint space optimization has to be applied without perturbing the tasks (it is of lowest priority).

3. Review of solutions

Following the steps from the simplest inverse kinematic problem to the more general one, we start with a single task, and search for a solution $\dot{q}$ to the kinematic equation :

$$\dot{x} = J\dot{q} \tag{1}$$

If J^{-1} exists, there is one exact solution to (1), but in the general case there may be an infinity of solutions or no solution at all. The best solution to (1) is then given by the least-squares inverse (for a detailed review, see [11]), in the sense that it minimizes the residual norm $\|J\dot{q} - \dot{x}\|$ as well as the solution norm $\|\dot{q}\|$:

$$\dot{q} = J^{\dagger}\dot{x} \tag{2}$$

In redundant cases (m<n), a more general solution to (1) projects an arbitrary vector *z* (in joint velocity space) onto the null-space of J, allowing to select one of the many solutions:

$$\dot{q} = J^{\dagger}\dot{x} + (I - J^{\dagger}J)z \qquad (3)$$

The vector *z* can be exploited to optimize any desired criterion, without affecting the task : numerous propositions have been made as obstacle avoidance [9, 8], joint limits avoidance and singularity avoidance. In [6] multiple normalized criteria expressed in joint space are mixed together. Besides, enforcing multiple cartesian tasks may be achieved by integrating them in a single *augmented* Jacobian; as a consequence, when conflicts between tasks arise, a compromise is found among all tasks as they implicitly have the same priority, but none is fulfilled. For highly redundant manipulators that can achieve many tasks simultaneously, it is interesting to have some priority levels, so that when there is conflict, one is favoured over the others. This method has been proposed in [9] for two tasks, then used for control with collision avoidance [12, 8], and extended to any number of tasks in [16]. Recently, another formulation for task-priority control has been introduced in [4, 5].

Before discussing further these formulations, let us briefly stress the singularities problem. Near kinematic singularities, like at full arm extension, the solution norm grows to infinity. To limit the problem, numerous authors [17, 15, 13] have used a *regularization* technique called *damped least-squares*, that bounds the solution norm but also introduces some tracking error. Thus a compromise between accuracy and solution norm has to be made (accuracy can be improved by using more sophisticated methods like numerical filtering [13]).

4. Background on inverses

This section recalls the elements that are fundamental for the discussion of the task-priority formulations (for more details, see [3, 15] for example).

A generalized inverse of a matrix A is used when its inverse A^{-1} does not exist, that is when A is singular or rectangular. It is used here to find a solution $\dot{q}$ to $\dot{x} = A\dot{q}$.

The least-squares (LS-) inverse (or *pseudoinverse*) :

The properties that define the LS-inverse $A^{\dagger}$ are :

$$AA^{\dagger}A = A \qquad (4) \qquad A^{\dagger}AA^{\dagger} = A^{\dagger} \qquad (5)$$

$$(AA^{\dagger})^{*} = AA^{\dagger} \qquad (6) \qquad (A^{\dagger}A)^{*} = A^{\dagger}A \qquad (7)$$

where the star symbol denotes the matrix transpose. Other useful properties are :

$$A^{\dagger\dagger} = A \qquad (8) \qquad A^{\dagger *} = A^{*\dagger} \qquad (9)$$

and when B is idempotent and Hermitian [12] :

$$B(AB)^{\dagger} = (AB)^{\dagger}. \qquad (10)$$

However, a property of the usual inverse does not hold for the LS-inverse : $(AB)^{\dagger} \neq B^{\dagger}A^{\dagger}$. (11)

We often use the null-space projector $P = I - A^{\dagger}A$; it can be shown that $P^2 = P$, $P^{*} = P$ and $P^{\dagger} = P$, and that two arbitrary projectors do not commute : $P_A P_B \neq P_B P_A$.

The LS-inverse solution suffers of discontinuity at the transition between singular and non-singular configurations, and is of arbitrarily high norm in the singularity neighborhood. To alleviate this behavior, the damped least-squares inverse is used instead, and detailed hereafter.

The damped least-squares (DLS-) inverse :

It is defined by : $A^{\dagger_\lambda} = A^{*}(AA^{*} + \lambda^2 I)^{-1}$, where λ is a positive scalar called *damping factor*, the limit of $A^{\dagger_\lambda}$ being $A^{\dagger}$ when λ tends to 0. The solution $\dot{q} = A^{\dagger_\lambda}\dot{x}$ minimizes $\|A\dot{q} - \dot{x}\|^2 + \lambda^2\|\dot{q}\|^2$; λ can be chosen so that the solution norm stays within a given range [13]. Also the DLS-inverse solution is continuous.

Care must be taken when working with DLS-inverses : it is easily verified that while some LS-inverse properties still hold (6, 7, 9 and 10), others do not (4, 5 and 8); thus the DLS-inverse is not a generalized inverse. A consequence is that we cannot simply replace every LS-inverse by a DLS-inverse. For example, take the general solution (3) : the second term has no impact on the primary task achievement thanks to property (4), i.e. $J[(I - J^{\dagger}J)z] = 0, \forall z$. Because (4) does not hold for a DLS-inverse, $J[(I - J^{\dagger_\lambda}J)z] \neq 0$, leading to a priority violation. Thus damping should not be used in the projectors.

5. The first task-priority formulation (F_1)

This formulation was first proposed by [9], then simplified and applied to moving obstacles avoidance by [12]; it has been studied by a number of authors [7, 5, 10].

Given two tasks $T_1 = (J_1, \dot{x}_1)$ and $T_2 = (J_2, \dot{x}_2)$, where T_1 has priority over T_2, the solution is :

$$\dot{q} = J_1^{\dagger}\dot{x}_1 + [J_2(I - J_1^{\dagger}J_1)]^{\dagger}(\dot{x}_2 - J_2(J_1^{\dagger}\dot{x}_1)) \qquad (12)$$

The derivation of (12) is done in [9, 12] and is recalled now. First, one finds the set of solutions that satisfy the pri-

mary task : $\dot{q} = J_1^{\dagger}\dot{x}_1 + (I - J_1^{\dagger}J_1)z$, where z is an arbitrary vector. Then, this solution is substituted into $J_2\dot{q} = \dot{x}_2$, which is the equation we would like to see approached as much as possible without disturbing the primary task. This results in $J_2J_1^{\dagger}\dot{x}_1 + J_2(I - J_1^{\dagger}J_1)z = \dot{x}_2$; solving for z with the pseudoinverse yields $z = [J_2(I - J_1^{\dagger}J_1)]^{\dagger}(\dot{x}_2 - J_2J_1^{\dagger}\dot{x}_1)$ which can be substituted back, to finally obtain eq. (12), using property (10) to simplify the result.

The difficult point of this formulation is the need to compute an inverse of $J_2(I - J_1^{\dagger}J_1)$. This matrix gives the available range for the secondary task to perform without affecting the primary task. When the secondary task cannot be fulfilled without preventing the primary task from being realized, this matrix becomes singular, thus generating the same problems occurring at kinematic singularities, i.e. high joint velocities and oscillations. This kind of singularity has been termed *algorithmic singularity* [1]. Experience shows that close to such singularities the priority levels may even be inverted (as also noted in [5]); still, these singularities cannot be avoided and have to be handled. Once again, the DLS-inverse can be used in place of LS-inverse, but then the secondary task tracking is less accurate.

Extension to three or more tasks (priority levels) :

A first approach to extend eq. (12) would be to apply the method described in [9, 12] once for each task. But this leads to complicated intermediate equations, and it can be shown that the final expression is equivalent to the one given in [16], and that is discussed now.

Siciliano and Slotine [16] proposed a recursive extension allowing an arbitrary number of tasks (t tasks). We recall it there, using our slightly different notation :

$$\begin{aligned} \dot{q}_i &= \dot{q}_{i-1} + (J_iP_{i-1}^A)^{\dagger_{\lambda_i}}(\dot{x}_i - J_i\dot{q}_{i-1}) \\ \dot{q} &= \dot{q}_t \end{aligned} \qquad (13)$$

where

$$P_i^A = I - J_i^{A\dagger}J_i^A \qquad (14)$$

is the projector[1] onto the null-space of the augmented Jacobian

$$J_i^A = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_i \end{bmatrix}.$$

1.There is no damping in the definition (Cf. last note in Sect. 4).

The recursion stops with $P_0^A = I$ and $\dot{q}_0 = 0$. When $t = 2$, eq. (13) is equivalent to eq. (12) (with damping).

The mapping of $J_iP_{i-1}^A$ is the subspace allowed to task T_i to perform without affecting higher priority tasks.

Although eq. (13) is in a recursive form, the projector P_i^A is not, and has to be computed from scratch again for every task. As the augmented Jacobian is growing for every new task, this can be computationally heavy, especially because this is the most complex sub-expression of eq. (13).

Incremental computation of projector P_i^A :

To speed up the algorithm, we propose a recursive formula for the computation of the projector :

$$\begin{aligned} P_i^A &= P_{i-1}^A - (J_iP_{i-1}^A)^{\dagger}(J_iP_{i-1}^A) \\ P_0^A &= I \end{aligned} \qquad (15)$$

This result is analog to the simple formula derived from the iterative method of Greville [3 : App. 1.2.], that allows to compute a projector of a matrix, working out a row (denoted j_i) of the matrix at a time :

$$P_i^A = \begin{cases} P_{i-1}^A - \dfrac{P_{i-1}^A j_i^T j_i P_{i-1}^A}{\left\| P_{i-1}^A j_i^T \right\|^2} & \text{if } (j_iP_{i-1}^A) \neq 0 \\ P_{i-1}^A & \text{otherwise} \end{cases} \qquad (16)$$

The difference is that we introduce a sub-matrix (J_i) at each iteration step, instead of a vector. This result is based on a formula (due to R.Cline) that gives the pseudoinverse of a partitioned matrix, which is applied to the incremental computation of a projector in [3 : Section 6.1.1.].

In fact, eq. (15) is not faster than eq. (16) : this is due to the presence of a pseudoinverse in (15), while (16) exploits the very simple expression of the pseudoinverse of a vector[2]. Still, eq. (15) is worth using here, because the terms $(J_iP_{i-1}^A)$ and $(J_iP_{i-1}^A)^{\dagger}$ have already been computed at the previous priority level, in eq. (13).

To be more precise, a DLS-inverse is computed, while a LS-inverse is required in (15), but it is possible to compute both at the same time for little more than the cost of a single inversion (for example, the SVD [14] is the same for both inversions).

The speed-up obtained with the recursive formula (15) compared to (14) is now briefly analysed. For the sake of simplicity, we assume that all Jacobians J_i have the same

2. $v^{\dagger} = v^T/(v^Tv)$ if $v \neq 0$; 0 otherwise

size (mxn). Thus, when handling the i^{th} task, the "absolute" method (14) deals with an augmented matrix of size (im)xn, while the incremental method (15) deals with a matrix of constant size mxn. This matrix is inverted; we found experimentally that the time required to perform an inverse of an mxn matrix is ~ mn^2 (we are using an SVD). Thus, on the whole process (for t tasks), the speed-up is :

$$\frac{\sum_{i=1}^{t} (im)n^2}{\sum_{i=1}^{t} mn^2} = \frac{\frac{t(t+1)}{2}mn^2}{tmn^2} = \frac{t+1}{2}$$

In short, eq. (15) allows a very simple incremental computation of the null-space projector, with a speed-up factor growing linearly with the number of tasks.

To conclude this section, we put eq. (13) in a different form, to ease the comparison with the second formulation; the general expression for t tasks with the joint optimization term $\dot{q}_{opt}$ becomes :

$$\begin{aligned} \dot{m}_i &= (J_i P^A_{i-1})^{\dagger\lambda_i}\left(\dot{x}_i - J_i \sum_{j=1}^{i-1} \dot{m}_j\right) \\ \dot{q} &= \left(\sum_{i=1}^{t} \dot{m}_i\right) + P^A_t \dot{q}_{opt} \end{aligned} \tag{17}$$

where $\dot{m}_i$ is the joint velocity contribution of task T_i to $\dot{q}$ and the projector P^A_i is computed incrementally with (15).

6. The second task-priority formulation (F_2)

Chiaverini [4, 5 : Eq. 34] proposed the following formulation, with two tasks :

$$\dot{q} = J_1^{\dagger}\dot{x}_1 + (I - J_1^{\dagger}J_1)[J_2^{\dagger}\dot{x}_2] \tag{18}$$

Comparing with the first formulation, algorithmic singularities are absent, but there is a greater tracking error for the secondary task. To handle kinematic singularities, he introduces damping [5 : Eq. 46] :

$$\dot{q} = J_1^{\dagger\lambda_1}\dot{x}_1 + (I - J_1^{\dagger\lambda_1}J_1)[J_2^{\dagger\lambda_2}\dot{x}_2] \tag{19}$$

In eq. (19) the use of a DLS-inverse in the projector potentially causes some interference between the tasks (as noted in Section 4), when the primary task is approaching a kinematic singularity (i.e. when λ_1 is large). Even if in practice the undesirable effects are limited, there is no justification for damping the inverse used in the projector.

An intuitive explanation of this formulation is : for the low priority task we compute the joint velocity that should be produced to complete the task, then we hold only those components that do not affect the high-priority task, projecting on the null-space of its Jacobian J_1.

Extension to three or more tasks (priority levels) :

To extend this formulation to allow a third priority level, we follow the same philosophy to find the associated projector P : the joint velocity of the third task is computed independently ($\dot{r} = J_3^{\dagger\lambda_3}\dot{x}_3$), then it is filtered out so that it does not affect the primary and secondary task.

The form of the solution is :

$$\dot{q} = J_1^{\dagger\lambda_1}\dot{x}_1 + (I - J_1^{\dagger}J_1)(J_2^{\dagger\lambda_2}\dot{x}_2) + P\dot{r} \tag{20}$$

The filtering conditions are mathematically expressed as :

$$\forall \dot{r}, \begin{cases} J_1 P\dot{r} = 0 \\ J_2 P\dot{r} = 0 \end{cases} \Leftrightarrow J_2^A P\dot{r} = 0 \tag{21}$$

Clearly, (21) holds if we choose $P = P_2^A$, because $J_2^A(I - J_2^{A\dagger}J_2^A) = 0$. Of course, this can also be applied for the subsequent tasks; thus we can write the general expression of the second formulation for t tasks, with the help of the null-space projector P^A_i computed with (15) :

$$\begin{aligned} \dot{m}_i &= P^A_{i-1} J_i^{\dagger\lambda_i}\dot{x}_i \\ \dot{q} &= \left(\sum_{i=1}^{t} \dot{m}_i\right) + P^A_t \dot{q}_{opt} \end{aligned} \tag{22}$$

7. Discussion

In this section we point out advantages and weaknesses of both formulations. We first compare the structure of the equations, and explain the impact of the differences and the behavior they induce. Then their computational cost is briefly analysed.

The general expressions (17) and (22) only differ in the evaluation of $\dot{m}_i$. Two differences are identified :

- in (17), a term is subtracted from the commanded velocity vector $\dot{x}_i$: it can be interpreted as a *compensation* term taking into account the velocity induced in the T_i task space by the higher-priority tasks, to adjust the commanded velocity.

- the (task) mapping from task space to joint space is different : $(J_i P^A_{i-1})^{\dagger\lambda_i}$ for F_1, and $P^A_{i-1} J_i^{\dagger\lambda_i}$ for F_2. Let us describe this difference in more detail. In F_1, the optimal solution is found in the subspace $J_i P^A_{i-1}$ of joint velocities that can be realized without affecting higher priority tasks. In F_2, the operations are performed in reverse order : first, the optimal solution to (1) is found with (2), independently from the other tasks, then it is filtered by projecting on the null-space of all higher-priority Jacobians (through P^A_{i-1}). While F_2 is optimizing the search in a wrong subspace (because the result is then filtered, and the optimality is lost), F_1 directly works in the subspace of admissible solutions, searching for the best one. Thus the effective T_i task velocity, given by $J_i \dot{q}$, is closer to the desired velocity $\dot{x}_i$ with F_1 than with F_2.

To better understand their impact, consider two end-effectors E_1 and E_2, of main and secondary task respectively, on a simple chain. We have tested three situations for each formulation :

- move E_1 and study the movement of E_2 : with the first formulation, E_2 stays fixed if no damping is used, but near algorithmic singularities this causes severe oscillations and even the violation of the priority levels, thus requiring the use of damping[1]. In this case, E_2 is no more fixed, but still quickly comes back to its target point. Conversely, the use of F_2 leads to very large movements of E_2, ultimately starting to converge to its target point only when we stop to move E_1. Thus F_2 misses a desirable property featured by F_1, but at least avoids the delicate problem of algorithmic singularities.
- move E_2 and study the movement of E_1 : as expected, E_1 does not move with both formulations. The difference lies in the way E_2 converges to its target point : while there is no major problem with F_1 (the path is straight, or slightly perturbed near algorithmic singularities), the use of F_2 results in anisotropic displacements (the convergence is fast in some directions and very slow in others, so that the end-effector path is not straight at all). This is another weak point of F_2.
- attract E_2 out of its reachable space : what is expected is to have E_2 attain the reachable space boundary and minimize the distance to the unreachable target. This is what effectively happens with F_1, while with F_2 the end-effector first reaches the limit but then moves to a non-optimal position.

The many problems of F_2 are to be attributed to the differences identified previously, and in particular the different task mapping. The problems are amplified with three or more tasks, and thus F_2 quickly becomes impractical.

Finally, we compare the computational cost of the formulations (17) and (22), keeping in mind that the speed of an iterative algorithm not only depends on the computational cost of a single iteration, but also on the quality of the convergence, which is clearly better with F_1 (Cf. Section 7). We focus our attention on the most expensive operation, that is matrix inversion : both require $(J_i P^A_{i-1})^{\dagger}$ for the projector, while F_1 requires $(J_i P^A_{i-1})^{\dagger\lambda_i}$ and F_2 requires $J_i^{\dagger\lambda_i}$. As already noted in Section 5, it is a nice property of F_1 that two different inverses are required for the *same* argument, which can be exploited to optimize the computation. This is not possible with F_2, thus two inverses have to be computed separately. This suffices to conclude that F_1 is also slightly faster than F_2.

8. Application : control of a human figure

Because of the better behavior of F_1, and despite the occasional oscillations due to the algorithmic singularities, we prefer it here to control a complex articulated human-like 3D figure (49 DOFs with joint limits). To this purpose, we define three tasks :

T_1 : obstacle a voidance (the figure is pre vented from entering some simple-shaped regions of space); each joint is tested against each obstacle and, if within a given range, it is constrained to move away from the obstacle, according to the surface normal vector. This approach is similar to the one described in [12], except that multiple constraints (with equal priority) may be active simultaneously.

T_2 : center of mass (CoM) position control; to ensure stability of the figure, the CoM is constrained to be on a vertical line passing through the support area. Given a mass distribution, the CoM can be controlled like any other end-effector, by using the *kinetic* Jacobian relating unit joint velocities to the CoM cartesian velocity [2].

T_3 : control of one or more end-ef fectors (a hand in the following example).

Obstacle avoidance is given the highest priority (because it is a physical constraint); then with a lower priority we constrain the CoM position, because we are generally interested only in balanced postures. Finally, we want to interactively move arbitrary end-effectors, and see how far they can be moved without affecting the other tasks (here, to identify their reachable space boundary).

1.In practice, the damping factor is computed with a formula that bounds the solution norm to a limit provided by the user [13].

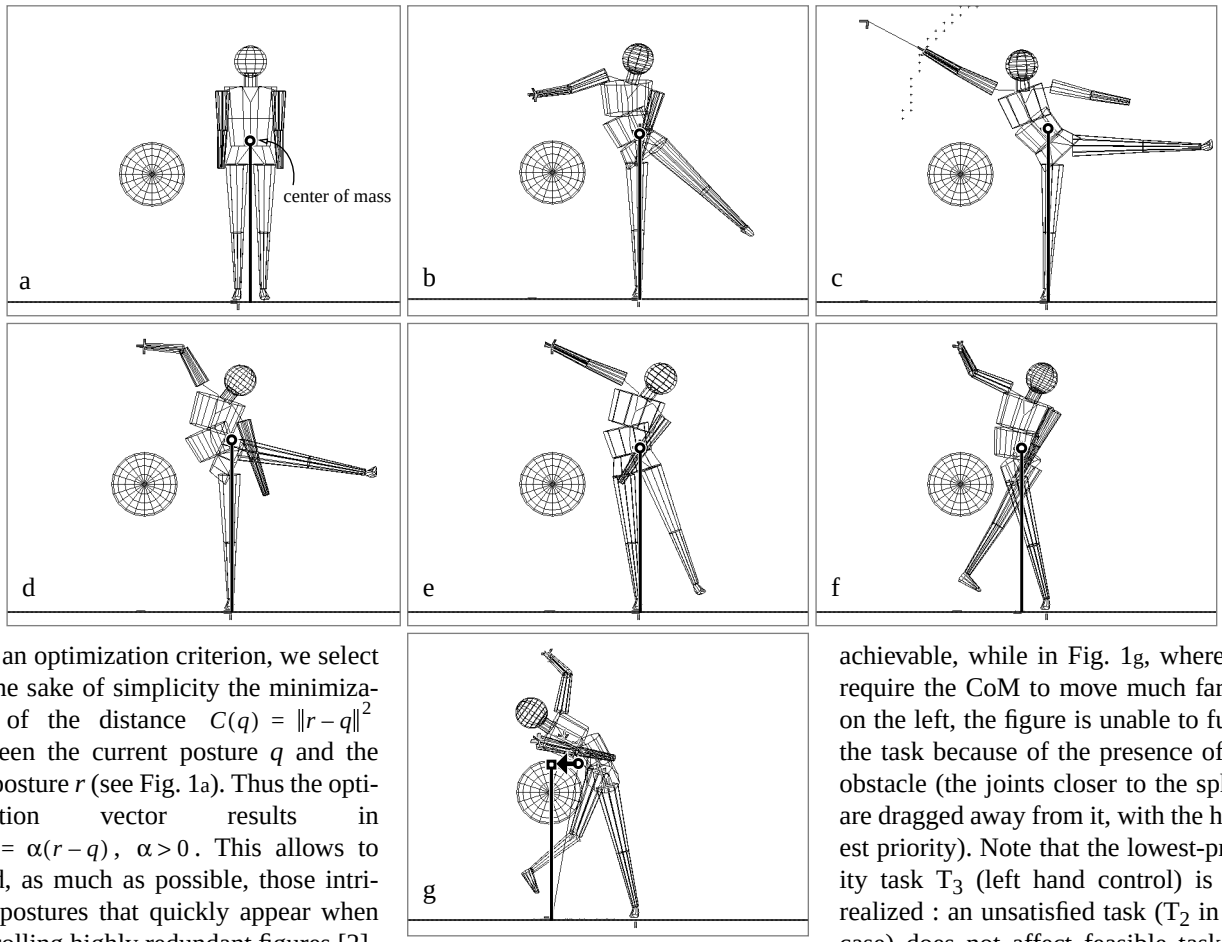

Fig. 1 : control of a human figure

As an optimization criterion, we select for the sake of simplicity the minimization of the distance $C(q) = \|r - q\|^2$ between the current posture q and the rest posture r (see Fig. 1a). Thus the optimization vector results in $\dot{q}_{opt} = \alpha(r - q)$, $\alpha > 0$. This allows to avoid, as much as possible, those intricate postures that quickly appear when controlling highly redundant figures [2].

Successive frames are taken from an interactive session in a 3D space (only an orthogonal projection is shown here). Fig. 1a shows the initial rest posture of the human figure, whose left foot is the motion root (fixed point of contact with the floor). The sphere on the left represents an obstacle that has to be avoided by the figure. We start by controlling the left hand as well as the CoM, which is constrained to project vertically on the left foot. The hand is constrained to move to a point above the sphere, while avoiding the obstacle and maintaining balance (Fig. 1b : the final posture). The hand is then asked to move far on the left, but this time the goal is unreachable (it could be reachable, without the higher-priority tasks). Fig. 1c shows the result, as well as the limit of the local reachable space for that hand. In Fig. 1d, the hand goal is moved back in the reachable space, leading to a somewhat uncomfortable posture. In Fig. 1e, the rest posture optimization is activated to show its effectiveness (during the transition towards the more natural posture, all the constraints stay satisfied). Finally, the priority of task T_1 over task T_2 is illustrated by asking the CoM to move to the left (leading to unbalanced postures). In Fig. 1f, the CoM goal is still achievable, while in Fig. 1g, where we require the CoM to move much farther on the left, the figure is unable to fulfill the task because of the presence of the obstacle (the joints closer to the sphere are dragged away from it, with the highest priority). Note that the lowest-priority task T_3 (left hand control) is still realized : an unsatisfied task (T_2 in this case) does not affect feasible tasks of lower priority (T_3), as already noted in [16, p. 1213].

9. Conclusion

Simulation results clearly show that priorities among multiple cartesian tasks of diverse nature have been respected. The optimization of a desired criterion in joint space may be applied with the lowest priority, that is without affecting the progress of these tasks.

Two formulations that allow for task prioritization have been presented, as well as their extensions for handling an arbitrary number of tasks, which is important for highly redundant structures. We improved the computational efficiency of the formulation introduced in [16], with an incremental formula for the nullspace projector evaluation (15). This is useful for real-time applications with many task-priority levels. Besides, we have described the major problems related to the second formulation, mainly due to the inefficient task mapping that searches the least-squares solution in the unconstrained subspace. On the other hand,

the first formulation faces the algorithmic singularities problem, but the artifacts can be reduced with proper damping. In the end, we preferred this latter solution for the control of highly redundant structures with the task-priority strategy.

Acknowledgments

We thank the reviewers that let us know the existence of the paper of Siciliano and Slotine [16]. The research was supported by the Swiss National Research Foundation.

References

[1] Baillieul J., "Avoiding obstacles and resolving kinematic redundancy", *IEEE Int. Conference on Robotics and Automation*, pp. 1698-1704, 1986

[2] Boulic R., Mas R., Thalmann D., "Interactive Identification of the Center of Mass Reachable Space for an Articulated Manipulator", *Proceedings of ICAR'97*, Monterey July 1997

[3] Boullion T., Odell P.L., "Generalized Inverse Matrices", John Wiley and Sons, New York, 1971

[4] Chiaverini S., "Task-Priority Redundancy Resolution with Robustness to Algorithmic Singularities", *Preprints Syroco '94*, Capri, Italy, pp. 453 - 459, 1994

[5] Chiaverini S., "Singularity-Robust Task-Priority Redundancy Resolution for Real-Time Kinematic Control of Robot Manipulators", *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 3, June 1997, pp. 398 - 410

[6] Cleary K., "Incorporating Multiple Criteria in the Operation of Redundant Manipulators", *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 618-624, 1990

[7] Egeland O., Sagli J.R., Spangelo I., Chiaverini S., "A damped least-squares solution to redundancy resolution", in *Proc. 1991 IEEE Int. Conference on Robotics and Automation, Sacramento*, CA, pp. 945 - 950

[8] Espiau B., Boulic R., "Collision Avoidance for Redundant Robots with Proximity Sensors", *3rd Int. Symposium on Robotics Research*, Gouvieux, France, Oct. 1985

[9] Hanafusa H., Yoshikawa T., Nakamura Y., "Analysis and Control of Articulated Robot with Redundancy", *IFAC, 8th Triennal World Congress*, 1981, Vol. 4, pp. 1927-1932

[10] Huang L-G., Yaotong L., "Kinematic Control of Redundant Manipulators : Implementation Issues", *ICAR '97*, pp. 147 - 152

[11] Klein C.A., Huang C.H., "Review of Pseudoinverse control for Use with Kinematically Redundant Manipulators", *IEEE Trans. Systems, Man, Cybern.*, Vol. 13, 1983, pp. 245 - 250

[12] Maciejewski A.A., Klein C.A., "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", *The International Journal of Robotics Research*, pp. 109-117, Vol. 4, No. 3, 1985

[13] Maciejewski A.A., Klein C.A., "Numerical Filtering for the Operation of Robotic Manipulators through Kinematically Singular Configurations", *Journal of Robotic Systems*, 5 (6), pp. 527 - 552, 1988

[14] Maciejewski A.A., Klein C.A., "The SVD : computation and applications to robotics", *Int. Journal of Robotic Research*, Vol. 8, No. 6, pp. 63 - 79, 1989

[15] Nakamura Y., Hanafusa H., "Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control", *Journal of Dynamic Systems, Meas., and Control*, Sept. 1986, Vol. 108, pp. 163 - 171

[16] Siciliano B., Slotine J-J., "A general framework for managing multiple tasks in highly redundant robotic systems", *ICAR'91*, pp. 1211 - 1216

[17] Wampler C.W., "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods", *IEEE Trans. Syst., Man, Cybernetics*, Vol. 16, pp. 93-101, 1986