INVERSE KINEMATICS TECHNIQUES OF THE INTERACTIVE POSTURE CONTROL OF ARTICULATED FIGURES

THÈSE N^{O} 2383 (2001)

PRÉSENTÉE AU DÉPARTEMENT D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Paolo BAERLOCHER

Ingénieur informaticien diplômé de l'EPF
de nationalité suisse et originaire de Thal (SG)

acceptée sur proposition du jury:

Dr r. Boulic, directeur de thèse
Prof. H. Bieri, rapporteur
Prof. M.-P. Cani, rapporteur
Prof. R. Clavel, rapporteur
Prof. B. Espiau, rapporteur
Prof. D. Thalmann, rapporteur

Lausanne, EPFL
2001

Often people attempt to live their lives backwards;

they try to have more things, or want more money,

in order to do more of what they want,

so they will be happier.

The way it actually works is the reverse.

You must first be who you really are,

then do what you need to do,

in order to have what you want.

- - Margaret Young

Table of contents

Remerciements

Cette thèse est le résultat d'un projet financé par le Fonds National Suisse de la Recherche Scientifique (1996-2000), et effectué au laboratoire d'infographie (LIG). Je tiens avant tout à remercier très sincèrement mon directeur de thèse, Ronan Boulic, qui a initié et supervisé ce projet avec beaucoup d'enthousiasme, de disponibilité et de compétence. Ce fut un plaisir et un privilège de travailler ensemble pendant ces quatre années.

Je tiens aussi à remercier le Professeur Daniel Thalmann, directeur du laboratoire, qui m'a permis de travailler principalement sur ce projet.

Je remercie les membres du jury de thèse, Madame le Professeur Marie-Paule Cani, le Professeur Bernard Espiau, le Professeur Hanspeter Bieri, le Professeur Reymond Clavel ainsi que le Professeur Daniel Thalmann pour le soin qu'ils ont apporté à l'examen de cette thèse. Leurs nombreuses critiques m'ont aidé à améliorer de façon significative ce manuscrit. De plus, je remercie Monsieur Clavel pour ses nombreux conseils, et pour avoir déniché plusieurs erreurs dans la version préliminaire. Je remercie aussi le Professeur W. Gerstner qui a présidé ce jury.

Je remercie vivement Luc Emering, Marcelo Kallmann, et Ik Soo Lim pour avoir relu attentivement ce manuscrit et avoir suggéré des modifications très pertinentes. Je remercie aussi toutes les personnes qui m'ont encouragé lors de la rédaction: cela n'a pas été inutile.

Je tiens aussi à saluer les personnes que j'ai eu le plaisir de rencontrer au LIG, en particulier:

- Thierry et Angela, dont la gentillesse et la sympathie sont sans limites !
- Amaury, Philippe, Rémy, Christophe, Ralf et Etienne, qui ont partagé avec moi la passion pour la fondue au fromage (qu'elle soit moitié-moitié, vacherin ou au champagne) !
- Thierry, Amaury, Ralf, Luc, Walter, Pascal, Tom, Marcelo, Jean-Sé, Srikanth et Hansrudi pour les fructueux échanges d'idées et discussions techniques (ou de toute autre nature...).
- Mireille, Fabrice, Joaquim, Nathalie, Maja, Olivier ... et tout ceux que j'oublie.

Merci aussi aux administrateurs système (SGI et PC), sans lesquels nous ne serions rien !

J'aimerais aussi remercier mes anciens enseignants. En particulier, Uberto Cattaneo et Marco D'Anna, du lycée de Locarno, pour leur enseignement rigoureux et passionnant des mathématiques et de la physique. Le regretté Professeur Charles Rapin, pour ses cours mémorables sur la programmation et la compilation.

Merci aussi à la joyeuse équipe de Mme Salmond, pour leur gentillesse ainsi que pour leurs excellentes soupes qui m'ont fait grandir encore un peu !

Enfin, je remercie mes parents pour m'avoir soutenu et encouragé pendant ces nombreuses années passées à l'EFPL, et pour avoir toujours été disponibles quand j'en avais besoin.

Summary

The context of this thesis is the interactive manipulation of complex articulated figures by means of geometric constraints (here called *tasks*), for the purpose of posture control and design. The goal is to determine a posture satisfying a set of prescribed tasks, usually expressed in the Cartesian space. This approach is known as Inverse Kinematics, and a number of analytic and numerical resolution methods have been developed for the control of robot manipulators. These methods have been applied to the computer animation of articulated figures, and to the control of human models for computer-aided ergonomic evaluations of products or workplaces.

When dealing with figures that possess a large number of degrees of freedom, such as animal or human figures, their posture is usually controlled by several simultaneous tasks. There are tasks of different nature and function: they can control extremities such as the hands and the feet (for reaching or supporting purposes), as well as the center of mass, for balance control. They can also be used to avoid collisions with surrounding obstacles. The concurrent resolution of multiple tasks inevitably leads to conflicts that must be resolved with an appropriate strategy. A typical policy is to find a compromise solution that considers weights assigned to each task to indicate their relative importance. However, no task is precisely satisfied with this approach, and selecting appropriate weights is not always straightforward.

In this thesis, we introduce a priority strategy for conflict resolution. With this policy, a task is not affected by other tasks of lower priority, and is satisfied as much as possible without affecting tasks of higher priority. The relative priority between two tasks is thus strictly enforced, which is appropriate for situations that cannot tolerate compromises. For example, keeping balance is more important than reaching an object with a hand, and avoiding inter-penetration of bodies is more important than any other task. Priorities are well-suited to express such hierarchical relationships.

Based on a task-priority algorithm developed in robotics for simple manipulators, we introduce a framework that integrates the two conflict resolution strategies: first, the priorities assigned to the tasks are considered and, second, a weighting strategy solves the conflicts between tasks having same priority. We have improved the efficiency of the original algorithm by means of recursive relations, which is beneficial for interactive applications. Joint limits and joint couplings are also integrated in the framework to avoid unfeasible body postures.

An interactive application, called *BALANCE,* has been developed to test the algorithm with a palette of task types: it allows us to illustrate the utility of task priorities for the manipulation of generic articulated figures, and of human models in particular. Besides simple geometric tasks, the application also proposes a task to keep the figure balanced under a set of static forces due to the interaction with its environment. It is shown that this task is easily integrated in the inverse kinematics framework, and that it is useful to generate postures in multiple supports with force exertions such as push and pull activities.

Résumé

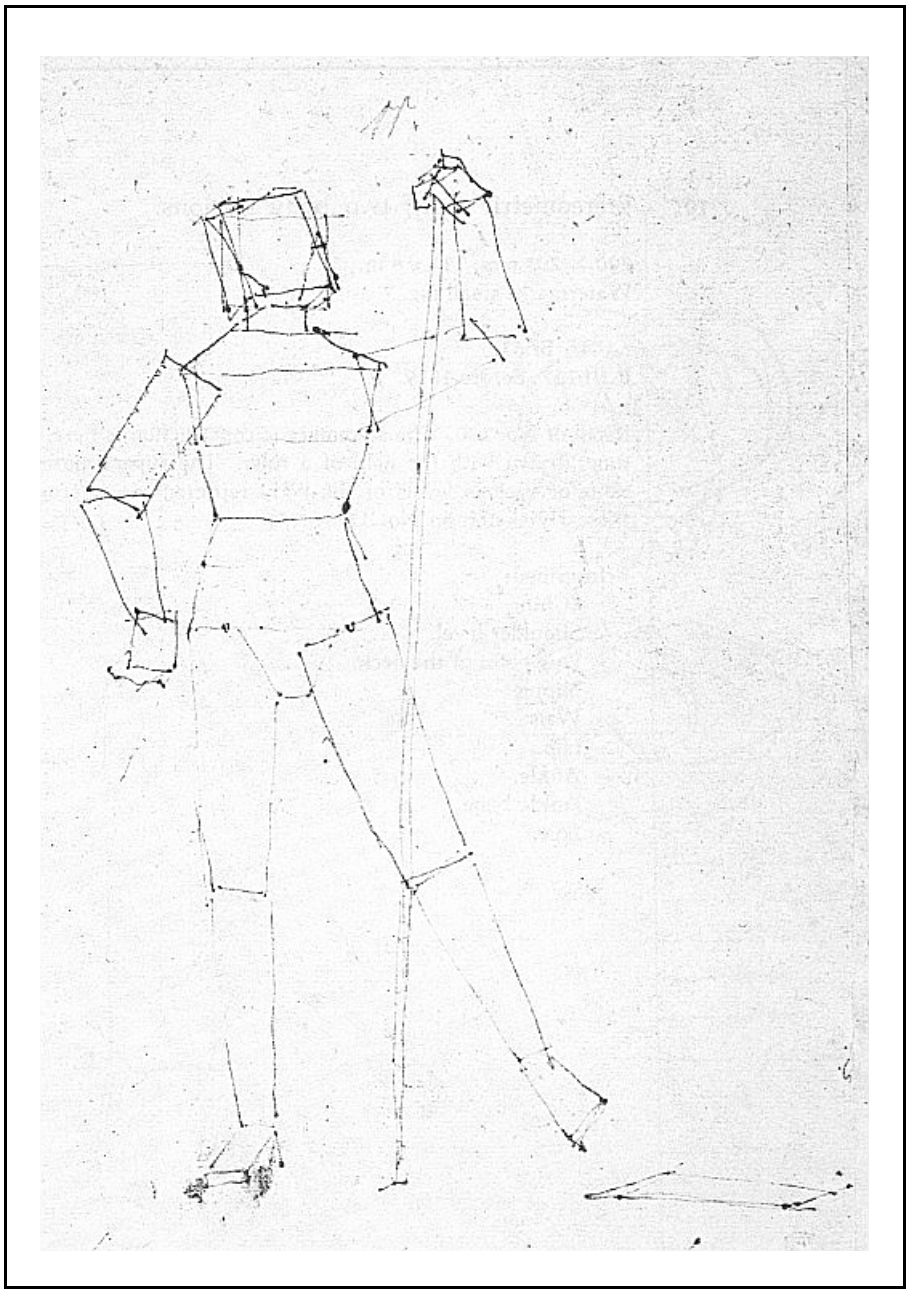
Le contexte de cette thèse est la manipulation interactive de figures articulées complexes, par le biais de contraintes géométriques (appelées *tâches*), dans le but de contrôler et d'éditer leur posture. Il s'agit de déterminer une posture satisfaisant un ensemble de tâches imposées, typiquement exprimées dans l'espace Cartesien. Cette approche s'appelle Cinématique Inverse, et plusieurs méthodes de résolution analytiques et numériques ont été développées pour le contrôle de robots manipulateurs. Ces méthodes ont ensuite été appliquées à l'animation par ordinateur de figures articulées, et au contrôle de modèles humains pour l'évaluation ergonomique assistée par ordinateur de produits ou d'espaces de travail.

Lorsqu'on traite des figures ayant un grand nombre de degrés de liberté, tels que des figures animales ou humaines, le contrôle de leur posture est effectué par le biais de plusieurs tâches simultanées. Ces tâches sont de nature et de fonctions différentes: elles peuvent contrôler des extrémités telles que les mains ou les pieds (pour atteindre un but ou supporter le poids), mais aussi le centre de masse, pour garantir l'équilibre. Les tâches peuvent aussi être utilisées pour éviter les collisions avec les obstacles environnants. La résolution simultanée de plusieurs tâches conduit inévitablement à des conflits, qui doivent être résolus par une stratégie appropriée. Une politique habituelle est de trouver un compromis en tenant compte d'une pondération des tâches indiquant leur importance relative. Cependant, avec cette approche aucune tâche n'est satisfaite précisément, et il n'est pas toujours évident de choisir les poids.

Dans cette thèse, nous introduisons une stratégie à base de priorités pour la résolution des conflits. Avec cette politique, une tâche n'est pas affectée par des tâches de plus basse priorité, et en même temps elle est satisfaite au mieux sans pour autant perturber les tâches de plus haute priorité. La priorité relative entre deux tâches est donc strictement imposée, ce qui est plus approprié pour des situations qui ne tolèrent pas de compromis. Par exemple, rester en équilibre est plus important que d'atteindre un objet avec la main, et éviter les inter-pénétrations d'objets est certainement plus important que toute autre tâche. Les priorités sont bien adaptées à l'expression de telles relations hiérarchiques entre tâches.

A partir d'un algorithme développé en robotique pour des manipulateurs simples, nous proposons une méthode qui intègre les deux stratégies pour la résolution de conflits: en premier lieu, les priorités affectées aux tâches sont respectées et, en deuxième lieu, la pondération de tâches ayant la même priorité est prise en compte. Nous avons amélioré l'efficacité de l'algorithme original grâce à l'utilisation d'une relation de récurrence, ce qui est avantageux pour des applications interactives. Les limites et couplages articulaires sont aussi intégrés dans l'algorithme, ce qui permet d'éviter les postures infaisables.

Une application interactive, appelée *BALANCE*, a été développée pour tester l'algorithme avec une variété de types de tâches: elle nous permet d'illustrer l'utilisation des priorités pour la manipulation de figures articulées quelconques, et de modèles humains en particulier. En plus de contraintes géométriques simples, l'application propose une tâche pour le contrôle de l'équilibre, sous l'action de forces statiques dues à l'interaction avec l'environnement. Nous montrons que cette tâche est facilement intégrée dans le cadre de la cinématique inverse, et qu'elle est utile pour la génération de postures de figures en support multiple exerçant des forces pour pousser ou tirer des objets.



"**Stereometric man**", Albrecht Dürer, about 1523 (source: Strauss [STR 72]).

Chapter 1

Introduction

1.1 Motivation

This thesis addresses the problem of manipulating articulated figures in an interactive and intuitive fashion for the design and control of their posture. The main motivation for this application comes from computer animation, but other areas (such as robotics and ergonomics) are concerned as well.

1.1.1 The need for an interactive tool for articulated figures manipulation

In Computer Graphics, articulated figures are a convenient model for humans, animals or other legged creatures that now appear in films and video games. The animation of such models is often based on motion-captured data, or procedurally generated motions. Despite the availability of such techniques, the manual design of key postures (see Fig. 1.1) is still widespread because animators have total control over the results. However, this is a laborious task because of the high number of degrees of freedom present in the models (typically fifty for a human model without considering the fingers). Commercial animation tools such as Maya [ALI 98] already provide good graphical user interfaces to assist the animators in this process. However, and because of the complexity of the problem especially when dealing with human figures, these tools certainly need further improvements.

Figure 1.1 A virtual human-like creature (source: Nichimen advertisement).

The field of computer-aided ergonomics is also concerned with articulated figures, especially human models developed for simulation and prediction purposes. Rather than appearance, the main issue here is an accurate biomechanical modelling and body sizing based on anthropometric data. Simulations are performed to evaluate the reaching and viewing capabilities of the model as well as the safety and performance of workers executing manual tasks. In the field of computer-aided design (CAD), virtual mock-ups are built to speed up the development process and lower the costs for prototyping new engineering products such as cars and airplanes. Possible design flaws can be identified at an early stage by simulating the interaction of human models with the virtual products. Such applications require simple and interactive tools to place the human models with respect to their environment (see Fig. 1.2).

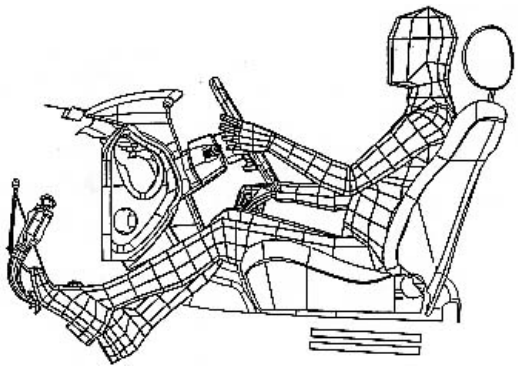


Figure 1.2 Evaluation of a driver's seat with the MAN3D model (INRETS).

Similarly, in robotics, there is a need for the rapid prototyping of robot manipulators with simple tools that can be understood and exploited by non-specialists [FLU 98]. At the conception stage, robots can be quickly designed and their kinematic and dynamic capabilities simulated and evaluated. For example, the reachable space of a robot can be visualized (see Fig. 1.3) in order to evaluate its ability to perform a task.

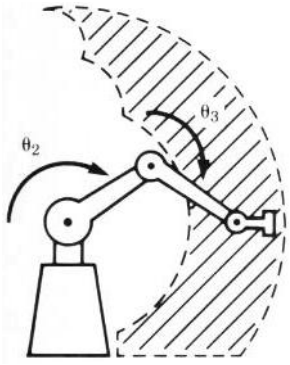


Figure 1.3 A simple articulated robot manipulator (source: [CRA 89]).

1.2 Posture control: issues and techniques

1.2.1 Posture constraints

A posture is simply a skeletal configuration of the figure. However, not all postures are acceptable. In order to be realistic, they must satisfy a set of criteria: for example, the natural limits of the articulations should not be violated, and inter-penetration of the body with other objects or with itself is not permitted. Physical laws should also be taken into account. For example, for static postures, the laws of statics must be respected: this requires the information of the body weight and of the other forces acting on the figure. Obviously, these forces must also be realistically defined: for example, the friction limits of the supporting surfaces should not be transgressed.

Besides such general constraints that apply to all articulated figures, the special case of human posture control presents major difficulties, because of the large number of degrees of freedom, intrinsic unstable equilibrium, and the numerous personal factors (such as gender, age, muscular strength, experience, ...) that influence the selection of a posture. The intricacy of the musculo-skeletal system also affects the posture: tendons and muscles may couple the motion of adjacent joints, for example.

Noteworthy, a small change in the posture dramatically affects our perception of a character: the human eye is very sensitive to small postural details, that in fact convey much information. A similar observation can be made in ergonomics analysis: according to Chaffin and Erig [CHA 91], postural analyses are very sensitive to small postural differences. Hence, posture manipulation and evaluation tools must provide the necessary accuracy in order to capture these significant postural details.

1.2.2 Inverse Kinematics for posture control

Within the set of admissible postures, the user is free to manipulate the figure at will. Rather than specifying the value of each individual degree of freedom, the Inverse Kinematics method automatically computes these values in order to satisfy a given task usually expressed in Cartesian space. Roboticians have studied this problem a long time ago, for the control of robot manipulators. This technique requires the resolution of complex non-linear equations, and is usually expressed as a constraint-satisfaction problem. Actually, a large class of geometric modelling, design [SUT 63] [GER 85] [KRA 92] [AND 96], manipulation [GLE 94], and animation tasks [BAR 88] [VAN 92] are expressed by means of constraints, and the inverse kinematics problem is a particular case of constraint satisfaction for articulated figures. Here, we use the term "*task*" rather than "*constraint*" to express a desired relationship between joint angles and Cartesian coordinates, because they are not always achievable. The term "*constraint*" is reserved to conditions that must hold in any case.

The control of a complex figure by means of inverse kinematics requires that multiple tasks be simultaneously applied. For example, a task may control the position of a hand, to simulate a reaching action, while another task controls the point of interest that the figure is looking at. The balance of the figure can also be controlled by a task, provided that the necessary information about mass distribution and supported forces is available. The combination of multiple tasks is a powerful approach to the problem of posture control. However, the problem is often redundant because the number of tasks is lower than the number of degrees of freedom: hence an infinite number of solutions may exist, and additional optimization criteria usually expressed in joint space must be specified in order to select the "best" solution.

1.3 Goal and contributions

In this thesis, we focus on a particular problem that arises when solving multiple tasks simultaneously. Conflicts inevitably occur between tasks, and a strategy must be selected in order to resolve the situation. A common strategy is to find a least-squares solution, which is in fact a

compromise solution between all conflicting tasks. We propose another strategy that satisfies the tasks by order of priority: with this approach, an important task can be satisfied even if this prevents a less important task to be satisfied as well. We show the usefulness of this strategy, especially when tasks of different nature are combined.

We propose a numerical resolution framework, based on robotics techniques, that combines both strategies, and that also allows to exploit any redundancy left available by the tasks in order to optimize a desirable criterion. We also improve the efficiency of the original robotics algorithm by means of recursive relations.

The method is illustrated on a variety of figures, from simple ones to complex human models with about fifty degrees of freedom. A variety of tasks can be assigned to the figure. In particular, we give a simple derivation of a balance task under the action of external forces in a static equilibrium context. The task easily fits within the framework because it is expressed as a function of kinematic quantities used for the control of end-effectors and of the center of mass.

1.4 Plan of this document

The plan of this document is as follows.

- In chapter 2, we review the main previous works in computer animation, robotics and ergonomics, for the problem of posture control.
- In chapter 3, we discuss the articulated body model and, in particular, the joint models which are the key components for our purposes.
- In chapter 4, a numerical resolution method for the inverse kinematics problem with a single task is detailed. A few important task types are presented.
- In chapter 5, we discuss two strategies for solving the problem of conflicting tasks. The task-priority strategy algorithm is presented. We then introduce a recursive identity that speeds up this algorithm. Finally we show how linear equality and inequality constraints can be integrated in the algorithm, for example to ensure joint limits.
- In chapter 6, the static forces occurring at the interfaces of the figure with its environment are taken into account. With this additional information, the posture can be adjusted to stay balanced, and the joint torques can be minimized to obtain more comfortable postures.
- In chapter 7, we present our interactive testbed application *BALANCE*, that integrates all elements introduced in the previous chapters. It is used to illustrate the usefulness of task prioritization together with a set of tasks of different nature including balance control.
- In chapter 8, we summarize the contributions of this thesis, and suggest future research directions.

At the end, the symbols used in this document are listed for convenience.

1.5 Mathematical notation and conventions

In this document, we use the *column vector* convention and *right-handed coordinate frames*. Scalars are denoted by small letters such as *s*. Vectors are denoted by small boldface letters such as $\boldsymbol{v}$. The i^{th} component of a vector $\boldsymbol{v}$ is noted v_i, starting with $i = 1$. Alternatively, the components of a 3D vector may be noted v_x, v_y and v_z. The three basis vectors of a coordinate frame are denoted by $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$. Matrices and frames are denoted by capital letters such as M, while points are denoted by capital boldface letters such as $\boldsymbol{P}$. The $n \times n$ identity matrix is noted I_n.

The unary cross operator is defined as $[\boldsymbol{v}\times] := \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$

Note that $[\boldsymbol{v}\times]\boldsymbol{w} = \boldsymbol{v} \times \boldsymbol{w}$. Other properties of this operator are listed in Appendix A.

Additional notation will be introduced when necessary.

Chapter 2

State of the art

2.1 Introduction

In this chapter, we present previous work related to the problem of posture control in the computer animation field, in robotics and ergonomics. Despite the different goals, results obtained in these application areas can often be shared. We start with the special and difficult problem of human body modelling that arises in ergonomics and in computer graphics applications.

2.2 Human body modelling

When dealing with human figures, an appropriate model must be defined depending on the intended application. Its accuracy may have a profound impact on the validity of the predictions that result from its use. However, simplifying assumptions must be performed in order to have a tractable model. They are acceptable as long as the user is aware of the resulting limitations. Hence, a good model is not necessarily a very complex model, but it is a model that matches the user's needs with the minimum set of pertinent parameters.

Most body models assume that body parts are rigid, and cannot be broken. Of course this is an approximation of reality. For more precise but also much more computationally expensive models, the finite element method is the classical technique for computing strains due to applied stresses, according to the constitutive laws of the material. These laws are extremely complex for living tissues.

Virtual human modelling starts in the sixties with models developed by the aerospatial and automotive industry to design and evaluate cockpits. A number of increasingly more complex models have followed, to evaluate more general man-machine interactions [KIN 81]. A great deal of research has been performed by Badler and co-workers [BAD 93] that eventually lead to the Jack model. However, much work is still to be done in this area. The problem comes from the complexity of the human body with its intricate musculo-skeletal system and complex motor control system.

We now list a number of important aspects in virtual human modelling, related to human posture control. Because of the extent of the question, this list is not meant to be exhaustive.

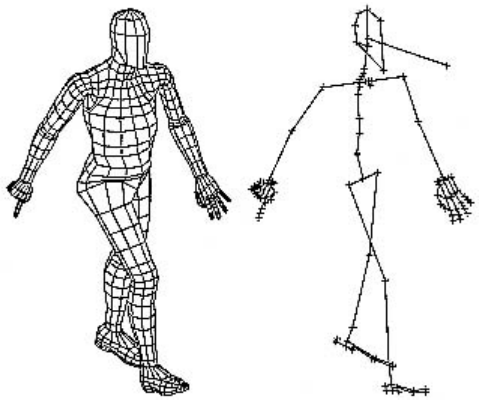


Figure 2.1 The MAN 3D model (INRETS).

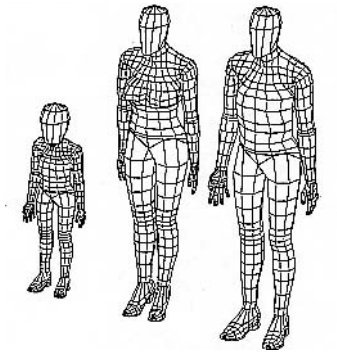


Figure 2.2 Different instantiations of an anthropometric model based on MAN3D (INRETS).

2.2.1 The skeleton

At the most basic level, the skeletal structure is modeled as a hierarchy of rigid segments connected by joints (see Fig. 2.1). Segments are usually defined by their length, shape, volume and mass properties, but the bones are not necessarily modeled as 3D objects. The joints are used to modify the posture of the body. A mesh can then be attached to the skeleton.

2.2.2 Anthropometry

Anthropometry is concerned with the measurement of size, shape and proportions of the human body and its segments. Collections of anthropometric data have been performed on large populations to obtain statistics [KRO 90]. These data are essential to build human models with realistic dimensions and proportions. Samples of a given population can be generated for the ergonomic evaluation of a product over a valid range of human sizes, or to populate a heterogeneous crowd. Fig. 2.2 shows three human models in different sizes based on the MAN3D model developed at the french institute INRETS in Lyon. Similarly, the SASS module developed for the Jack system performs the sizing process via a graphical user interface [GRO 89] [BAD 93].

2.2.3 Joint modelling

A joint is the body component concerned with motion: its essential feature is that it permits some degree of relative motion between the two segments it connects. Ideal kinematic joint models are defined in order to formalize this permitted relative motion, called *range of motion*, characterized by the number of parameters that describe the motion space and constrained by joint limits. Modelling real joints can be very complex because the motion range depends on many factors, especially in the articulations of complex living organisms and the human in particular [BAD 93]: typical examples are the shoulder and spine structures which are in fact made of coupled joints. Despite this, ideal and simplified joint models must be defined in order to be tractable.

The simplest example of joint model is the *revolute joint*: it allows a rotation about an axis fixed in both segments it connects, usually within some angular limits. This joint is said to have one degree of freedom (DOF) and, because of its simplicity, is by far the mostly used joint in robotics [CRA 89]. It is a convenient model for interphalangeal joints. More complex joint models with multiple degrees of freedom are required to represent articulations such as the shoulder and hip. The joint limits are also an important component of a joint model, since they restrict the motion space to the realistic range of motion. Korein [KOR 85] uses spherical polygons as boundaries for the directional component of spherical joints like the shoulder. Biomechanical studies on the motion range of the human shoulder have been done by Engin *et al.* [ENG 89] and Wang *et al.* [WAN 98]. Maurel and Thalmann also discuss the use of joint sinus cones for the shoulder and scapula joints [MAU 00].

Further on, within the valid motion range of a joint, it is also possible to define a potential function that indicates the "comfort" or "naturalness" of a given configuration [HIR 96] [AYD 99]. For example, the comfort perceived at the neck decreases when its joint boundaries are approached. This information can be exploited to select more natural postures.

Joints may be dependent on each other, especially in living organisms. This coupling (of motion and limits) can be integrated directly in the body definition, for example with the concept of *joint group* [BAD 93], which is a set of joints seen as one "black box" entity that ensures the coupling between its elements. Another possibility is to leave the application deal with the problem: usually, this can be done with the use of kinematic constraints.

2.2.4 Shoulder complex modelling

An example of joint coupling is provided by the *shoulder complex* composed of clavicle, scapula and shoulder joint. The kinematic relationship that holds between these components is very complex and has been studied in biomechanics. For example, the scapulo-thoracic constraint (ST) has been studied by Maurel *et al.* [MAU 00], and enforced with inverse kinematics (see Fig. 2.3).

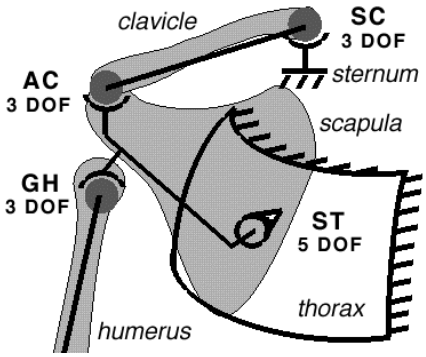


Figure 2.3 The scapulo-thoracic constraint ([MAU 00]).

Shoulder joint limits are also a complex issue. Engin and co-workers [ENG 89] have studied the directional limits of the upper arm while Wang *et al.* [WAN 98b] have investigated its twisting limits.

2.2.5 Spine modelling

The spine is a complex arrangement of 24 vertebrae, whose motion is coupled by ligaments [WHI 90]. Still, for simplicity, the spine is often modelled as a chain of (uncoupled) joints, thus leaving to the motion generators (or to the user) the difficult task of constraining the spine shape to realistic postures. With this approach, the worst case would be to have independent mobility at each one of the vertebrae. Reducing the number of vertebrae leads to less parame-

ters, but still is not an ideal solution.

Korein models the spine as a curve [KOR 85], but it is difficult to obtain a realistic configuration simply by controlling its curvature. A more sophisticated model of the spine and torso taking coupling into account has been proposed by Monheit and Badler [MON 91] [BAD 93]: the movements of the spine are described in terms of total bending angles in the sagittal, lateral and axial directions. The total bending is distributed on the individual vertebrae according to weighting factors.

2.2.6 Hands modelling

The hands have a large number of joints (about fourteen per hand), with coupling between interphalangeal joints due to tendons. Hands are certainly the most versatile part of our body. Indeed, grasping is a very difficult task to simulate [RIJ 91]. Grasping robots have been developed, often inspired by the human hands [MUR 94].

2.2.7 Strength modelling and force exertion

The motion of an articulated figure, and the postures it assumes, are mainly due to the forces that the skeletal muscles apply on the bones. Hence, the muscular strength, which is highly variable among people, is an important component of a human model for prediction purposes, of manual lifting tasks for example [AYO 87]. Typically, an ergonomist is interested in the maximum force that a person may exert at a given point, in a given direction. Another question is to determine the safest (or most comfortable) posture for exerting a force. Hence, a model of strength available at the joint level may help to answer to questions on the force exertion in Cartesian space.

Strength is an extremely complex information and is difficult to measure for each muscle separately. Usually indirect measures are performed at the joint level: the measured torque is due to the group of muscles acting on that joint, for a given direction of exertion (e.g. flexion or extension). A joint strength is also influenced by its own position and velocity, as well as those of adjacent joints. Moreover, strength decreases over time, according to the endurance of a person: this variability may be described by a *fatigue* model. Other global factors such as age and gender also affect the strength of a person. Many studies, such as [AYO 81], have been performed to collect strength data with the aim of developing an *atlas of strength*. However, it is an arduous task to establish the whole human strength model: at the present time only partial results based on various simplifying assumptions are available.

Because of the inherent difficulty in dealing with strength at the joint level, a number of researchers bypass the joint level and directly relate postures to the forces that can be exerted by an end-effector in the Cartesian space [HAS 90]. A review of such studies is given in [DAA 94], with the aim of developing an *atlas of force exertion* for product design.

Garg and Chaffin [GAR 75] present a three-dimensional simulation of human strength, that predicts the maximum hand forces that an individual would be able to exert safely. A complete

program, called 3D-SSPP (Static Strength Prediction Program), has been developed from this early studies [CHA 99].

Lee *et al.* [LEE 90] [BAD 93] describe a system that generates motions of lifting tasks based on a static strength model of the arm. The strength information is exploited to define a comfort criterion whose maximization allows the selection of realistic joint trajectories.

An alternative, anatomy-based, approach is to directly model the muscles with their attachment sites and other physiological parameters. and to compute the forces exerted by those muscles on the bones. Komura *et al.* [KOM 99] use this kind of model to generate dynamically realistic motions, and to evaluate the dynamic abilities of the human body such as the maximal force that can be exerted by an end-effector.

2.2.8 Body appearance

A realistic appearance is important in many applications. On top of the skeletal structure, additional data must be added for the generation of realistic skin, face, skin and cloth, among others. Generating skin, for example, can be achieved with complex techniques such as the anatomy-based modelling of the underlying muscles [SCH 97]. A large number of polygons is required to display a smooth shape, and texture mapping improves the appearance of the materials. In interactive systems, the complexity of the rendering phase may slow down the performance of the system, which is detrimental to the interactivity. A possible solution to alleviate the problem is to use several models with different levels of detail. If appearance is not the main issue, as in ergonomics evaluation systems, a simpler rendering is certainly acceptable. For example, the skin can be approximated by a set of simple geometric primitives (like cylinders or frustums) or by a rough mesh (as in Fig. 2.1).

2.3 Techniques for the animation and control of articulated figures

Once a body model has been defined, it can be manipulated, animated or used for simulation purposes.

By *manipulation*, we mean that its posture is adjusted in an interactive manner, directly by the user. The design or adjustment of realistic postures is an important issue in computer animation, where the generation of motion often relies on a set of well-designed key postures.

The *animation* of a figure is the generation of a sequence of postures. In some applications, such as video games, this must be performed in real-time. Robot control applications also must adjust the configuration of robot manipulators in real-time, in order to perform a given task.

Simulating human articulated figures is important for prediction purposes, and hence is a tool for ergonomics evaluations and product design. Simulation can answer to questions related to the suitability of a posture with respect to accessibility and visibility constraints.

2.3.1 What is motion ?

Motion is a change in the position of an object with respect to a reference, and mechanics is the science that studies the motion of objects. For practical purposes, its treatment is split in two fields:

- **kinematics**, deals with the geometry of motion regardless of its physical realization (in other words is concerned with the position, velocity and acceleration of bodies);
- **dynamics**, based on Newton's laws of motion, relates the causes of motion (i.e. the forces) to the acceleration of a body, taking into account its mass.

This distinction leads to two classes of techniques for the animation of articulated figures: kinematic methods and dynamic methods.

2.3.2 Kinematic manipulation techniques

Manipulating a virtual articulated figure is similar to the manipulation of a marionette: the objective is to obtain a desired posture. A first possibility is to control the relative rotation of the segments by adjusting the joint angles: determining the posture corresponding to a given set of joint parameters is called the direct kinematics problem. This is a very useful posing tool for the designer, as shown by Watt and Watt [WAT 92]. However, when the position of a body part matters, it is difficult to adjust the joints to obtain the given position. The typical example is to adjust the joint angles of the shoulder, elbow and wrist joints in order that the hand exactly reaches an object. This problem is known as the inverse kinematics problem, and can be seen as a constraint on the set of possible joint parameters. Solving this problem is hence very useful for the manipulation of articulated figures, and indeed has been extensively studied, first in robotics and later in computer graphics.

2.3.3 Kinematic animation techniques

The goal of any animation technique is to generate believable motion. Technically, the simplest solution is to rely on skilled animators, that manually define key postures of the articulated figure by directly setting the parameters of its joints. Then, the postures are smoothly interpolated over time to generate the full set of frames required for an animation, and the animator is free to adjust the resulting postures if they are not satisfying. This is known as the keyframing technique.

The motion capture technique [MEN 00] consists in the tracking and recording of a set of markers strategically positioned on the object of interest (typically a person). To obtain its posture, the Cartesian positions of the markers have to be converted into a set of joint angles (examples of this process are given by Badler *et al.* [BAD 93b], Hirose *et al.* [HIR 98] and Molet *et al.* [MOL 99]). This approach requires expensive hardware but also results in natural-looking motion, hence it is extensively used in the entertainment industry (special effects and video games). Due to the errors introduced by the measurement, the data should be corrected (manually or automatically) in order to obtain more accurate motions. Additional difficulties

arise, depending on the tracking technology (magnetic or optical). Moreover, as with keyframing, the resulting motions only suit a particular figure with fixed dimensions. To overcome this limitation, complex motion retargeting techniques must be employed (examples are given by Gleicher *et al.* [GLE 98] and Monzani *et al.* [MON 00]).

Procedural techniques have been developed in order to generate specific motions such as walking or grasping. An example of a purely kinematic model based on biomechanical data is given by Boulic *et al.* [BOU 90]: the resulting gaits are controlled by a set of high level parameters in order to personalize the style of walk.

When the Cartesian position of particular end-effectors is important, inverse kinematics techniques are needed. Goals for end-effectors may be animated over time, and a sequence of postures satisfying them is generated (when possible): this has been called *goal-directed motion* [KOR 82]. However, inverse kinematics is merely a constraint-satisfaction tool, and there is no guarantee that the resulting "motion" is continuous or looks "right": actually, a very small change of the goal in Cartesian space may result in a huge change of the configuration, because the inverse kinematic problem is intrinsically ill-conditioned near singular configurations, as shown by Maciejewski [MAC 90]. Despite this limitation, IK can still successfully animate figures if it is reasonably used. It can also be used to locally adjust motions, as shown by Boulic *et al.* with the coach-trainee method [BOU 92]. This is useful also for the correction of captured motions and for motion retargeting applications [MON 00].

In Computer Animation, one of the first examples of articulated figure animation based on inverse kinematics is due to Girard and Maciejewski [GIR 85] [GIR 87]: the feet of multi-legged figures are constrained on the floor, while a kinematic model of locomotion controls the coordination of the legs. Moreover, simple dynamics are used to provide an overall feel of inertia (see Fig. 2.4). A similar example is provided by the multi-legged animal-like figures animated by Sims and Zeltzer [SIM 88] over uneven terrain.

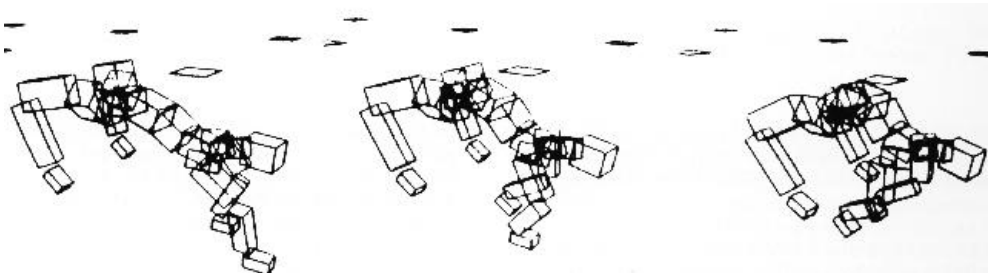


Figure 2.4 A running, legged, animal figure, animated with inverse kinematics for the legs and simple dynamics for the body (source: [GIR 85]).

2.3.4 Dynamics-based animation techniques

A great deal of work exists on the dynamics of articulated bodies [WIT 77] [HUS 90], and efficient direct dynamics algorithms have been developed in robotics for structures with many

degrees of freedom [FEA 86]. In Computer Animation, these algorithms have been applied to the dynamic simulation of insects and later of the human body [MCK 96]. Given a set of external forces (like gravity or wind) and internal forces (due to muscles), these algorithms compute the motion of the articulated body according to the laws of rigid body dynamics. However, the control of such systems by means of internal torques to perform a given task, which is the inverse dynamics problem, is a complex one. In this respect, impressive results have been achieved by Hodgins and co-workers with the simulation of dynamic human activities such as running and jumping [HOD 95]. Another example is given by the virtual creatures of Sims [SIM 94] whose control systems are automatically designed by an evolutionary process based on genetic algorithms. Despite the fascinating results, this technique is limited to simple figures and tasks. The use of constraints to avoid the direct specification of torques has also been researched: for example, in the computer graphics community, the satisfaction of "space-time constraints" have been proposed by Witkin and Kass [WIT 88], with the minimization of an objective function such as the total energy expenditure. Again, this technique is limited to simple problems and is computationally expensive.

2.3.5 Blending and combining animation techniques

Blending different techniques is necessary to animate figures, since no single method solves the difficult problem of motion generation. An example of a motion blending system is given by Emering *et al.* [EME 00].

2.4 Overview of the inverse kinematics problem

As we have seen, the inverse kinematics technique is useful both for the manipulation and animation of articulated figures.

We now focus on the inverse kinematic problem, and on the issues raised by its resolution. Basically, the problem is to determine a joints configuration for which a desired task, usually expressed in Cartesian space, is achieved. For example, the shoulder, elbow and wrist configurations must be determined in order that the hand precisely reaches a position in space. The equations that arise from this problem are generally non-linear, and are thus difficult to solve in general. In addition, a resolution technique must also deal with the difficulties described below.

2.4.1 Dealing with multiple tasks, and resolving conflicts

Specifying a single task at a time is not a very practical way for controlling a complex figure. Hence, it is desirable for a resolution technique to manage multiple tasks simultaneously. As a consequence, it may happen that some of them cannot be satisfied at the same time, whereas they can separately. This conflicting situation must be resolved in some way with an appropriate strategy.

Such questions arise in many other contexts where conflicting decisions have to be combined. Multicriterion optimization techniques have been developed in the field of mathematical pro-

gramming to deal with complex engineering problems that must consider several criteria simultaneously [OSY 84]. Several strategies have been proposed: an obvious solution is to find a compromise, that however satisfies none of the criteria exactly. Weights can be assigned to each criterion in order to define their relative importance. Besides this, so-called hierarchical optimization techniques place the criteria at different levels [OSY 84] [ANA 92]: each criterion is then satisfied as much as possible, but with the constraint of not affecting the satisfaction of the more important criteria. This is clearly a more drastic resolution of conflicts, which is preferable is some situations.

Similar strategies have been proposed in robotics and computer graphics. For the positioning and animation of articulated figures, the weighting strategy is the most employed: some typical examples are given by Badler *et al.* [BAD 87] [ZHA 94] for posture manipulation (see Fig. 2.5) and by Phillips *et al.* [PHI 90] [PHI 91] to achieve smooth solution blending. In the field of robotics however, researchers have developed *task-priority strategies* to precisely arbitrate conflicts by establishing a clear priority order among the tasks [HAN 81] [MAC 85] [NAK 87] [SIC 91].

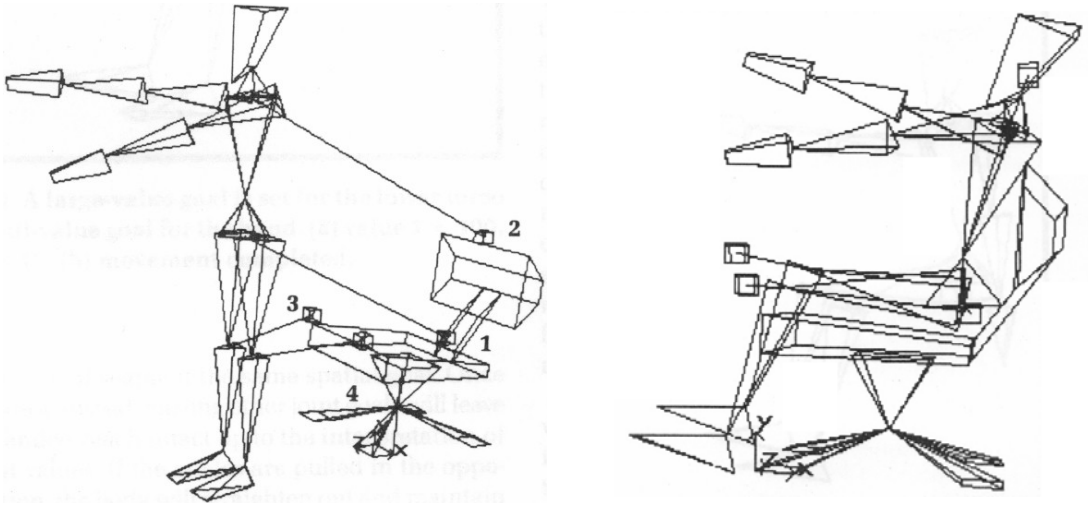


Figure 2.5 Posing a figure on a chair by multiple tasks (source: [BAD 87]). Different weights are associated to each task (w_1=100, w_2=w_3=w_4=10).

2.4.2 Dealing with under-constrained and over-constrained problems

We now discuss the question of the number of solutions. Three situations can occur:

- **The problem has no exact solution.**

  This arises when a task cannot be satisfied (i.e. when a goal is unreachable), or when two or more task are in conflict and cannot be satisfied simultaneously. This is known as an *over-constrained* problem (see Section 2.4.1).

- **The problem possesses a single solution**.

  This case raises no questions, but rarely occurs in practice.

- **The problem possesses two or more solutions, or even an infinity of solutions**.

  This case occurs when there are more degrees of freedom in the structure than tasks: the problem is said to be *under-constrained* or *redundant*. A structure is not redundant by itself, but only with respect to a task that requires less degrees of freedom than those available in the structure. However, in robotics the expression “redundant manipulator” is often used to signify that the usual task leaves freedom of motion to the manipulator. Redundancy is an unavoidable matter of fact when one is concerned with the human body, and it has to be dealt with by an appropriate solution selection mechanism. In robotics instead, robots may be intentionally built with more degrees of freedom than what is necessary for performing the task: the additional flexibility can be for example used to dynamically avoid external obstacles [MAC 85].

We now see typical methods to deal with the extra degrees of freedom.

2.4.3 Dealing with redundancy

In robotics, criteria to exploit the redundancy are typically for joint limits avoidance [KLE 83] and singularities avoidance [LIE 77]. These criteria are not necessarily valid for computer graphics applications: in the standing posture of the human body, many joints (such as the knee) are on their limit, and also at a singular configuration with respect to an end-effector controlling a foot.

Instead, in computer graphics applications, the following criteria have been used to select more natural solutions. A simple and handy criterion is the minimization of the distance with respect to a reference posture, for example to attract to the mid-range posture or a natural rest posture [BOU 94]. More sophisticated criteria are based on mechanical or biomechanical considerations: minimization of joint torques due to the weight has been proposed by Boulic *et al.* [BOU 97b]. A degree of comfort can also be associated to each joint posture, and the total comfort is a criterion that can be maximized [HIR 96] [AYD 99].

Lepoutre [LEP 93] compares upper body postures of a 2D seated operator model resulting from the minimization of different criteria such as articular torques or nearness from articular limits, under constraints imposed by a visibility and reachability task.

While appealing, this redundancy resolution method hides the complexity of determining the “right” function for obtaining realistic postures.

2.4.4 Dealing with joint limits

Another important issue is the respect of joint limits, since their violation may significantly affect the plausibility of a posture. In robotics, joint limits are usually avoided Lie77, because it is not advisable for a robot joint to approach its mechanical limits. This is in contrast with animal or human figures whose rest postures are usually close to some joint limits (consider the human knee during standing, for example). Hence, a resolution method must integrate a mechanism to precisely enforce these important constraints.

2.5 Review of inverse kinematics resolution methods

In this section, an overview of the main resolution methods is given. Because of the vastness of the subject, this is by no means an exhaustive review.

2.5.1 Analytic methods

For very simple robotic manipulators with few degrees of freedom, analytical (or closed-form) solutions can be found by direct resolution of the non-linear equations [PAU 81] [CRA 89]. Robot manipulators are often designed so that analytical solutions to the IK problem exist: this is important for real-time applications because these solutions are very fast to compute. Moreover, the resolution methods are robust. This is also advantageous in computer animation, and several researchers have addressed the case of the anthropomorphic arm and leg: Korein [KOR 85] provides an interesting analytic solution for a 7-DOF arm that deals with joint limits, and Tolani *et al.* [TOL 96] [TOL 00] also discuss similar procedures.

However, analytic solutions do not exist for general articulated structures. In that case, iterative, numerical techniques can be used to solve the IK problem.

2.5.2 The resolved motion-rate method

In his pioneering work, Whitney [WHI 69] introduces the *resolved motion-rate control*, which is the basis of more complex resolution schemes. Given an initial configuration, Whitney performs a linearization of the non-linear equations, characterized by a Jacobian matrix relating differential changes of the joint coordinates to differential changes of task coordinates. The linear system is solved in order to obtain a new configuration closer to the goal. By repeated resolution of this process, the system usually converges to a solution satisfying the constraint if the initial configuration was ”sufficiently close” to it. This method is inspired by the well-known iterative Newton-Raphson method for the resolution of non-linear equations [ORT 70].

The main research issue has been to extend this method to exploit the redundancy of the problem at the differential level. Whitney [WHI 69] uses a generalized inverse [BOUL 71] [BEN 74] to minimize the weighted norm of the variation of joint coordinates. Deo *et al.* [DEO 97] have proposed to use the infinity-norm to obtain the lowest possible magnitudes of joint variations. Liégeois [LIE 77] proposes an extension for the optimization of a criterion expressed in joint space, by exploiting the null space of the Jacobian matrix. Klein and Huang [KLE 83] provide more insight on this topic and on the meaning of the pseudoinverse solution in terms of the Singular Value Decomposition representation [PRE 92] of the Jacobian matrix. Cleary *et al.* [CLE 90] and McGhee *et al.* [MCG 94] discuss the integration of multiple, weighted optimization criteria in the scheme. Hanafusa, Nakamura and Yoshikawa [HAN 81] [NAK 87] further extend the redundancy exploitation with criteria expressed in Cartesian space: hence a secondary Cartesian task that can be satisfied without affecting the primary task. This simultaneous resolution of two tasks with different priorities is known as the *task-priority strategy.* Maciejewski and Klein [MAC 85] improved the resolution scheme of

Hanafusa *et al.* by exploiting pseudoinverse properties. The scheme has then been generalized to an arbitrary number of priority levels by Siciliano *et al.* [SIC 91] and Garziera [GAR 94b].

Another important issue is the management of singularities of the Jacobian matrix. Merely using a pseudoinverse near singular configurations results in high joint increments that wreak havoc in the resolution process. Traditionally, regularization techniques are used for solving such ill-posed problems [NEU 98]. Wampler [WAM 86] and Nakamura *et al.* [NAK 86] independently proposed to use the damped-least squares (or singularity-robust) inverse, a generalization of the pseudoinverse. Afterwards, more sophisticated methods have been proposed by Maciejewski and Klein [MAC 88]. So-called algorithmic singularities appear in the task-priority strategy when two tasks are in conflict: Chiaverini [CHI 94] [CHI 97] introduces a new formulation that overcomes the effects of these singularities.

2.5.3 The Jacobian transpose method

The Jacobian transpose method, discussed by Welman [WEL 93], only differs from the resolved motion rate method in that the transpose of the Jacobian matrix is used instead of its inverse. The method was introduced by Wolovich and Elliot [WOL 84], and extended by Sciavicco and Siciliano to redundant manipulators [SCI 88], and by Das *et al.* for the satisfaction of secondary criteria [DAS 88]. With this method, an iteration can be performed very quickly since no matrix inversion is required. However, because of its poor convergence properties, especially near a solution, the number of steps required to reach the goal may be much higher than with pseudoinverse-based techniques. Hence, on the whole, the Jacobian transpose method is not necessarily more efficient than pseudoinverse-based methods.

2.5.4 Optimization-based methods

Optimization is a vast and fundamental field of numerical mathematics. Many problems can be stated as optimization problems for which a large number of resolution methods have been developed [WAL 75]. The IK problem is no exception: it can be formulated as a constrained optimization problem and thus solved with nonlinear programming methods. For real-time applications, local optimization methods are preferred to the much more expensive global optimization methods [PAR 87] even if there is a risk of being stuck in a local optimum of the objective function.

In computer graphics, Zhao and Badler [ZHA 89] [ZHA 94] use an optimization method [GOL 69] for the manipulation of an articulated figure: a nonlinear function (describing the degree of satisfaction of all constraints) is minimized under a set of linear equality and inequality constraints describing joint limits. An application of this method for the manipulation of articulated figures within the Jack system is described by Phillips *et al.* [PHI 90].

More recent optimization methods such as genetic algorithms have also been applied to the inverse kinematics problem [KHW 98]. It is not clear if such algorithms are a real alternative to traditional numerical resolution methods based on the gradient, especially for structure with a high number of degrees of freedom. However, due to their non-deterministic nature, they

may overcome the problem of local minima.

2.5.5 Database-guided (or model-based) inverse kinematics algorithms

Methods or extensions specific to the human body have also been developed: the human behavior is directly integrated in the process to obtain more realistic, natural-looking, solutions. This goal is of course important for both computer graphics applications and ergonomics. For the purpose of posing a human arm with a given hand position, Wang and Verriest [WAN 98] propose an efficient geometric inverse kinematic algorithm for the human arm that incorporates realistic shoulder limits. Koga *et al.* [KOG 94] use a two-step algorithm: first, an approximate arm posture is found based on a sensorimotor transformation model developed by neurophysiologists, and based on measurements; second, the approximate posture is refined to precisely match the hand position constraint, with the help of a simple numerical method. Aydin *et al.* [AYD 99] use a huge database of realistic, predefined body postures, also with a final refinement process. Wiley *et al.* [WIL 97] interpolate between a set of prerecorded postures to generate real-time reach and walk sequences. In ergonomics, Beck and Chaffin [BEC 92] use an inverse kinematic method based on statistical regression equations developed from a database of measured human behaviours.

The advantage of all these methods is that they are likely to produce more realistic solutions than those resulting from purely constraint-satisfaction based methods. On the other hand, their application is limited by the underlying model or database.

2.5.6 A modal approach for hyper-redundant structures

Chirikjian and Burdick [CHI 94b] introduce a modal approach to solve the inverse kinematics problem of hyper-redundant snake-like robots modeled as a continuous curve. Solving the IK problem with pseudo-inverse methods would be extremely expensive because of the high number of degrees of freedom. Instead of this, they use an analytical curve described by a set of basis functions to model the overall shape of the hyper-redundant chain: the IK problem is then solved in the space of the parameters of these basis functions. Finally, a “fitting” of the discrete chain to this curve is performed in order to compute the joint parameters. This technique could be useful to deal with the spine of a human model.

2.5.7 Other techniques

For completeness, we mention here other methods that have not gained wide acceptance.

The *Cyclic-Coordinate-Descent method* is an iterative heuristic method that attempts to minimize the position and orientation errors by varying one joint at a time. Similarly to the Jacobian transpose method, the cost of a single iteration is very low, but again the convergence rate towards a solution is very poor. A comparison of both methods is done by Welman [WEL 93].

Badler *et al.* [BAD 87] present an elegant recursive constraint satisfaction algorithm for the positioning of complex articulated figures. The algorithm can deal with multiple constraints

simultaneously, and conflicts are resolved with a weighting strategy. The problem is recursively decomposed into smaller sub-problems until they are solvable by a simple inverse kinematic algorithm. This approach has since been abandoned by Badler.

Also to be mentioned is the *reach hierarchy algorithm* proposed by Korein [KOR 85] which is based on the knowledge of the workspaces for each distal subchain of the whole chain. This method indirectly deals with joint limits since they affect the workspace data. However, the computation of these workspaces may be very complex and computationally expensive for complex chains, hence the scope of this algorithm is limited.

Finally, artificial neural networks have been applied to the inverse kinematics problem by learning the non-linear inverse task function. For a review of such attempts, see [OMI 97].

2.5.8 Comparison of resolution methods

As we have seen, there is a large number of methods to solve the inverse kinematics problem. Each has its own advantages and drawbacks. They can be compared on the following important criteria: efficiency (speed), robustness, generality, naturalness of the result (except for robot manipulators), and complexity of the method.

The choice of a resolution method clearly depends on the intended application. For real-time applications, analytic methods are always preferable, if a closed-form solution exists. However, except for simple robot manipulators, this is seldom the case. With this respect, the main advantage of numerical methods is their generality and flexibility: they can deal with arbitrary linkages, and new types of constraints can be easily integrated and combined. The price to pay for this generality is a higher computational cost and complexity of the resolution methods (due to their iterative nature), and a low reliability since the convergence to a solution is not guaranteed. Moreover, for articulated figure positioning, realistic results can only be obtained if a sufficiently detailed model of the figure being manipulated is provided (with joint limits, joint coupling, comfort criteria and so on). This modelling task may require a significant effort. For this reason, database-guided techniques are a simpler way to obtain more realistic results, but their scope is limited to a particular problem, in a particular situation.

2.6 Balance control

Balance control is an essential problem for the realistic computer animation of articulated figures, and of humans in particular. Mobile legged robots, and especially biped robots, also have to deal with this problem in order to perform tasks without compromising their safety [ESP 97] [HIR 98]. While people take for granted the action of keeping balance, it is still a challenge for roboticians to build balanced walking machines, and also for neurophysiologists to understand the mechanisms of balance in human and animal beings [ROB 95].

Here, we focus on techniques for balance control in static equilibrium developed in the computer graphics community, and well-suited to the postural control problem. Clearly, more

advanced methods are required in dynamic situations.

2.6.1 Control of the center of mass

The center of mass is an important characteristic point of a figure. In the Computer Graphics community, Phillips [PHI 91] pioneered the control of the center of mass, for example to change the distribution of body weight on the feet (see Fig. 2.6). Balance of a figure, which is a very important behaviour for the realism, can be obtained by ensuring that the center of mass stays over the polygon of support defined by the position of the feet on the ground. The control of the elevation of the center of mass is also interesting for obtaining a certain kind of posture (e.g. squat, standing, standing on the tip-toes). Phillips achieves the control of the center of mass by constraining the angular values of the ankle, knee and hip joints of the leg that supports most of the weight [BAD 93].

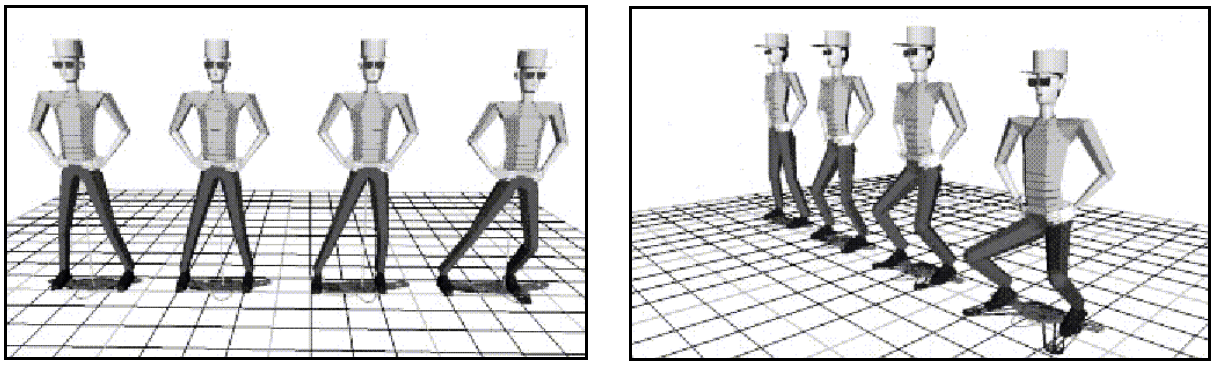


Figure 2.6 Shifting (*left*) and lowering (*right*) the center of mass of Jack (source: [BAD 93]).

A more general approach for the control of the center of mass, called Inverse Kinetics, has been proposed by Boulic *et al.* [BOU 94] [BOU 96]. No special knowledge about the articulated figure is exploited, except the mass information of each rigid body segment. The constraint on the position of the center of mass is treated as any other task, and solved at the differential level with a special-purpose Jacobian matrix that relates differential changes of the joint coordinates to differential changes of the Cartesian coordinates of the center of mass. This method is readily applicable to any articulated figure in single support (Fig. 2.7), and in particular to a complex human figure (Fig. 2.8). Later, the Inverse Kinetics method has been extended to deal with multiple supports [MAS 96] [BOU 97] [BOU 97b], taking into account how the body mass is distributed on the different supporting sites.

Recently, we have extended the control of mass properties to the moments of inertia of the articulated structure [BAE 00]: while this is not required for balance control, il allows to control the distribution of the mass about an arbitrary axis.

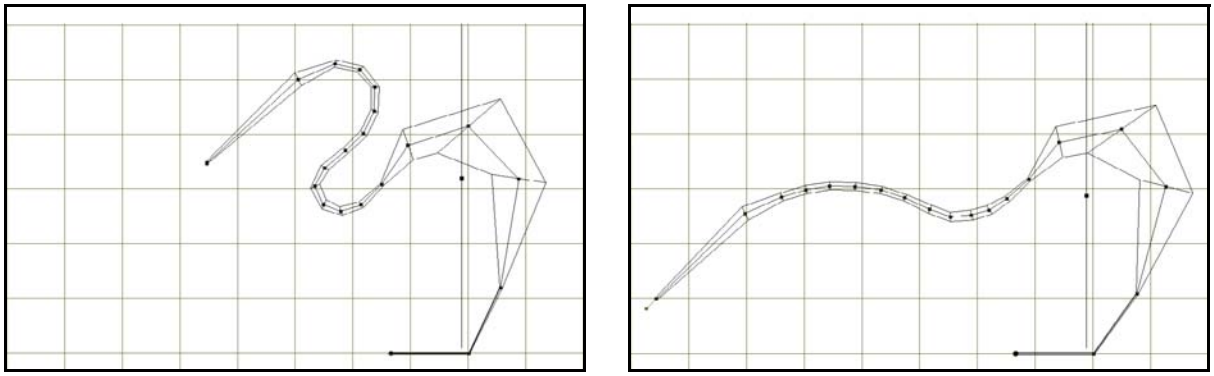


Figure 2.7 The bird keeps its balance while reaching a point with its beak [MAS 96].

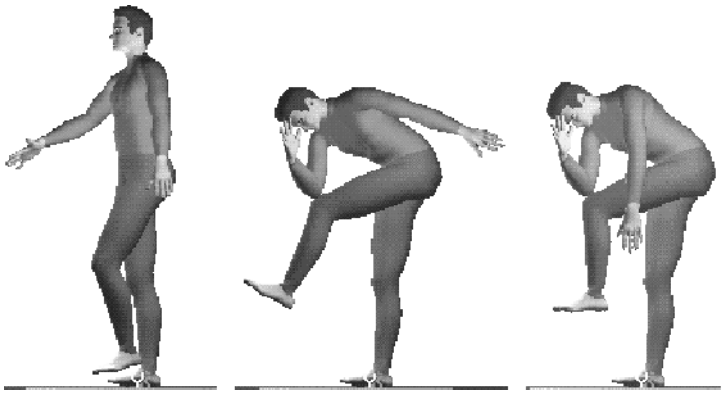


Figure 2.8 Balanced human figures in single support (source: [MAS 96]).

2.6.2 Force exertion and torque control

An approach similar to the Inverse Kinetics method is proposed by Aydin *et al.* [AYD 99b] [AYD 99c]: instead of controlling the center of mass, the total amount of torque perceived by the figure is controlled. The balance of the body may be ensured by constraining the total torque to be null. This method allows to take into consideration external forces in addition to weight, and hence may be used to simulate static pushing or pulling of heavy objects (see Fig. 2.9).

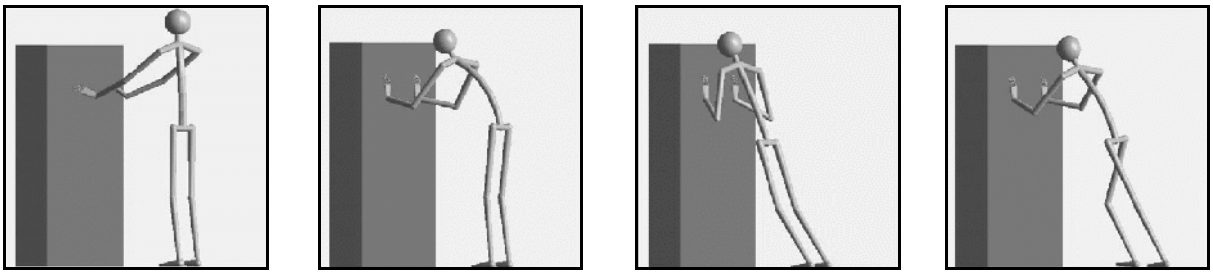


Figure 2.9 Pushing against a heavy object (source: [AYD 99c])

2.7 Collision detection and collision avoidance

In a virtual environment, a simulator must deal with the inter-penetration of objects: first it must be able to *detect* collisions between objects, and then to take the appropriate actions to *respond* to these events, in order to resolve the collision. The first problem is the *collision detection* problem (or self-collision detection, when an object collides with itself), while the second is the *collision response* problem. Many solutions have been developed both in robotics e.g. [CAM 90] and computer graphics [GAR 94], for rigid objects as well as deformable ones [MOO 88]. The collision response can be either kinematic (with a direct change of the position of the bodies) or dynamic (with the application of repulsive forces).

Another difficult problem is the *collision avoidance* problem: it is often desirable that a controlled entity be able to perform a task without colliding with surrounding obstacles or with itself. The obstacle avoidance problem is well-known in robotics: the first solutions were based on an initial planification of a collision-free path for the robot [LAT 91]. However, such solutions are expensive and limit the interactivity of the robot with its environment: hence new approaches have been developed to deal with dynamically varying obstacles in real-time. A classical technique is the artificial potential field method proposed by Khatib [KHA 85]: the manipulator moves in a field of forces, that attract the end-effector to its goal, and that repulse the manipulator parts from the surface of the obstacles. Similar but purely kinematic methods have been proposed by Espiau and Boulic [ESP 85] and by Maciejewski and Klein [MAC 85]: a secondary task controls the point on the robot which is closest to the obstacle, in order to maximize its distance to the surface of the obstacle. Specific methods have also been developed for hyper-redundant manipulators [CHI 92].

The self-collision avoidance is especially interesting when applied to human figures because this provides some degree of self-awareness. In computer graphics, the robotics methods have been adopted and adapted to the human case. For example, Zhao and Badler [ZHA 94b] use a method based on potential fields for the Jack figure, together with inverse kinematics to perform collision response. However, not all possible self-collisions are checked as this would be quite expensive: only the most frequently occurring cases are tested, such as collisions of hands with hands, and hands with the body. Koga *et al.* [KOG 94] proposes a solution to the multi-arm manipulation planning problem: a path planner automatically computes the collision-free trajectories for several cooperating arms in order to manipulate a movable object between two configurations. For controlling a human arm in the process of reaching an object, Huang [HUA 97b] uses a secondary task to keep the elbow outside of the torso, when they become in contact. While inter-penetration is avoided, the resulting motion is not necessarily realistic.

A recent attempt to tackle the self-collision avoidance problem for articulated figures is due to Nebel [NEB 99]: the goal of the system is to automatically generate collisions-free animation sequences based on keyframe interpolation. The principle of his scheme is to detect self-collisions in the sequence generated with classical inbetweening methods, and then to modify these keyframes in order to remove the collisions. Finally, these new keyframes are used for a new

classical interpolation. This is a potentially useful tool for animators, but is not suited for real-time applications that require on-the-fly generation of motion.

2.8 Conclusion

Despite its inherent limitations, the inverse kinematics method is a good choice for the interactive manipulation of figures, especially as we are dealing with postures and not motions. We necessarily rely on numerical resolution methods because we are interested in general-purpose and flexible tools that can deal with complex figures.

Among the numerical methods, there is still a choice between optimization-based methods and the resolved-motion rate method. While the former are black boxes that encapsulate complex numerical algorithms, we prefer to start from a simple method and to improve it. For this reason, we have chosen to use the resolved-motion rate and some of its extensions. It is flexible enough and relatively inexpensive.

As we have seen in this chapter, this well-known robotics technique has also been employed in Computer Animation, but without exploiting the concept of task priorities. In this thesis, we will show how this concept can be efficiently implemented and we will illustrate its benefits for the interactive manipulation of articulated figures.

Chapter 3

The articulated body model

3.1 Introduction

This chapter describes a model of articulated structures, flexible enough to represent a wide variety of figures, including serial robot manipulators as well as human body models. The complexity of a model depends on its intended use. Since we are dealing with the posture of the figures and hence with their kinematics, we focus on the issue of joint modelling, while body segments are assumed to be merely rigid. The essential feature of a joint is that it permits some degree of relative motion between the two segments it connects. Ideal kinematic joint models are defined in order to formalize this permitted relative motion, called *range of motion*, characterized by the number of parameters that describe the motion space, and constrained by joint limits. This is particularly difficult for multiple degrees-of-freedom joints such as the ball-and-socket joints. However, it is important for a motion generator to rely on a precise kinematic model, so that the resulting motions or postures satisfy the anatomic constraints.

3.1.1 Mathematical notation

Before proceeding, we introduce a few operators.

Rotation about an axis by an angle

The rotation (in the right-handed sense) by an angle θ about an axis passing through the origin and whose direction is given by vector $\boldsymbol{a}$, is noted $R_{\boldsymbol{a}}(\theta)$. The Rodrigues formula gives the matrix form of this operator [MUR 94]: if $\boldsymbol{a}$ is not zero, then

$$R_{\boldsymbol{a}}(\theta) = I_3 + \sin(\theta)\left[\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}\times\right] + (1-\cos(\theta))\left[\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}\times\right]^2 \tag{3.1}$$

If $\boldsymbol{a}$ is zero, the rotation matrix is the identity if $\theta = 0$, and is undefined otherwise.

Direct rotation

Given two unit vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, we define $R_D(\boldsymbol{a},\boldsymbol{b}) := R_{\boldsymbol{a}\times\boldsymbol{b}}(\operatorname{acos}(\boldsymbol{a}^T\boldsymbol{b}))$ as the *direct rotation* that transforms $\boldsymbol{a}$ into $\boldsymbol{b}$ (see Fig. 3.1). Note that any rotation whose axis of rotation lies in the bisector plane of $\boldsymbol{a}$ and $\boldsymbol{b}$, with the appropriate angle of rotation, transforms vector $\boldsymbol{a}$ into

vector $\boldsymbol{b}$. The direct rotation is the one with minimum angle of rotation, that is $\text{acos}(\boldsymbol{a}^T\boldsymbol{b})$. Also note that while $R_D(\boldsymbol{a}, -\boldsymbol{a})$ is undefined, $R_D(\boldsymbol{a}, \boldsymbol{a})$ is equal to I_3.

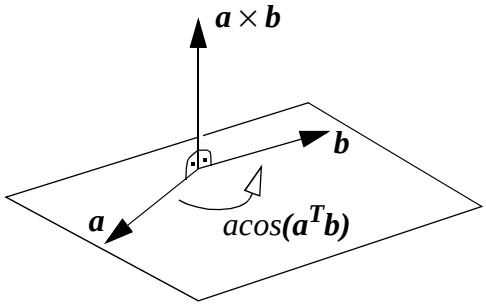


Figure 3.1 The direct rotation $R_D(\boldsymbol{a}, \boldsymbol{b})$ transforms unit vector $\boldsymbol{a}$ into unit vector $\boldsymbol{b}$.

3.2 The hierarchical structure of the body model

An articulated structure is a set of rigid bodies (or segments) connected by joints. A distinction can be made between closed-chain structures, that contain loops, and loop-free open-chain structures. Closed-chain structures, also known as *parallel manipulators* in robotics [MUR 94], are not considered here. It is generally more difficult to deal with closed-chain structures, and fortunately most figures do not contain loops. Moreover, as we shall see later, loops can be ensured in a open-chain structure by means of kinematic constraints.

An open-chain structure can be represented by a hierarchy (or tree) of nodes, that represent either a joint or a segment. This introduces a parent-child relationship among the nodes, where each node has a single parent, except the ancestor of all other nodes, which is the root of the hierarchy (see Fig. 3.2). Each node is placed with respect to its parent node by a local transformation, and the root is fixed with respect to a global reference frame, but can be positioned at will in order to place the figure in the world.

The great advantage of a tree structure is that the direct kinematics problem is very easily solved by a recursive traversal of the tree, starting from the root, and evaluating and concatenating the local transformation matrices.

We define the *upper body* of a joint (or *augmented body* [BOU 94]) as the sub-structure whose nodes are descendants of the joint, and the *lower body* of the same joint as the complementary sub-structure. Of course, these bodies depend on the choice of the root node, but only the configuration of the upper body as seen from the global reference frame depends on the configuration of the joint.

The choice of the root node has no impact on the relative position between segments. However, the position of the root is important in the posture design process, with direct and inverse kinematics. First, the root node represents a fixed frame of the hierarchy. Second, changing the

configuration of a joint affects the overall position and orientation of all its children nodes. Hence a designer should be aware of how the motion propagates into the tree. In a human model, the pelvis is typically chosen as the root node (see Fig. 3.2), but sometimes this may be inappropriate and the hierarchy must be re-rooted.

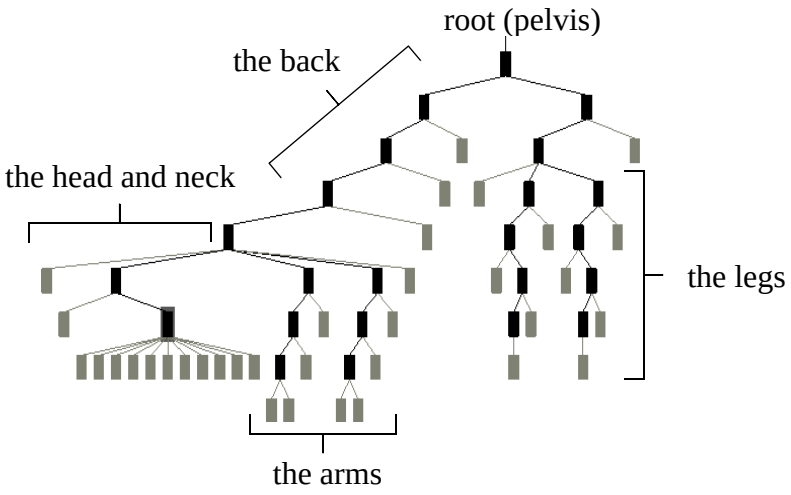


Figure 3.2 A hierarchy of nodes that models a human figure. A black node denotes a joint, while a grey node denotes a rigid geometric object (such as a cylinder) attached to a joint.

3.2.1 Re-rooting the hierarchy

The process of changing the root of the hierarchy is called re-rooting. While most robotic manipulators have a fixed base, animal and human figures are mobile and no part is permanently fixed. The possibility of re-rooting a hierarchy is thus useful when the current root becomes unconstrained and that another body part needs to be fixed. With the human model for example, if the pelvis is allowed to move, it cannot be the root of the hierarchy, while a fixed supporting site such as a foot is a better candidate.

The connectivity is not affected by the re-rooting operation, but the parent-child relationship is inverted for the nodes that lie between the new and the old root. Hence their local transformation matrix must be inverted, in order to keep the same posture when the hierarchy is re-rooted.

3.3 Parametrization of the body configuration

The body posture can be modified simply by changing the local transformation matrix of each joint node. This allows any posture to be set for the articulated figure, even unfeasible ones since arbitrary rotations and translations can be specified. This is a problem for general-purpose motion generation engines that do not have any implicit knowledge about the model they are dealing with. Constraints must be introduced to specify the motion that each joint is capable of. This can be achieved by the use of a set of generalized coordinates that parametrize the

local transformation matrix of all joints. Inequality constraints on these coordinates further constrain the set of allowed postures, and allow to define the range of motion of each joint. Given these constraints, a motion generation engine is prevented from providing unrealistic postures, since they are specified only through the set of generalized coordinates.

Introducing more degrees of freedom in the body model than what is strictly necessary augments the chance of obtaining unrealistic postures. Hence the choice of pertinent joints to be included in the body model must be done carefully.

3.3.1 The problem of joint coupling

Moreover, joints may be dependent on each other. This coupling (of motion and limits) can be integrated directly in the body parametrization, with the concept of *joint group* [BAD 93], or at the application level, with kinematic constraints resolved by an inverse kinematics engine (for example, the scapulo-thoracic constraint [MAU 00]). Here, we follow the second approach: it has the advantage of simplifying the body definition, but then anatomical constraints must be ensured (see Section 3.8).

3.3.2 Generalized coordinates

The state vector of generalized coordinates $\boldsymbol{q} = (q_1, \ldots, q_n)^T$ expresses the configuration (or posture) of the articulated structure, which is the configuration of all its joints. Hence the set of all possible configurations is termed the *joint space*. The total number of degrees of freedom is n, while the number of joints is k, with $k \leq n$. Since we have multiple-DOF joint models, there is no simple one-to-one correspondence between a degree of freedom and a joint: hence the generalized coordinates of the jth joint are noted $\boldsymbol{q}_j$, whose number of degrees of freedom is noted n_j. Hence:

$$\sum_{j=1}^{k} n_j = n$$

As example, for a typical human figure, the number of degrees of freedom (without the fingers) is about n=50, while the number of joints is about k=20.

Now the problem is reduced to find a parametrization for each joint, and to impose limits on its motion. For this purpose, a minimal set of joint models must be defined.

3.4 The joint models

The first purpose of a joint model is to compute a local transformation matrix, noted $L(\boldsymbol{q}_j)$, as a function of the set of generalized joint coordinates $\boldsymbol{q}_j$. These parameters represent either translational or rotational degrees of freedom. However, translations are not considered here, since they do not appear in the human body (except to a very limited extent).

The second purpose of a joint model is to define limits on the joint coordinates $\boldsymbol{q}_j$, in order to avoid self-collisions between adjacent segments and also the specification of unfeasible postures.

The main joint models that are needed are enumerated below.

3.4.1 The revolute joint model (one DOF)

The revolute joint is the simplest joint model that allows rotational motion: rotation occurs about a single, fixed axis. We arbitrarily choose the axis of rotation to be the ***z*** axis of the local joint frame. Its natural parametrization is the angle of rotation θ, with respect to a reference configuration: hence, the local transformation matrix of the joint is simply $R_{\boldsymbol{z}}(\theta)$.

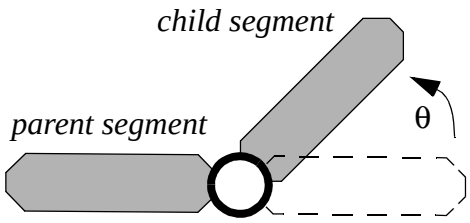


Figure 3.3 A revolute joint connecting two segments, performing flexion motion.

A revolute joint is typically used as a hinge joint (for flexion purposes). If the axis of rotation is aligned with the distal (moving) segment, a rotation results in a twist of the distal segment about itself. Hence the revolute joint model may be used for two conceptually different motions: flexion and twist.

Because of its simplicity, and due to mechanical design considerations, the revolute joint is by far the most used joint in robotic manipulators. In human modeling, it is a convenient model for the flexion of the interphalangeal joints of the hands, for example. It is tempting to combine two or three revolute joints at the same point with different axes of rotation to model more complex joints such as the shoulder. However, such mechanisms exhibit the gimbal lock problem discussed in Section 3.5.1, which is an impossibility for the joint to move in a certain direction when two axes of rotation become aligned.

3.4.2 The elbow joint model (two DOF)

A situation where two revolute joints can be easily combined to model a more complex joint is for joints with a flexion/extension motion, combined with a twist of the outgoing segment. Typical examples in the human body are the elbow and knee joint. The two axes or rotation are independent, and joint limits can also be specified independently one each degree of freedom.

3.4.3 The ball-and-socket joint model (three DOF)

A ball-and-socket joint possesses three rotational degrees of freedom. Hence, it is the most mobile of the purely rotational joints. It allows an *axial* motion (or *twist*) of the segment (one DOF), as well as a *spherical* motion (or *swing*) that determines its direction (two DOFs). A mechanical illustration of this joint is given in Fig. 3.4. By convention, in the following discussion the moving segment is aligned with the ***z*** axis of the local joint frame.

Ball-and-socket joints are used to model articulations such as the human shoulder and hip. Joints such as the wrist can also be modelled, but the twist motion must be forbidden: hence they have two degrees of freedom only.

The accurate kinematic modelling of such articulations is a difficult task. First, a clear mathematical description of the allowed relative motion must be given by a proper parametrization: because of the complex non-Euclidean nature of rotations, this must be done carefully, or one may incur in the problem of singularities. Second, the range of motion should be constrained by some joint limits, to restrict the parameter space to some more realistic subset. The situation is complex, because the boundaries on the three independent parameters are generally coupled. These two aspects are discussed in more detail in the following sections.

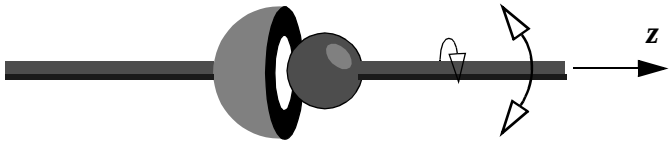


Figure 3.4 A ball-and-socket joint. The outgoing segment is aligned with the ***z*** axis.

3.5 Parametrization of the ball-and-socket joint

3.5.1 Parametrization of rotations

The motion space of a ball-and-socket joint is the set of 3D rotations. There are many well-known parametrizations of rotations. The most widely used are:

- the *Euler angles* (the angles of three successive rotations about principal axes)
- the *unit quaternion* (also known as the *Euler parameters*)
- the *axis-angle* vector (also known as the *exponential map* or *versor*).

Good comparisons of such parametrizations for the purposes of animation of articulated bodies can be found in [GRA 98] [WAT 92]. As noted by Grassia [GRA 98], no single parametrization of rotations is best. Each one possesses its advantages and drawbacks, with respect to the intended application. Hence, it is likely that several parametrizations be used simultaneously, with conversions between them. For example, the unit quaternion is ideally suited for interpolation [SCH 85] [WAT 92], while the axis-angle vector is a more appropriate parametrization for differential control with inverse kinematics [GRA 98]. Euler angles would not be a good choice in both applications.

An important point to consider when comparing two parametrizations is the presence of singularities. Singularities are locations in the parameter space that result in the same orientation of the joint. Sometimes these singularities are purely mathematical and only result from the choice of parametrization, but they may also reflect a physical reality. In that case, we encoun-

ter the problem known as gimbal lock [WAT 92] [GRA 98]. Because of the problems induced by the singularities not only at the singular point but often also in their neighborhood, the configuration of a joint should always be kept as far as possible from these points.

As a matter of fact, any three-dimensional parametrization of rotations present at least one singularity [MUR 94]. Those of the Euler angles and of the exponential map are discussed in [GRA 98], and will be recalled later. The unit quaternion parametrization is singularity free, but at the cost of using four parameters instead of three, with a quadratic constraint of unitary norm that must then be ensured [GRA 98].

3.5.2 Parametrization for the purpose of range of motion definition

For the purpose of defining a range of motion, an appropriate parametrization is needed. Certainly, one can impose limits on any parametrization. For example, it is possible to impose limits on Euler angles or on quaternion parameters. For example, Lee [LEE 00] describes simple analytical constraints (such as axial, spherical or conical constraints) enforced directly in quaternion space. More complex constraints can then be defined by combining the simple ones with boolean operators. While simple and elegant, this method is not precise enough for an accurate modelling of the limits of complex joints such as the shoulder, and placing more complex meaningful limits on quaternions is difficult.

To simplify the problem, the joint limits may be decoupled. For example, independent limits may be specified on each Euler angle, or on each element of the axis-angle. However, the resulting range of motion can hardly match real motion ranges with sufficient precision [MAU 00].

For the purpose of defining a range of motion, neither the axis-angle nor the unit quaternion reflect the intuitive decomposition of the rotation into a swing and a twist component. Euler angles do, since the third angle may be used to perform the twisting motion. However, in the following sections we see that the first two Euler angles can be replaced by an axis-angle vector with zero component along the $\boldsymbol{z}$ axis: this allows to alleviate the problem of singularities that affects the Euler angles.

3.5.3 The swing and twist decomposition of an orientation

Intuitively, the orientation R of a ball-and-socket joint can be thought as being composed of a swing component, that controls the direction of the limb directly attached to it, and a twist component that lets the limb rotate about itself [KOR 85] [GRA 98]. This may be written as:

$$R = R^{Twist} R^{Swing}$$

The twist component is easily parametrized by a single angle of rotation, noted τ: hence, $R^{Twist} = R_{\boldsymbol{z}}(\tau)$. However, this rotation must be done with respect to a well-defined orientation, here called the *zero twist reference orientation*. In fact, this reference orientation merely results from the swing rotation, and is not necessarily a good reference. Hence a relative twist,

τ_{offset}, as a function of the swing parameters, can be added. An example of such an offset function is given by Badler *et al.* [BAD 80].

The purpose of the swing motion is to orientate the outgoing limb in a prescribed direction given by a unit vector ***d***. To transform the **z** vector into the ***d*** vector, a rotation matrix R^{Swing} must be defined. We consider two solutions.

- The first is to perform two successive rotations, for instance one about the ***x*** axis and then a second one about the rotated ***y*** axis: $R^{Swing} = R_y(\beta)R_x(\alpha)$. This is equivalent to the first two rotations of the ZYX Euler angles sequence [MUR 94] (Fig. 3.5-A).
- The second is to perform a single, direct rotation: $R^{Swing} = R_D(\boldsymbol{z}, \boldsymbol{d})$ (Fig. 3.5-B). Note that the axis of rotation always lies in the ***x-y*** plane.

The second solution has been used by Korein [KOR 85] and Grassia [GRA 98]. However, Korein parametrizes this rotation with two angles, called *half-plane* and *deviation*, that are the spherical coordinates describing the direction vector ***d***, while Grassia uses the ***x*** and ***y*** components of the axis-angle, here noted s_x and s_y.

As already noted by Korein, the difference between the two rotations lies in the final twist about the ***d*** axis, which is given by the different orientations of the rotated ***x*** and ***y*** vectors. Table 3.1 shows a sampling of the zero twist on the sphere for the two parametrizations: the outgoing arrow at each point on the sphere indicates the direction of the rotated ***x*** axis, which is taken as a reference to indicate the twist.

As mentioned before, singularities of a parametrization must also be considered, because the presence of singularities may be problematic for several applications. For the purpose of defining a range of motion, the twist component is affected by a singularity of the swing component: for example, no zero twist may be defined at a singularity, since an infinity of twists are possible. An arbitrary twist may be assigned to this point, but there is still a discontinuity with respect to its neighborhood. Table 3.1 compares the position of the singularities on the sphere, while the corresponding locations in the parameter space are shown in Fig. 3.6, and the next two sections discusses and compares them.

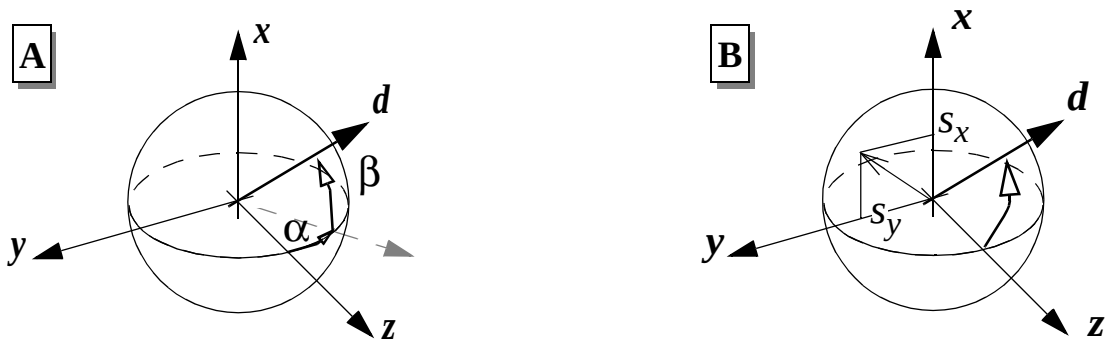


Figure 3.5 Euler angles (**A**) and axis-angle (**B**) parametrization of the swing motion.

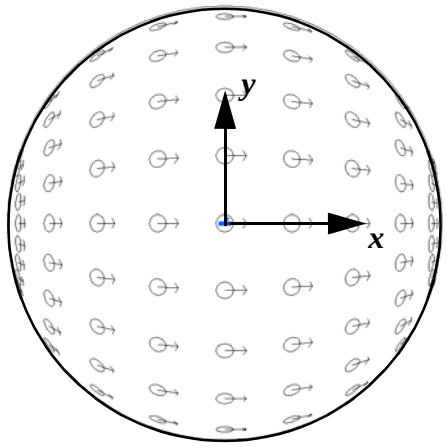
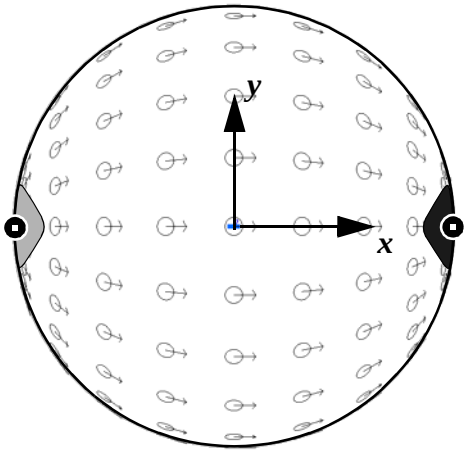
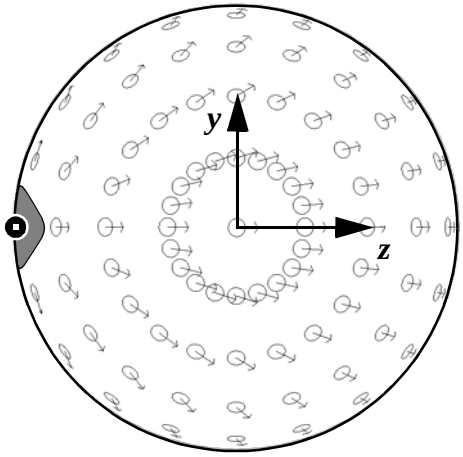
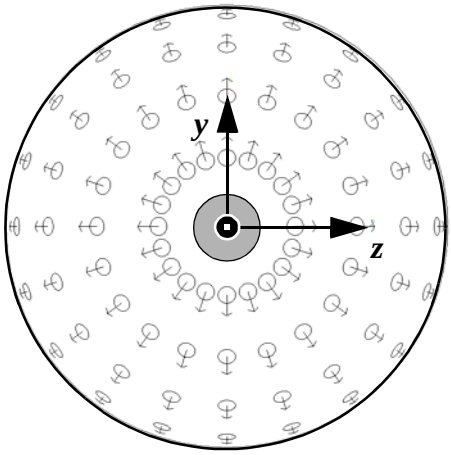
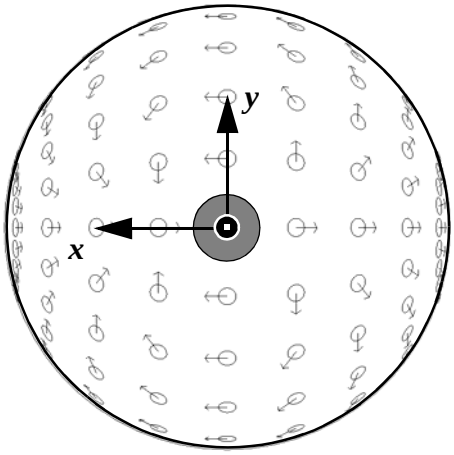
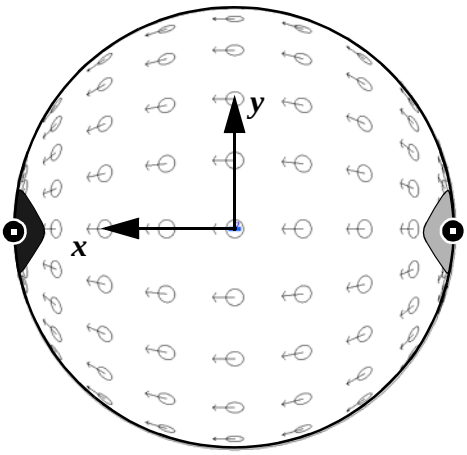
View	Axis-angle parametrization of swing (with one singularity)	Euler angles parametrization of swing (with two singularities)
Front		
Side		
Rear		

Table 3.1 Comparison of "zero" twist and singularities (●) for two parametrizations of swing. The vicinity of each singularity is marked by a grey area with a different shade.

Figure 3.6 Singular locations of the Euler angles parametrization (at $\beta = \pm\pi/2$) on the left, and of the axis-angle parametrization (a circle of radius π) on the right.

3.5.4 Singularities of the XY Euler angles swing parametrization

This parametrization possesses two singularities: one at $\beta_1 = \pi/2$ and another at $\beta_2 = -\pi/2$. In Cartesian space, these singularities correspond to directions $\boldsymbol{d}_1 = (1\ 0\ 0)^T$ and $\boldsymbol{d}_2 = (-1\ 0\ 0)^T$ respectively, and any twist is possible there. Furthermore, moving close to these directions results in large variations of twist. For example, moving along a closed path close to, and around the singularity, results in a complete rotation of the segment about itself (i.e. a twist of 2π radians).

Note that another convention of rotation axes could have been chosen. For example, one can perform a first rotation about the ***z*** axis, and then a second one about either the rotated ***x*** or ***y*** vector. In this case the singularities are located on directions $\boldsymbol{d}_1' = (0\ 0\ -1)^T$ and $\boldsymbol{d}_2' = (0\ 0\ 1)^T$. This is equivalent to our original choice, up to a rotation by 90^o about the ***y*** axis, but having a singularity exactly at the initial configuration is not a good idea.

To understand the meaning of the singularities, consider an universal joint, made as a sequence of two revolute joints whose axes of rotation are orthogonal, as shown in Fig. 3.7. A rotation about the ***x*** axis or the ***y*** axis changes the direction of the outgoing segment, and apparently no twisting is performed. However, this is not always true. When $\beta = \pm\pi/2$, which is the angle of rotation about the ***y*** axis, the outgoing segment becomes aligned with the ***x*** axis (Fig. 3.8): as a consequence, a change in α does not change its direction anymore, but its twist. Actually, any twist is possible in this direction, but the segment cannot move up and down anymore. This phenomenon is known as gimbal lock, and is a well-known flaw of Euler angles [WAT 92]. Also note how the vertical swing component (along the ***x*** axis) gradually transforms into a twist of the outgoing limb, as the singular configuration is approached. This shows that the problem not only exists at the singularity, but also in its vicinity.

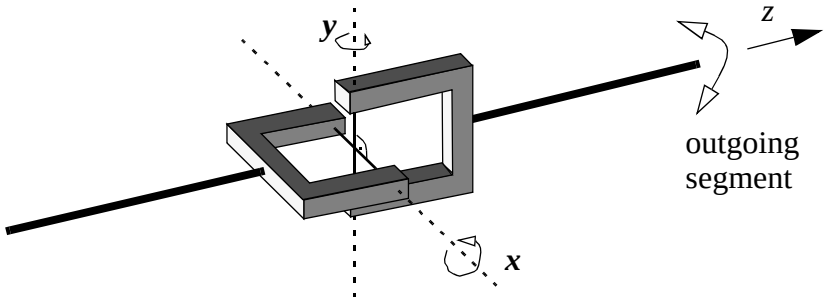


Figure 3.7 Illustration of the *universal* joint, with two orthogonal rotation axes.

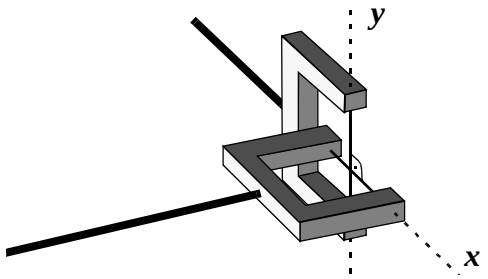


Figure 3.8 The universal joint in a singular configuration ($\beta = \pi/2$): the outgoing segment becomes aligned with the ***x*** axis of rotation.

Ball-and-socket joints are often built as a series of three revolute joints with intersecting axes: two for swinging (as in a universal joint), and one for twisting. Thus, it also experiences the singularities of the universal joint. An example of this is the very common joint used to transmit a torque for the control of a window blind. At a singularity however, a twisting torque is completely transformed in a swinging torque applied to the outgoing segment. Thus no twist can be transmitted anymore. Moreover, in a dynamics simulation, this singular configuration could lead to numerical problems: a torque applied about the two aligned axes of rotation of the joint may result in an infinite acceleration since mass is usually not present between the axes of the same joint. Fortunately, for simulation purposes we can choose another parametrization, such as the axis-angle parametrization.

3.5.5 Singularity of the axis-angle swing parametrization

The axis-angle possesses only one singularity on direction $\boldsymbol{d} = (0\ 0\ -1)^T$, where $s_x^2 + s_y^2 = \pi^2$. Again, any twist is possible there. However, this singularity is more “severe” since a closed path close to, and around the singularity, performs two complete rotations of the segment (i.e. a twist of 4π radians).

A geometric interpretation of this singularity is the following. Consider the problem of finding the shortest path between two given points lying on a unit sphere. The solution is the great arc connecting these two points. This solution is always unique, except when we deal with two

antipodal points, since there is an infinity of great arcs between them. This corresponds to the singular situation of our swing parametrization. Now, when we are close to this singular situation, notice how a small change of one of the two points may result in a dramatic change of the solution. Hence solving the problem at the singularity, for example by choosing an arbitrary solution among the valid ones, does not necessarily solves the problem in the vicinity of the singularity.

To summarize, the axis-angle parametrizations is preferable to the Euler angles parametrization, since it is easier to avoid one single singular point than two antipodal singular points on the sphere. To be as far as possible from the singularity, the motion range should be centered about the ***z*** axis in its default configuration, or at least the singular point should not be part of the motion range.

3.6 Joint limits for the ball-and-socket joint

Based on the swing and twist decomposition, it is possible to impose independent limits on both components. The limits of the swing component are best visualized as a curve on a sphere centered at the joint center. This curve delineates the valid region for the outgoing limb, and can be seen as the directrix of a general conical surface whose vertex is the center of the joint. In the next two sections, we review two possible methods for defining this curve, and the limits of the twist component are discussed.

3.6.1 Swing limits: the spherical ellipse and the spherical polygon

An analytical method to set limits on the swing component is to use a function $f(s_x, s_y)$ which is negative only for a valid swing (s_x, s_y). A simple example given in [GRA 98] is an ellipse with semi-axes r_x and r_y, that describe the maximum angle of rotation around the ***x*** axis and the ***y*** axis respectively: in this case, the function is $f(s_x, s_y) = (s_x/r_x)^2 + (s_y/r_y)^2 - 1$, with $r_x < \pi$ and $r_y < \pi$. This results in a "spherical" ellipse in Cartesian space (see Fig. 3.9). Its advantage is that, with a minimum of parameters, a smooth and intuitive boundary can be defined for the swing component.

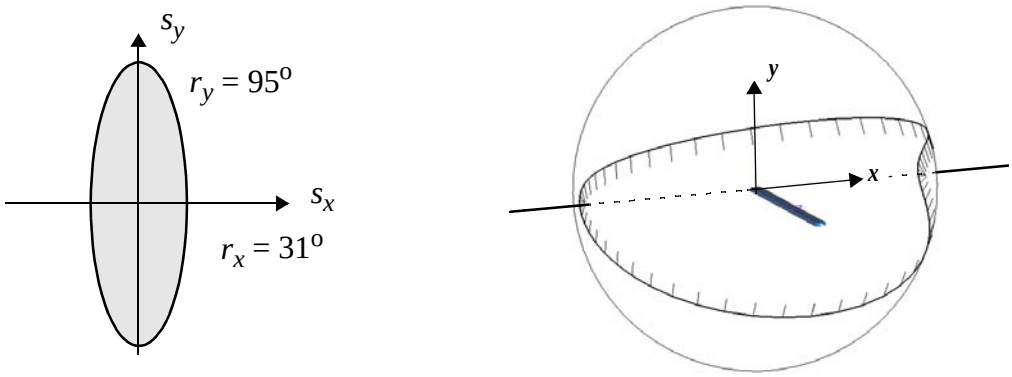


Figure 3.9 An example of spherical ellipse (the ticks indicate its "inside" region).

In his excellent book, Korein [KOR 85] suggests the use of a spherical polygon as directrix for the limiting cone. The edges of the spherical polygon are great arcs connecting an ordered set of vertices lying on a unit sphere. A great arc is the shortest path on the sphere that connects two vertices (it is a *geodesic*). Note that, by definition, only great arcs that subtend angles less than π radians are possible. The order of the vertices defines an inside region: inverting this order swaps the inside and outside regions of the polygon (see Fig. 3.10).

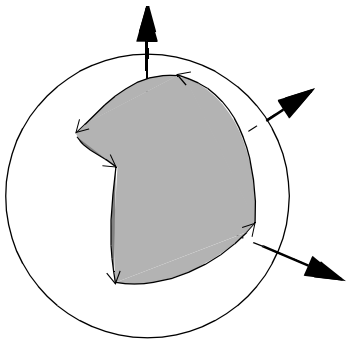


Figure 3.10 A spherical polygon with five directed edges.

Of course, spherical polygons are more general than spherical ellipses. They are also more complex to deal with. A similar method has been used by Maurel [MAU 00] and Aydin *et al.* [AYD 99], but with planar polygons. The limitation is that the possible motion ranges are less general than those obtained with spherical polygons. However they may suffice for the human joints, and the *point-in-planar-polygon* test algorithm is much simpler than its spherical counterpart (described in [KOR 85]).

More general boundaries can be obtained simply by using several non-overlapping spherical polygons: the set of admissible points on the sphere is then the set of points that are inside all the spherical polygons. This allows to create holes in the admissible space. However, this possibility is not needed for human articulations.

3.6.2 Twist limits

The twist motion possesses a single degree of freedom, parametrized by the angle of rotation τ about the outgoing segment. The limits on this parameter are relative to a “zero” twist, that results from a pure swing motion (see Table 3.1).

In a globographic representation involving a twist freedom, the twist range of motion may be visualized by the symbol depicted in Fig. 3.11: the valid twist range indicates the orientations that can take the reference vector (here, the $\boldsymbol{x}$ basis vector of the joint frame).

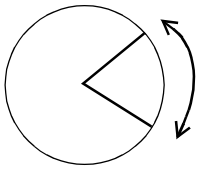


Figure 3.11 Representation of twist range of motion, for a given swing.

In general, twist limits may depend on the swing component. For this reason, the twist limits may be defined as functions of the swing component: $\tau_{min}(s_x, s_y)$ and $\tau_{max}(s_x, s_y)$, with the requirement that $\tau_{min} \leq \tau_{max}$, for all valid swings s_x, s_y.

3.6.3 An example of shoulder boundary with swing and twist components

Fig. 3.12 shows two boundaries for the shoulder complex, based on a spherical ellipse on the left and on a spherical polygon on the right. The distal segment (the arm) is shown in its default position. The twist limits are constant over the range of swing motion (the twist motion range is about 105^o). However, this is only a gross approximation since Wang *et al.* [WAN 98b] have shown that the twist limits depend on the position of the arm, and the twist range of motion can vary between 104^o and 160^o on average. The spherical polygon is based on the results of Engin [ENG 89].

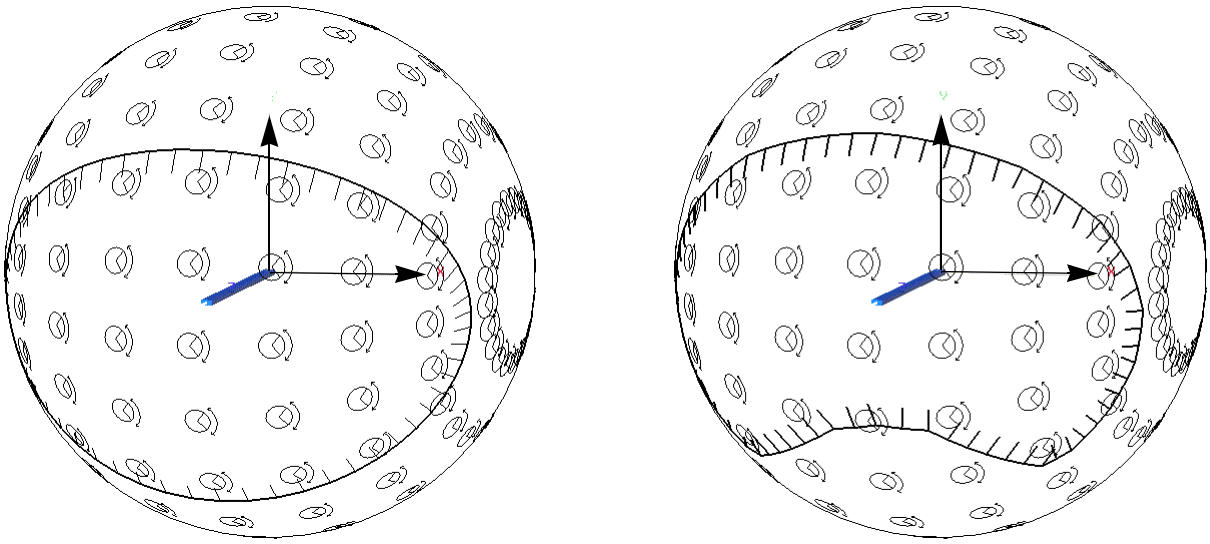


Figure 3.12 Shoulder motion range, with a spherical ellipse (*left*) and a more precise spherical polygon (*right*). The twist motion range, sampled on the sphere, is constant in this example.

3.7 Special problems in the positioning of the ball-and-socket joint

Once a joint is parametrized, its configuration can be specified or modified only through its set of parameters $\boldsymbol{q}_j$. Incremental modifications (that is, with respect to the current configuration) occur frequently: for example, if a joint has exceeded its joint limits, it must be adjusted to its closest valid position. Numerical inverse kinematics resolution techniques also incrementally modify the configuration of the joints, until a solution is found.

When a modification is achieved in an incremental fashion, a few pitfalls arise for the ball-and-socket joint model because of its complex nature. They are mentioned below.

3.7.1 The occurrence of induced twist

Direct rotations are a desirable way of performing a swing, since the **z** vector of the local joint frame is rotated to reach a given direction, without any twist being performed about that axis. However, a direct rotation performed from a direction other than the default **z** vector not only

affects the swing component of the joint but also its twist component, by an amount called the *induced twist*, noted $\tau_{induced}$. This phenomenon has been previously discussed by Korein [KOR 85], and occurs independently of the parametrization used for the swing component. It is not a major problem, but it has to be dealt with, otherwise unwanted twist may be introduced. As shown in Fig. 3.13, for arbitrary unit vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, we have that $R_D(\boldsymbol{z}, \boldsymbol{a})R_D(\boldsymbol{a}, \boldsymbol{b}) \neq R_D(\boldsymbol{z}, \boldsymbol{b})$. A solution is to extract the induced twist from the rotation matrix $R_{\boldsymbol{z}}(\tau_{induced}) = R_D(\boldsymbol{z}, \boldsymbol{a})R_D(\boldsymbol{a}, \boldsymbol{b})R_D(\boldsymbol{b}, \boldsymbol{z})$, and then to subtract it from the twist variable τ [KOR 85]. In [BAD 93], it is shown how the induced twist can be computed, and thus removed, when using the Euler angles parametrization. Another possibility is to perform the update of the joint orientation with a direct rotation, and then to convert the resulting orientation back into the swing and twist parametrization (see Appendix C), to check for possible joint limits violations.

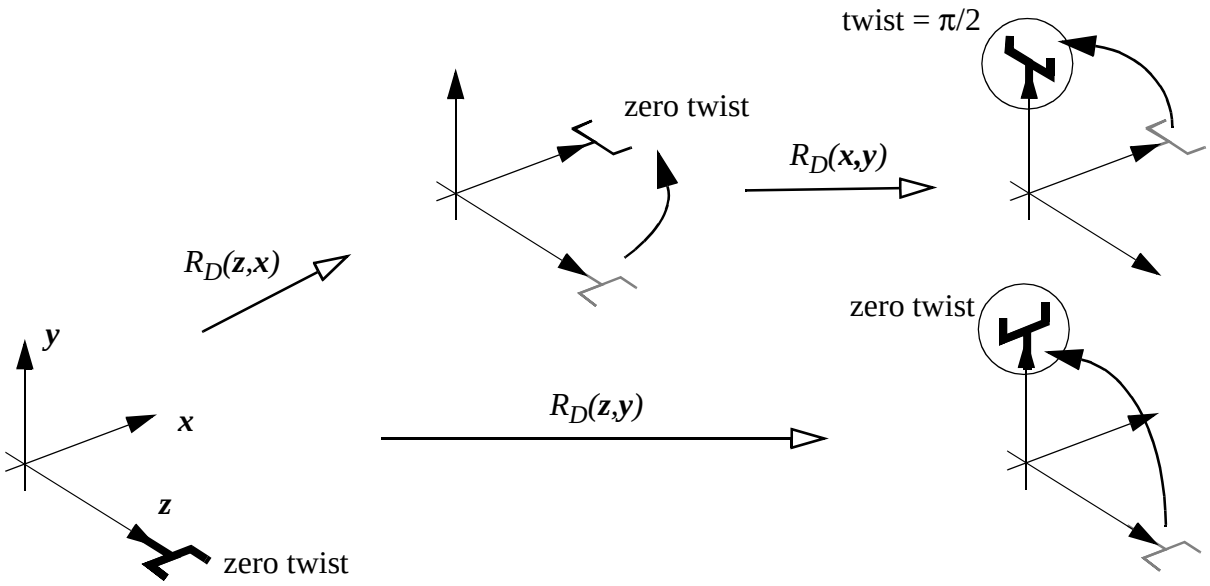


Figure 3.13 Reaching a target direction (here, the ***y*** axis) either with a single direct rotation (bottom) or with a combination of two direct rotations (top) does not result in the same final twist. The difference is called the *induced twist* (here: π/2 radians).

3.7.2 Clamping to the joints boundaries

When a given joint configuration must be checked against the boundaries for possible violation, the first operation to do is to compute its equivalent swing-twist decomposition (see Appendix C). Then a first test can be performed to check if the swing is in its range of motion. If it is not, most applications require that an alternative, close, valid configuration be supplied. For this purpose, the outgoing segment is orthogonally projected on the swing boundary, and a direct rotation is performed to rotate the segment into the new, valid direction: this ensures that the closest possible valid configuration is found. A priori, clamping on a spherical ellipse can be performed directly in the axis-angle space of the swing component, by projecting the invalid

swing on the 2D ellipse (see Appendix D). This may be simpler than performing the computation in Cartesian space: however it can be objected that the use of the Euclidean distance to find the closest projection in the axis-angle space does not result in an exact orthogonal projection in Cartesian space, as could be naively expected. As a consequence, the segment may unnecessarily "slide" on the spherical ellipse. Moreover, since only the swing component is adjusted, the twist induced by this relative adjustment must then be compensated.

Once the clamping of the swing component (s_x, s_y) has been performed, the twist angle τ must be clamped into its valid interval $[\,\tau_{min}(s_x, s_y)\,,\ \tau_{max}(s_x, s_y)\,]$, so that a final, valid orientation can be computed and assigned to the joint.

3.7.3 Dealing with concave joint limits

If the joint boundaries are concave, another problem arises when moving from a valid configuration to a new one: the great arc connecting those two points may not be fully contained in the valid region, even if the two extremities do. This may cause some unexpected behaviors: for example, the distal segment abruptly moves from a valid region to another, which are close in space but that are connected only by a long path (as shown on Fig. 3.14).

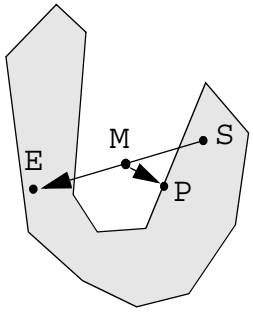


Figure 3.14 Computation of the next valid point (***P***), from a valid initial configuration (***S***) and a computed new configuration (***E***).

A correct solution would be to check if the great arc is entirely contained in the valid region. However this solution is complex and is prone to numerical inaccuracies when an extremity is very close to the joint boundaries. Therefore, we suggest a much simpler, yet approximate, solution. Instead of testing only the final configuration (***E***), we propose to test the middle point (***M***) of the great arc as well. If both final and middle point are valid, then we keep the final configuration. However, if the middle point is not valid, we keep its projection (***P***) on the joint boundary. This simple discretization of the arc has the effect of reducing to a minimum the unexpected behaviors described above, at least for reasonably small increments.

3.8 Coupling between joints

As said at the beginning, the joint models are independent on each other. Hence, possible coupling, due for example to the presence of tendons or muscles spanning several joints, must be defined as additional constraints. These constraints must then be integrated in the posture manipulation tool. We distinguish between equality constraints between joint parameters, and inequality constraints to model coupled joint limits.

3.8.1 Coupling between joint parameters

A coupling between joint parameters means the motions of two or more joints become depen-

dent: this has the consequence of reducing the number of actual degrees of freedom of the model.

A simple example is given by Rijpkema *et al.* [RIJ 91]: they use the following linear relationship that approximately holds between the angles of the two distal joints in a finger other than the thumb (see Fig. 3.15):

$$\theta_{f1} - \frac{2}{3}\theta_{f2} = 0$$

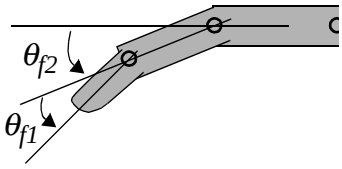


Figure 3.15 Finger model.

However, this constraint is no more valid when forces are applied on the finger.

A more difficult case is the spine, with its 24 vertebrae coupled by ligaments. Modelling each vertebra as a joint without taking into consideration the coupling that exists among them is not recommended, since too much freedom is left to the animation engine. This choice usually leads to unrealistic spine postures. It is preferable to use only a few uncoupled joints strategically placed on the spine, in order to have a more realistic rigidity of the system, even if this solution is not ideal too. More advanced attempts to model the back with its couplings have been done by Monheit *et al*. [MON 91].

3.8.2 Coupling between joint limits

Because of muscles and tendons spanning multiple joints, couplings also arise between the joint limits of adjacent joints. In the human body, such couplings exist between the knee and the ankle, or between the interphalangeal joints of adjacent fingers.

Probably the most apparent coupling of this kind occurs between the knee and hip joints, spanned by several muscles that cannot stretch enough to allow simultaneous full range of motion at both joints [KRE 90]. Hence the leg cannot adopt all the postures allowed by the skeleton, since it is constrained by the limits imposed of muscle lengths.

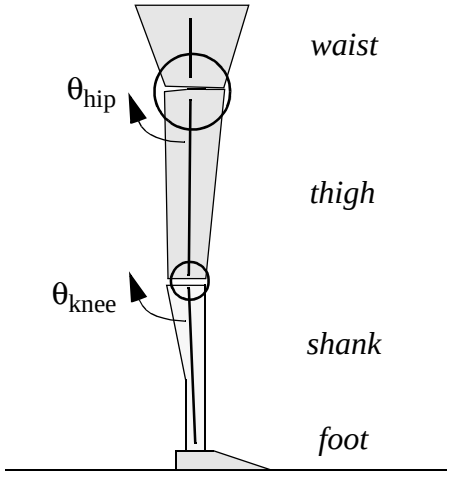


Figure 3.16 Hip and knee joints (side view of the default posture).

Let us note θ_{hip} as the swing component of the hip that moves the leg in the sagittal plane, and $\theta_{knee} \geq 0$ as the flexion angle of the knee. In the default posture, both angles are equal to zero (see Fig. 3.16). While the upper limits on these angles can be considered constant, the lower limits become coupled when the muscles spanning both joints reach their length limits. We have measured that the joint angles are roughly constrained as follows:

$$\theta_{hip} + \theta_{knee} \geq -70° \qquad (3.2)$$

The constant may vary according to the flexibility of the muscles involved. This inequality constraint evaluates a leg posture and determines its validity (see Fig. 3.17).

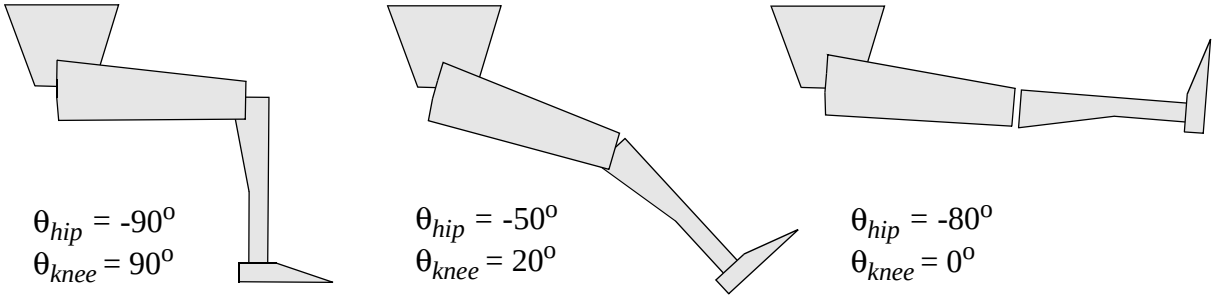


Figure 3.17 Three leg postures. The posture on the right is invalid since it violates Eq. (3.2).

3.9 Integration of additional data

To complete the body model, additional data can be attached to the nodes in the hierarchy for different purposes.

3.9.1 Simple geometric primitives

The simplest representation of an articulated body is with lines connecting the centers of adjacent joints: this is known as stick figure. However, this mode does not convey enough information about the current posture of the body: for example, the twist of a body segment about itself cannot be seen. Moreover, the body volume is neglected.

A less crude appearance can be obtained with simple geometric primitives such as boxes, spheres and generalized cylinders (see Fig. 3.18) directly attached to the hierarchy. This provides a volume and a surface to the body parts. Besides the rendering aspect, this additional information is also useful to select points on the surface with a pointer. For a human figure, the back and torso are vertically divided in several small slices, to show the bending of the spine.

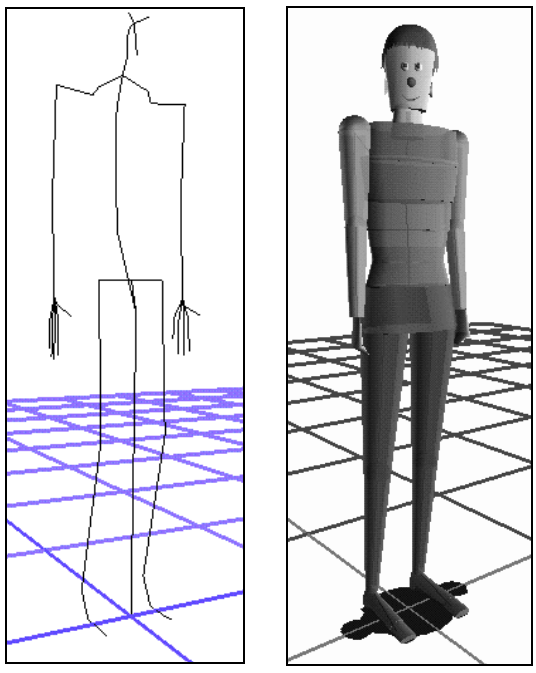


Figure 3.18 Two representations of a human figure: stick figure (left) and with simple geometric primitives (right).

More sophisticated rendering techniques allow to generate photorealistic representations of the human body appearance in real-time. However, this is not indispensable for our purposes.

3.9.2 Mass properties

Mass properties of a rigid body include its mass, as well as its center of mass and possibly its inertia tensor, which is mainly used for dynamic simulations. Here, the mass properties are used for static balance control. The mass properties of each body segment can be obtained from measurements, or be computed from the volume and density of the geometric primitives that approximate the body shape.

Once the mass properties of each single rigid body are known, the mass properties of the whole articulated structure (such as its mass m_{tot} and center of mass $\boldsymbol{G}_{tot}$) and those of the upper body (m_j^U and $\boldsymbol{G}_j^U$) and lower body (m_j^L and $\boldsymbol{G}_j^L$) of each joint can be efficiently computed with a recursive traversal of the tree. Of course, the position of the centers of mass depends on the configuration $\boldsymbol{q}$ of the articulated structure.

3.9.3 Sites

Special frames can be attached to the skeleton to locate particular points of interest on the body, such as the center of the hands or a vision frame located between the eyes. These frames may be used as attachment points, or as end-effectors for inverse kinematics. In robotics, this is known as a *tool frame* [CRA 89], since it represents the exact position and orientation of the tool being controlled on the robot, and does not necessarily coincides with a joint frame. Here, such a frame is called an *end-effector frame*, and can be anything from the center of the hands to the vision frame located between the eyes, whose orientation can be constrained to control the gazing direction of the figure.

3.10 Conclusion

In this chapter, we have reviewed the components of an articulated figure model, for manipulation and animation purposes. The key components are the joint models, since they constrain the postures to a set of feasible ones and describe the mobility of each joint.

While joints with one degree of freedom present no particular difficulties, the more complex models, such as the ball-and-socket joint, are more arduous to define, because of the complex nature of rotations. As already noted by Grassia [GRA 98], there is no ideal parametrization of 3D rotations. However, we have seen that for the purpose of joint limits definition, the swing-and-twist decomposition provides an intuitive and practical basis. For the purpose of motion parametrization, as is required for an inverse kinematics algorithm to manipulate the joint configuration, it is important to select singularity-free parametrizations (such as quaternions), or at least a parametrization with avoidable singularities (such as the axis-angle parametrization proposed by Grassia).

Once a body model has been defined and parametrized, it is ready to be manipulated by an animation engine. Of course, the engine has to take care of the joint limits and joint couplings defined in the model.

Chapter 4

Numerical resolution of the inverse kinematics problem

4.1 Introduction

This chapter presents the numerical resolution method based on the popular resolved motion rate technique that stem from robotics [WHI 69]. The choice of a numerical method for the problem of manipulating arbitrary articulated structures is motivated by their generality. The issues related to this resolution are discussed in detail, except the problems of joint limits and of solving conflicts between multiple tasks, that will be addressed in the next chapter.

First, the inverse kinematics problem is stated.

4.2 Statement of the inverse kinematics problem

Given an articulated structure whose generalized coordinates are the n components of the state vector $\boldsymbol{q}$, the inverse kinematics problem is to determine a solution to the following non-linear equation:

$$\boldsymbol{x}(\boldsymbol{q}) = \boldsymbol{g} \tag{4.1}$$

where $\boldsymbol{x}(\boldsymbol{q})$ and $\boldsymbol{g}$ are m-dimensional vectors expressed in the so-called *task* space (m is called the *valency* of the task). This equation can integrate as many independent scalar or vectorial equations, simply by “piling” them.

In practice more high level concepts are required to express meaningful task specifications in order to represent what the user actually wants to control. Typically, the task function $\boldsymbol{x}(\boldsymbol{q})$ represents the position or orientation of an end-effector frame while $\boldsymbol{g}$ is the goal to be reached.

The task is expressed in a given frame of reference, here called the control frame *C*. For Cartesian tasks, the control frame may coincide with the world reference frame (noted *W*). However, if the end-effector position is only partially constrained by the task, the use of a general control frame permits the specification of general point-on-line or point-on-plane constraints (see Fig. 4.1).

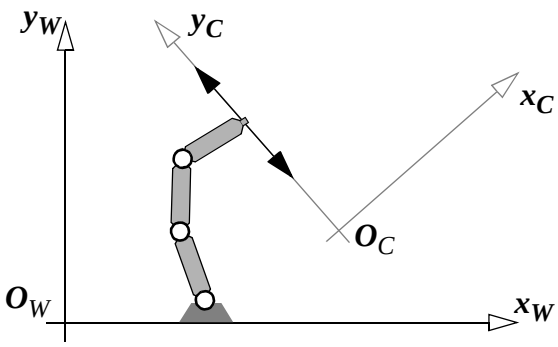


Figure 4.1 Partial position control of the chain tip, which is free to move along the y axis of the control frame *C*.

Since an exact solution to Eq. (4.1) does not necessarily exists, we also specify that the "best" approximate solution is required in that case. Hence the problem may be reformulated as an optimization problem that minimizes the residual error

$$e(\boldsymbol{q}) \;=\; \|\boldsymbol{x}(\boldsymbol{q}) - \boldsymbol{g}\| \tag{4.2}$$

The Euclidean norm is used here, and is supposed to be meaningful for $\boldsymbol{x}(\boldsymbol{q})$.

4.3 Overview of the numerical resolution method

Since Eq. (4.1) is non-linear in general, it cannot be solved by simple inversion of the function $\boldsymbol{x}(\boldsymbol{q})$. We use the resolved motion rate method to solve the problem (see Section 2.5.2), which is quite similar to the Newton-Raphson method [ORT 70]. As most numerical methods, it is an iterative procedure based on a linearization of the constraint equation about an initial point $\boldsymbol{q}^0$, that results in a Jacobian matrix. By inversion of the Jacobian matrix, the resulting set of linear equations can be solved for an increment $\Delta\boldsymbol{q}$. This phase must deal with the singularities of the Jacobian matrix. Then, a step in this direction is taken to a new configuration $\boldsymbol{q}^i$ that approaches a solution of the task equation. By iteratively repeating the process, the system converges towards a solution that either satisfies the task equation, or that locally minimizes the residual error when there is no exact solution.

4.4 Linearization of the task equation

A first-order (linear) approximation of the task function is given by the Taylor series expansion about the current configuration $\boldsymbol{q}^i$:

$$\boldsymbol{x}(\boldsymbol{q}^i + \Delta\boldsymbol{q}) \;=\; \boldsymbol{x}(\boldsymbol{q}^i) + J(\boldsymbol{q}^i)\Delta\boldsymbol{q} + \ldots \tag{4.3}$$

where $J(\boldsymbol{q})$ is the $m \times n$ Jacobian matrix of the function $\boldsymbol{x}(\boldsymbol{q})$.

Keeping only the linear terms of Eq. (4.3) results in:

$$\Delta\boldsymbol{x} \cong J(\boldsymbol{q}^i)\Delta\boldsymbol{q} \tag{4.4}$$

where $\Delta \boldsymbol{x} = \boldsymbol{g} - \boldsymbol{x}(\boldsymbol{q}^i)$ is a known desired task increment, and where $\Delta \boldsymbol{q}$ is the unknown increment of joint coordinates.

Once a direction $\Delta \boldsymbol{q}$ has been computed by solving Eq. (4.4), the next configuration can be computed by:

$$\boldsymbol{q}^{i+1} = \boldsymbol{q}^i + \Delta \boldsymbol{q}$$

Because of the non-linearity of the task function, the approximation of Eq. (4.4) only holds for "small" task increments. The error introduced when taking a step $\Delta \boldsymbol{q}$ is the difference between the desired increment $\Delta \boldsymbol{x}$ and the actual increment $\boldsymbol{x}(\boldsymbol{q}^{i+1}) - \boldsymbol{x}(\boldsymbol{q}^i)$: if this error is too large, the path of the end-effector over several iterations may be erratic, rather than straight. To select a sufficiently small task increment, $\Delta \boldsymbol{x}$ can be clamped to an empirically fixed threshold. A more sophisticated method is known as *line search* [PRE 92]: the direction vector $\Delta \boldsymbol{q}$ is scaled by a factor chosen to ensure that the error is minimized as much as possible. Choosing this parameter usually requires an iterative search along the direction vector. A similar strategy, described by Sims *et al.* [SIM 88] and Watt and Watt [WAT 92], is to use adaptive steps based on several trials to find the largest step whose tracking error is below a given threshold. Using adaptive steps reduces the number of iterations and hence of expensive matrix inversions but, on the other hand, the evaluation of the error may also be costly. For this reason, we have selected to use fixed-length steps in our system.

4.5 Computing the Jacobian matrix

The Jacobian matrix, which is the multi-dimensional form of the derivative, maps differential changes of the joint coordinates $d\boldsymbol{q}$ to differential changes of the task coordinates $d\boldsymbol{x}$ expressed in the control frame, and is a function of the configuration $\boldsymbol{q}$:

$$d\boldsymbol{x} = J(\boldsymbol{q}) d\boldsymbol{q}$$

Hence, the (i, j) component of this matrix is

$$J(\boldsymbol{q})_{[i,j]} = \frac{\partial \boldsymbol{x}_i(\boldsymbol{q})}{\partial \boldsymbol{q}_j}$$

The Jacobian matrix can be computed either by analytical differentiation of $\boldsymbol{x}(\boldsymbol{q})$ or by numerical differentiation, which is less accurate and thus should be chosen only if an analytical solution is lacking. Numerous authors have obtained Jacobian matrices for typical tasks, with methods of variable efficiency [PAU 81] [CRA 89] [WAM 86] [WAT 92]. In this section, we summarize the analytical results for revolute and ball-and-socket joints, for the following tasks:

- absolute position and orientation control of an end-effector frame
- loop constraint between two end-effectors
- absolute position control of the center of mass of the structure

When the number of joint types and tasks is high, the analytic determination of Jacobian matrices becomes a tedious task, since a Jacobian is required for each combination of joint type and task type. We start by the definition of the tasks and of their Jacobian matrices, then we look at the details of the main joint models. The Jacobians are finally summarized in Table 4.1.

4.5.1 Task for the position control of an end-effector frame

This task controls the position $\boldsymbol{x}_T$ of the origin $\boldsymbol{O_E}$ of an end-effector frame E (see Fig. 4.2). Thus, the translation Jacobian $J_T(\boldsymbol{q})$ is defined by

$$d\boldsymbol{x}_T = J_T(\boldsymbol{q})d\boldsymbol{q} \tag{4.5}$$

If the end-effector frame E belongs to the lower body of the j^{th} joint, the end-effector position is independent on the joint parameters, thus the partial derivatives (and hence the corresponding Jacobian columns) are zero.

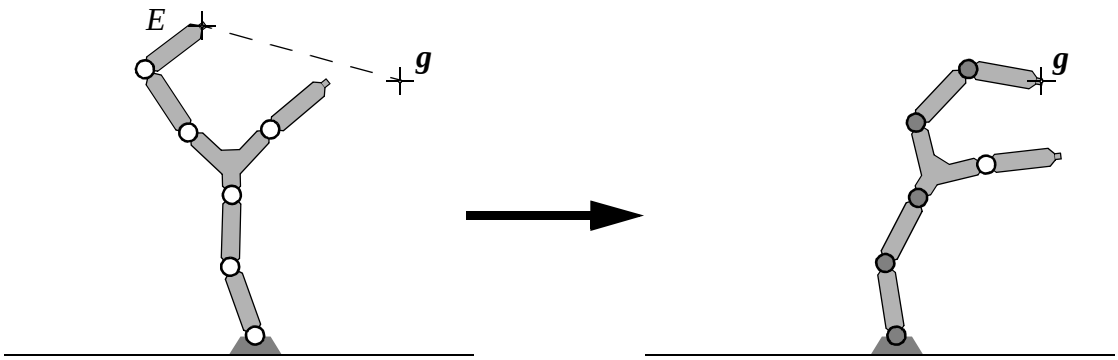


Figure 4.2 Absolute position control of the origin of an end-effector frame. The five grey-shaded joints are adjusted to satisfy the task.

4.5.2 Task for the orientation control of an end-effector frame

The orientation of an end-effector frame can also be controlled. The choice of task coordinates is less obvious than for position control, since many parametrizations exist for general 3D rotations. Actually, a simple solution is to solve the problem directly at the differential level, without relying on a particular parametrization of orientation [LEB 85]. For this purpose, the infinitesimal variation of rotation is represented by a vector noted $d\boldsymbol{x}_R$, which is analogous to the classical angular velocity vector $\boldsymbol{\omega}$. More precisely, they are related by $d\boldsymbol{x}_R = \boldsymbol{\omega}dt$, where t is the time variable.

The rotation Jacobian $J_R(\boldsymbol{q})$ is thus defined by

$$d\boldsymbol{x}_R = J_R(\boldsymbol{q})d\boldsymbol{q} \tag{4.6}$$

Additional special tasks can be defined in order to constrain an orientation with only one or two DOFs: a typical example is the “look at” mode: a vector, fixed in the end-effector frame, is

constrained to “aim” at a given target point. This requires two degrees of freedom, since the end-effector frame is still free to twist about the fixed vector.

4.5.3 Task for loops

Ensuring loops in a structure is useful because they cannot be defined in the tree-structured body model. Temporarily joining up body parts (such as the hands) to edit a posture can also be achieved with this task.

Given two end-effector frames (E_1 and E_2) in the hierarchy, the goal is that the position of their origin must coincide:

$$\boldsymbol{x}_{T_1/T_2}(\boldsymbol{q}) = \boldsymbol{x}_{T_1}(\boldsymbol{q}) - \boldsymbol{x}_{T_2}(\boldsymbol{q}) = \mathbf{0}$$

where $\boldsymbol{x}_{T_1}$ and $\boldsymbol{x}_{T_2}$ are the absolute positions of the two end-effectors, respectively. Their respective Jacobian matrices are noted J_{T_1} and J_{T_2}. On this basis, the Jacobian matrix for the loop task is equal to $J_{T_1} - J_{T_2}$, except for the joints that affect the position of both end-effectors. Indeed, these joints have no impact on the task (see Fig. 4.3) and, as such, their associated columns must be set to zero.

An analogous task can be formulated between the orientation of the two end-effector frames: applying such a constraint ensures that they have the same orientation.

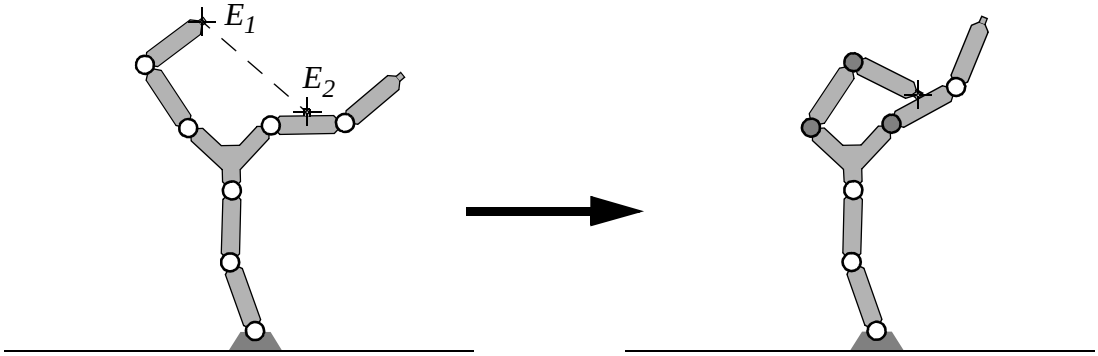


Figure 4.3 Joining two end-effectors (E_1 and E_2) together: only the three grey-shaded joints affect the satisfaction of the loop constraint.

4.5.4 Task for the position control of the center of mass

Provided that the mass properties are known for each segment of the articulated figure (see Section 3.9.2), the position $\boldsymbol{x}_G$ of its center of mass $\boldsymbol{G}_{tot}$ can be constrained like that of any other end-effector. For this purpose, Boulic *et al.* [BOU 94] [MAS 96] have introduced the kinetic Jacobian matrix, noted J_G, that maps differential changes of the generalized coordinates to changes of the position of the center of mass:

$$d\boldsymbol{x}_{\boldsymbol{G}} = J_G(\boldsymbol{q})d\boldsymbol{q} \tag{4.7}$$

4.5.5 Joint-specific derivatives

We do not give the full derivations of the Jacobian matrices, since this is done in the literature. Instead, we give the main results required to obtain the Jacobians, with the following derivatives for revolute and ball-and-socket joint types.

We compute the variation of a given generic vector $\boldsymbol{p}$ (expressed in the control frame of the task) attached to the frame of the jth joint, due to the variation of the parameters of that joint. This allows to determine the Jacobians J_T and J_G.

For a *revolute* joint, with angle θ_j and unit axis of rotation $\boldsymbol{a}_j$ expressed in the control frame, the variation is simply $\boldsymbol{a}_j \times \boldsymbol{p}$.

For a *ball-and-socket* joint, parametrized by an axis-angle $\boldsymbol{k}_j \in \Re^3$, the results are more complex. To simplify, the Euler parameters (i.e. unit quaternion) are introduced as an intermediate parametrization, and are noted $\boldsymbol{e}_v$ and e_s (see Appendix B). For this purpose, we introduce $J_{\boldsymbol{k}}^{\boldsymbol{e}}$, which is the 4×3 Jacobian matrix that maps differential changes of axis-angle parameters to changes of Euler parameters (see Appendix F for its expression). On this basis, we can define $J_{\boldsymbol{e}}^{\boldsymbol{p}}(\boldsymbol{p})$, which is the 3×4 Jacobian that maps differential changes of Euler parameters to changes of $\boldsymbol{p}$:

$$d\boldsymbol{p} = J_{\boldsymbol{e}}^{\boldsymbol{p}}(\boldsymbol{p}) \begin{bmatrix} d\boldsymbol{e}_v \\ de_s \end{bmatrix}$$

The Jacobian $J_{\boldsymbol{e}}^{\boldsymbol{p}}(\boldsymbol{p})$ is obtained by differentiation of the function that rotates a vector $\boldsymbol{p}$ by a unit quaternion (Eq. (B.2) in Appendix B). The result is:

$$J_{\boldsymbol{e}}^{\boldsymbol{p}}(\boldsymbol{p}) = [A | \boldsymbol{b}]$$

where

$$A = 2(\boldsymbol{e}_v^T \boldsymbol{p}) I_3 - [\boldsymbol{b} \times]$$
$$\boldsymbol{b} = 2(\boldsymbol{e}_v \times \boldsymbol{p} + e_s \boldsymbol{p})$$

By the chain rule [LUE 96], the Jacobians $J_{\boldsymbol{e}}^{\boldsymbol{p}}(\boldsymbol{p})$ and $J_{\boldsymbol{k}}^{\boldsymbol{e}}$ can then be combined to form the final 3×3 matrix $J_{\boldsymbol{k}}^{\boldsymbol{p}} = J_{\boldsymbol{e}}^{\boldsymbol{p}}(\boldsymbol{p}) J_{\boldsymbol{k}}^{\boldsymbol{e}}$, that maps axis-angle changes to position changes of $\boldsymbol{p}$.

See also [GRA 98] for information about axis-angle derivatives for orientation control.

Jacobian type	Revolute joint (1 DOF) with unit axis of rotation $\boldsymbol{a}_j$	Ball-and-socket joint (3 DOF) parametrized by an axis-angle $\boldsymbol{k}_j$
Translation Jacobian J_T	$\boldsymbol{a}_j \times \overrightarrow{\boldsymbol{O}_J\boldsymbol{E}}$	$J_{\boldsymbol{e}}^{\boldsymbol{p}}(\overrightarrow{\boldsymbol{O}_J\boldsymbol{E}})\ J_{\boldsymbol{k}_j}^{\boldsymbol{e}}$
Rotation Jacobian J_R	$\boldsymbol{a}_j$	$2(\gamma I_3 + [-\boldsymbol{k}_j \times] - \eta \boldsymbol{k}_j \boldsymbol{k}_j^T)^{-1}$ where $\Theta = \lVert \boldsymbol{k}_j \rVert$, $\gamma = \Theta \cot(\Theta/2)$ and $\eta = (\gamma - 2)/\Theta^2$. When Θ is small, the limit for $\Theta \to 0$ must be used instead ($\gamma = 2$, $\eta = 0$).
Kinetic Jacobian J_G	$\frac{m_j^U}{m_{tot}}\left(\boldsymbol{a}_j \times \overrightarrow{\boldsymbol{O}_j\boldsymbol{G}_j^U}\right)$	$\frac{m_j^U}{m_{tot}} J_{\boldsymbol{e}}^{\boldsymbol{p}}\left(\overrightarrow{\boldsymbol{O}_j\boldsymbol{G}_j^U}\right) J_{\boldsymbol{k}_j}^{\boldsymbol{e}}$

Table 4.1 Summary of the task Jacobian matrices for revolute and ball-and-socket joints.

4.5.6 Rank of the Jacobian matrix and singularities

An important characteristic of a Jacobian matrix is its rank *r*, that varies in the joint space $\boldsymbol{q}$. The rank is the largest number of linearly independent rows or columns of the matrix. It informs about the mobility of the end-effector in the task space. For example, in a full-rank configuration (*r=min (m, n)*), the end-effector can move in any direction. The configurations where the rank of the matrix abruptly drops are characterized by a loss of end-effector mobility. These kinematic singularities of the task function can be roughly classified into two categories:

- parametric singularities
- workspace boundary singularities

A *parametric* singularity is solely due to the choice of parametrization: as shown in Section 3.5, parametrizing a joint with Euler angles results in configurations where the end-effector loses its mobility (this is known as gimbal lock). If this does not correspond to a mechanical realization, such singularities can be removed by adopting a more appropriate parametrization.

A *workspace boundary* singularity occurs when the end-effector reaches the limits of its workspace: the Jacobian becomes singular as the end-effector cannot move anymore in the direction normal to the boundary. Obviously, this singularity cannot be removed since it is inherent to the problem.

As shown in the next section, the presence of singularities considerably complicates the Jacobian inversion process, especially in their vicinity, but they have to be dealt with since they are

hardly avoidable. For this purpose, the Singular Value Decomposition is a very useful tool.

4.6 The Singular Value Decomposition

The Singular Value Decomposition (SVD) is a powerful linear algebra tool [PRE 92] [BJO 96] [GOL 96] of particular interest for the analysis of redundant manipulators [KLE 83] [MAC 90], since it explicitly provides orthonormal bases for the fundamental subspaces of a matrix. The usefulness of the SVD also lies in its ability to detect the ill-conditioning of a matrix [PRE 92] [GOL 96].

4.6.1 The fundamental subspaces

To any $m \times n$ matrix J we can associate two linear subspaces, namely its range $R(J)$ and its null space $N(J)$ defined by [GOL 96]:

$$R(J) := \{ \boldsymbol{v} \in \Re^m \,|\, \exists \boldsymbol{w} \in \Re^n, J\boldsymbol{w} = \boldsymbol{v} \}$$
$$N(J) := \{ \boldsymbol{v} \in \Re^n \,|\, J\boldsymbol{v} = \boldsymbol{0} \}$$

These concepts are important for rectangular or singular matrices. The *range* is the subspace that can be "reached" by application of J, and its dimension is called the *rank* of J. The *null space* is the subspace that maps to the null vector, and its dimension is called the *nullity* of J. The orthogonal complements of these subspaces are noted $R(J)^\perp$ and $N(J)^\perp$ respectively. Moreover, the following fundamental relationships hold [GOL 96]:

$$\begin{aligned} N(J)^\perp &= R(J^T) \\ R(J)^\perp &= N(J^T) \end{aligned} \tag{4.8}$$

4.6.2 The definition of the SVD

The SVD of an $m \times n$ matrix J with rank r is:

$$J = U \Sigma V^T \tag{4.9}$$

where $U = [\boldsymbol{u}_1 \ldots \boldsymbol{u}_m]$ is an $m \times m$ orthogonal matrix, $V = [\boldsymbol{v}_1 \ldots \boldsymbol{v}_n]$ is an $n \times n$ orthogonal matrix, and Σ is an $m \times n$ matrix having the following form:

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \text{ where } D = \begin{bmatrix} \sigma_1 & & 0 \\ & \ldots & \\ 0 & & \sigma_r \end{bmatrix} \text{ is an } r \times r \text{ diagonal matrix,}$$

and where σ_i is the i^{th} singular value, which is always positive.

The columns of matrix U and of matrix V span the four fundamental subspaces associated with matrix J:

$$\begin{aligned} B_{R(J)} &= [\boldsymbol{u}_1 \dots \boldsymbol{u}_r] \\ B_{R(J)^\perp} &= [\boldsymbol{u}_{r+1} \dots \boldsymbol{u}_m] \\ B_{N(J)^\perp} &= [\boldsymbol{v}_1 \dots \boldsymbol{v}_r] \\ B_{N(J)} &= [\boldsymbol{v}_{r+1} \dots \boldsymbol{v}_n] \end{aligned} \tag{4.10}$$

The orthogonal projection matrix on any subspace S is easily computed as $P_S = B_S B_S^T$

The matrix J can also be written as

$$J = B_{R(J)} D B_{N(J)^\perp}^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{4.11}$$

4.7 Solving the linear system

Once the Jacobian matrix J and a task increment $\Delta \boldsymbol{x}$ have been computed, Eq. (4.4) can be solved in order to obtain an increment $\Delta \boldsymbol{q}$. This would be simple if the Jacobian was a square, non-singular matrix: in that rare case, the unique solution would be simply $\Delta \boldsymbol{q} = J^{-1} \Delta \boldsymbol{x}$. The number of constraints m is usually smaller than the number of degrees of freedom n. In that under-constrained case $m < n$, the classical matrix inverse is not applicable, and the solution is no longer unique. In the opposite case ($m > n$), the linear system may be over-constrained: in that case, no exact solution exist, and the *residual vector* $J\Delta \boldsymbol{q} - \Delta \boldsymbol{x}$ cannot be zero. In both cases, additional criteria must be defined in order to select an optimal solution. In the under-constrained case, the best solution must be selected among the valid ones, while in the over-constrained case, the best approximate solution must be selected.

4.7.1 Decomposition of the solution space

The solution space $\Re^n$ can be partitioned into two orthogonal subspaces: $N(J)$, that provides the set of solutions that do not contribute to the satisfaction of the problem of "reaching" $\Delta \boldsymbol{x}$, and its orthogonal complement $N(J)^\perp$ that of course contributes. Hence a general solution is composed of two terms: a particular solution to satisfy Eq. (4.4) as well as possible, and a homogeneous solution that can be used to satisfy other criteria.

4.7.2 A least-squares solution

The particular solution is usually based on the least-squares inverse $J^\dagger$ of J:

$$\Delta \boldsymbol{q}_{least-squares} = J^\dagger \Delta \boldsymbol{x} \tag{4.12}$$

The least-squares inverse, sometimes called Moore-Penrose inverse or pseudoinverse, is a generalized inverse [BOUL 71] [BEN 74] [NAS 76] that satisfies least-squares criteria in both under- and over-determined situations. More precisely, the least-squares solution $J^\dagger \Delta \boldsymbol{x}$ is the vector of minimum norm among those that minimize the residual error $\|J\Delta \boldsymbol{q} - \Delta \boldsymbol{x}\|$. In other words, the joint increment tries to match as well as possible the desired increment $\Delta \boldsymbol{x}$ for the

task, while keeping low joint increments. Note however that the minimization of $\|J\Delta\boldsymbol{q} - \Delta\boldsymbol{x}\|$ has priority over the minimization of $\|\Delta\boldsymbol{q}\|$.

The least-squares inverse $J^{\dagger}$ of an $m \times n$ matrix J is also defined as the unique matrix that satisfies the following four properties:

$$\begin{aligned} JJ^{\dagger}J &= J \\ J^{\dagger}JJ^{\dagger} &= J^{\dagger} \\ (JJ^{\dagger})^{T} &= JJ^{\dagger} \\ (J^{\dagger}J)^{T} &= J^{\dagger}J \end{aligned} \tag{4.13}$$

Additional properties and techniques to compute this inverse can be found in reference books [BOUL 71] [BEN 74] [LUE 96]. An important number of techniques has been developed to deal with least-squares problems [BJO 96] because of their importance in so many application areas [NAS 76] [KOH 89].

4.7.3 The homogeneous solution

The particular solution of Eq. (4.12) is confined to the subspace $N(J)^{\perp}$, because this is where the useful components for achieving the task increment lie. In the case of a redundant problem, its orthogonal complement $N(J)$ contains additional components that, by definition, do not affect the satisfaction of the task and that can be used for other purposes. This is why the null space is sometimes called the *redundant space* in robotics [HAN 81]. Hence, adding to the solution a component belonging to $N(J)$ allows to select one of the many solutions that can achieve the task: this additional component is called the *homogeneous solution* and can be exploited for a variety of purposes.

$$\Delta\boldsymbol{q}_{homogeneous} = P_{N(J)}\boldsymbol{z}$$

where

$$P_{N(J)} = I_n - J^{\dagger}J$$

is the $n \times n$ orthogonal projection operator (or *projector*) on $N(J)$ and $\boldsymbol{z} \in \Re^n$ is an arbitrary vector.

4.7.4 The general solution

In summary, a general solution to Eq. (4.4) is composed of a particular solution and of a homogeneous solution (see Fig. 4.4):

$$\Delta\boldsymbol{q} = J^{\dagger}\Delta\boldsymbol{x} + P_{N(J)}\boldsymbol{z} \tag{4.14}$$

The vector $\boldsymbol{z}$ can be exploited to locally minimize a scalar criterion $h(\boldsymbol{q})$ by setting $\boldsymbol{z} = \xi\nabla h(\boldsymbol{q})$, where $\xi < 0$ is a gain factor. This is a well-known result in robotics [LIE 77]. A typical application is to keep the joint angles as close as possible to some desired values, as shown in Fig. 4.5.

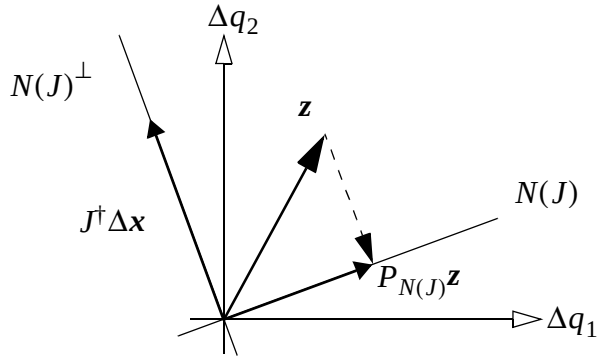


Figure 4.4 The two components of the general solution (Eq. (4.14)) for a one-dimensional task (m=1) shown in a two-dimensional joint variation space (n=2).

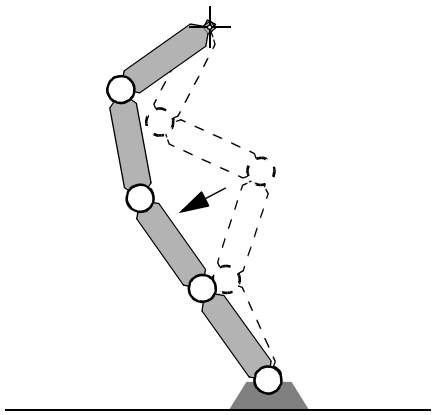


Figure 4.5 Minimizing a scalar criterion $h(\boldsymbol{q})$ without breaking the tip position constraint. The angles can be kept as close as possible to a desired posture $\boldsymbol{q}_{desired}$ by minimizing a criterion such as $h(\boldsymbol{q}) = \left\|\boldsymbol{q} - \boldsymbol{q}_{desired}\right\|^2$.

4.7.5 Drawback of the least-squares solution: the need for regularization

Despite its apparent attractiveness, the least-squares solution has one major drawback. In the proximity of a singularity, the problem becomes ill-conditioned: in an attempt to precisely minimize the residual error, which is the criterion having priority, the norm of the resulting least-squares solution may tend to infinity [MAC 90]. This is unacceptable in practice since it violates the small increments hypothesis of Eq. (4.4). So-called *regularization* techniques [NEU 98] have been developed to deal with this problem:

- the truncated singular value decomposition
- damped least-squares (also known as Tikhonov regularization [TIK 63])
- numerical filtering [MAC 88] (an extension of the damped least-squares technique)

We have used the damped least-squares technique, described below.

4.7.6 A regularization technique: damped least-squares

Instead of minimizing the tracking error alone, the following weighted combination of tracking error and solution norm is minimized:

$$\|J\Delta\boldsymbol{q} - \Delta\boldsymbol{x}\|^2 + \lambda^2\|\Delta\boldsymbol{q}\|^2 \tag{4.15}$$

where λ, known as the *damping factor*, weights the relative importance of tracking error *versus* the norm of the solution. When $\lambda = 0$, only the tracking error is minimized, and this is equivalent to the least-squares solution. As λ is increased, the solution norm is forced to decrease, at the expense of some tracking accuracy. Hence a trade-off must be found.

A generalization of the least-squares inverse, called the damped least-squares inverse, is used to compute a solution that minimizes Eq. (4.15). In robotics, this technique has been suggested by Wampler [WAM 86] and Nakamura *et al.* [NAK 86]. When $\lambda > 0$ the damped least-squares inverse of a matrix J is:

$$J^{\dagger\lambda} = J^T(JJ^T + \lambda^2 I_m)^{-1} \tag{4.16}$$

The two solutions are compared on Fig. 4.6, along a single dimension. Clearly, the damped solution is bounded while the least-squares solution is not.

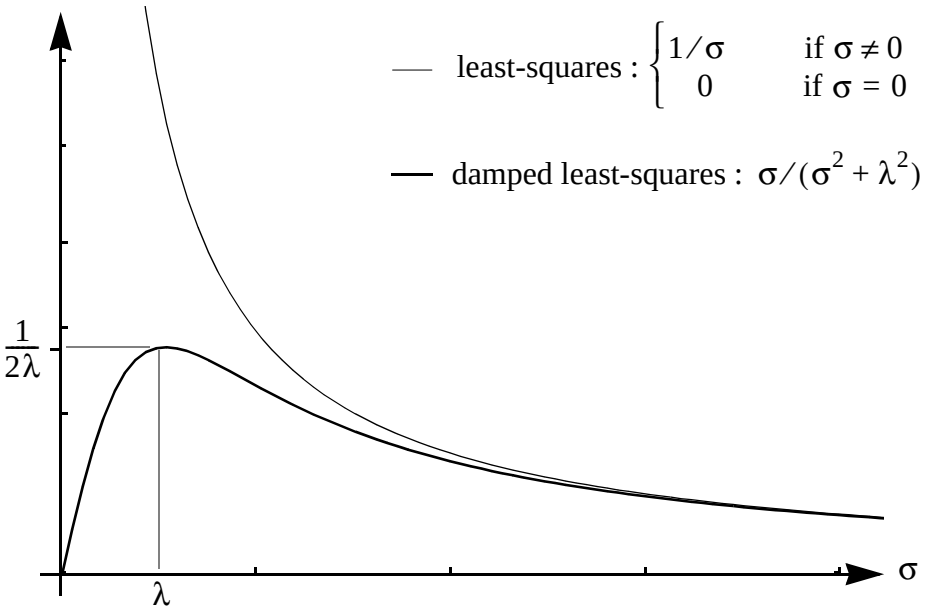


Figure 4.6 Comparison of the least-squares and damped least-squares functions of a scalar value σ. The least-squares function is discontinuous at the singularity $\sigma = 0$, while the damped least-squares function (with $\lambda > 0$) is continuous, and bounded by $1/(2\lambda)$.

When using the damped least-squares inverse, one must pay attention to the fact that some of the properties of the least-squares inverse do not hold anymore. For example, the first two properties of Eq. (4.13) do not hold when $\lambda > 0$:

$$JJ^{\dagger\lambda}J \neq J$$
$$J^{\dagger\lambda}JJ^{\dagger\lambda} \neq J^{\dagger\lambda}$$

Damping the inverse appearing in the projection matrix of Eq. (4.14) is tempting, however it would be incorrect since this may cause an undesirable task motion $J(I_n - J^{\dagger\lambda}J)\boldsymbol{z} \neq \mathbf{0}$ when $\lambda > 0$, thereby affecting the satisfaction of the task.

4.7.7 Computing the damped least-squares inverse with the SVD

Several useful quantities may be easily computed from the SVD of a matrix J with rank r, and in particular its damped least-squares inverse:

$$\left(J = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \right) \rightarrow \begin{cases} J^{\dagger} = \displaystyle\sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T \\ J^{\dagger\lambda} = \displaystyle\sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \boldsymbol{v}_i \boldsymbol{u}_i^T \\ J^{\dagger}J = P_{N(J)^{\perp}} = \displaystyle\sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{v}_i^T \end{cases} \tag{4.17}$$

The SVD has a reputation for being expensive to compute, and in robotics it is usually preferred for analyzing the problem rather than for actually performing computations. Maciejewski has shown that the cost can be reduced if an incremental evaluation of the SVD is used, from an iteration to another [MAC 89]. However, this method is complex to implement. On current machines the use of a SVD does not significantly affect the real-time interaction, but it is always preferable to improve the efficiency of an algorithm. The use of the thin SVD instead of a full SVD provides a simple speed-up.

The *thin* SVD is a much used, trimmed down version of the SVD [GOL 96] (p. 72). The only difference is that an orthogonal basis for $R(J)^{\perp}$ is not provided. Still, this is sufficient to compute Eq. (4.17), with the advantage of being faster than a full SVD. Source code for performing a thin SVD can be found in the *Numerical Recipes* book [PRE 92].

From a computational point of view, it is also interesting to note that the thin SVD of an $m \times n$ matrix J with $m < n$ is slower to compute than the thin SVD of its transpose J^T. A measurement of the speed, performed with the thin SVD code of [PRE 92], reveals a speed-up factor of $(n/m)^{3/4}$. This fact can be exploited to reduce the computational cost of $J^{\dagger\lambda}$ and $J^{\dagger}J$, since the SVD of J^T provides the necessary bases for these computations (thanks to Eq. (4.8)), yet at a lower cost.

4.7.8 Computing an appropriate damping factor

A difficult problem with the damped least-squares technique is the evaluation of the optimum

damping factor λ in all situations: it should be zero when far from singularities (to provide the best possible tracking), and high enough to attenuate the unwanted oscillations when close to a singularity. However, a too high value also results in poor tracking accuracy.

Nakamura *et al.* [NAK 86], Wampler [WAM 86] and Maciejewski *et al.* [MAC 88] discuss various ways of computing the damping factor. A common method is to set a bound b_{max} on the norm of the solution:

$$\left\| J^{\dagger\lambda} \Delta \boldsymbol{x} \right\| \leq b_{max}$$

With the help of SVD analysis (Eq. (4.17)), this can be rewritten as:

$$\left\| J^{\dagger\lambda} \Delta \boldsymbol{x} \right\| = \left\| \left(\sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \boldsymbol{v}_i \boldsymbol{u}_i^T \right) \Delta \boldsymbol{x} \right\| = \sqrt{\sum_{i=1}^{r} \left(\frac{\sigma_i}{\sigma_i^2 + \lambda^2} (\boldsymbol{u}_i^T \Delta \boldsymbol{x}) \right)^2} \leq b_{max}$$

Note that, as expected, the solution norm decreases monotonically as λ increases. Hence, when an SVD of J is available, the optimal λ for a given bound b_{max} can be found iteratively with Newton's method. When an SVD of J is not available (because it is considered too computationally expensive), the minimum singular value σ_{min} (or an estimate of it) is sufficient to compute an appropriate, albeit sub-optimal, damping factor. This approach, used in our system, is proposed by Maciejewski *et al.* [MAC 88] and summarized in Fig. 4.7.

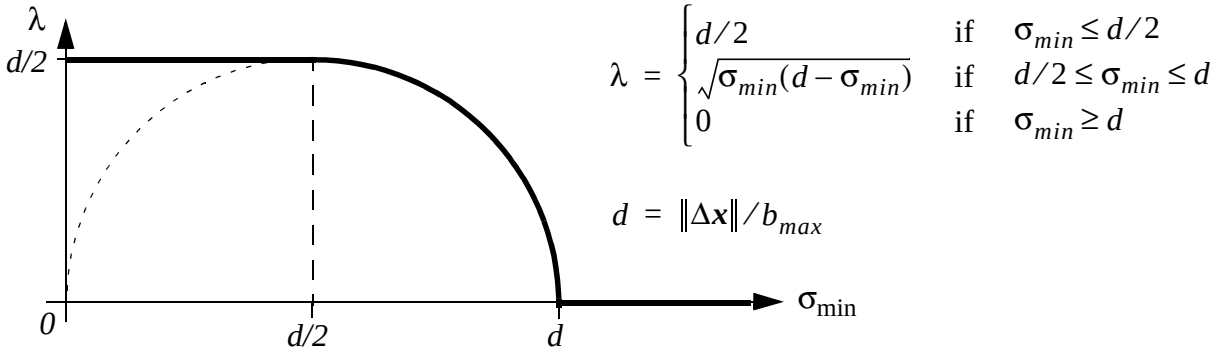


Figure 4.7 Choice of the damping factor λ, as a function of the minimum singular value.

4.8 Convergence of the iterative process to a solution

An important question that arises with iterative procedures is whether they converge towards a satisfying solution, if they converge at all.

The Newton-Raphson method is known to converge, under suitable conditions, to a solution of a set of non-linear equations, given a starting point which is sufficiently close to the solution [ORT 70]. The theoretical conditions for convergence to a solution are quite restrictive, but in practice the method behaves much better. A limitation of the original Newton method is that it

cannot deal with systems that do not have a solution. To overcome this limitation, Ben Israel [BEN 65] [BEN 66] has proposed to extend the method with a least-squares inverse in order to invert a possibly singular Jacobian matrix (as in the iterative method described above), and has studied its convergence. The iterative process is proven to converge to a stationary point $\boldsymbol{q}^*$ of $e(\boldsymbol{q})^2$, where $e(\boldsymbol{q})$ is the residual error defined by Eq. (4.2).

Thus, $\boldsymbol{q}^*$ is a least squares solution to Eq. (4.1): if no solution exists to the inverse kinematics problem, a solution that minimizes $e(\boldsymbol{q})$ is found. Interestingly, in that case the least-squares criterion used at the differential level is reflected on the choice of the final solution.

4.9 Termination

A termination condition must be provided in order to determine when the iterative process can be stopped, and the current configuration considered as the solution. A convergence tolerance ε must be defined, in order to check if the goal has been reached, up to a desired accuracy, i.e. when $e(\boldsymbol{q}) < \varepsilon$. This condition stops the process when a solution exists and when the convergence occurs. However, when no solution exists, this condition cannot detect the termination of the algorithm, because the distance to the goal is not necessarily small. Another possibility is to terminate the algorithm when the residual error $e(\boldsymbol{q})$ stops to decrease, but this is a reliable condition only if a line search algorithm [PRE 92] is used to determine the length of the step to be taken (see Section 4.4), since this guarantees that the error decreases between two successive iterations.

Within an interactive manipulation system, in which the tasks can change at any time during the iterative process, the need for a termination condition is less imperative because the algorithm keeps running continuously.

4.10 Conclusion

In this chapter we have recalled the main aspects of a classical technique for the numerical resolution of the inverse kinematics problem. It serves as a basis for two important extensions for the manipulation of articulated figures: first, an extension to deal with multiple tasks and to resolve their possible conflicts, and second, an extension to take into consideration the joint limits and the joint couplings of the body model. These two aspects are discussed in the next chapter.

Chapter 5

Simultaneous resolution of multiple tasks with possible conflicts

This chapter discusses two extensions to the numerical inverse kinematics technique described in the previous chapter. The first extension is the resolution of conflicts that arise when solving multiple tasks simultaneously. This is an important problem since manipulating a model with a single task would be impractical. Inevitably, conflicts arise between tasks that control the same joints, and a strategy must be selected in order to resolve such situations. Two resolution strategies can be established. The first strategy finds a compromise solution, according to weights assigned to each task in order to represent their relative importance, but then no one is exactly achieved. Another strategy is to arbitrate a conflict on the basis of a predefined priority order, hence creating a hierarchical relation between the tasks. This is believed to be useful for posture design, since tasks of different nature or of different functionality have usually to be performed with different priorities.

The second extension to the basic inverse kinematics method is the integration of joint limits and joint couplings in the resolution process. This aspect is of course important for the realism of the postures.

5.1 The occurrence of task conflicts and their resolution

Let us define a set of p tasks, each one with its goal $\boldsymbol{g}_i$:

$$\boldsymbol{x}_i(\boldsymbol{q}) = \boldsymbol{g}_i \text{ with } i = 1 \ldots p \tag{5.1}$$

In the case of multiple tasks, the optimal solution $\boldsymbol{q}^*$ should ideally satisfy all of them, i.e. $\boldsymbol{x}_i(\boldsymbol{q}^*) = \boldsymbol{g}_i, \forall i$. However, this may be impossible because one of the tasks is not achievable: for example a goal is not reachable. This may also happen if some tasks are not achievable simultaneously, whereas they can separately: in this case, these tasks are said to be *in conflict*. These conflicts may arise when one or more joints are shared by several tasks.

Several strategies can be devised to resolve a conflict. A first possibility is to find a trade-off

solution, where no task is achieved exactly, but each residual error is minimized. Since a weight can be assigned to each task to control the distribution of the residual error, this is known as a weighting strategy. A second possibility is to sort the tasks by order of priority, in order to satisfy the most important tasks in the first place. These two strategies are now discussed in more detail.

5.2 The weighting strategy

The resolution method presented in the previous chapter implicitly deals with multiple tasks, since Eq. (4.1) is vectorial, and thus can be interpreted as a set of p tasks with

$$\boldsymbol{x}(\boldsymbol{q}) = \begin{pmatrix} \boldsymbol{x}_1(\boldsymbol{q}) \\ \dots \\ \boldsymbol{x}_p(\boldsymbol{q}) \end{pmatrix} \text{ and } \boldsymbol{g} = \begin{pmatrix} \boldsymbol{g}_1 \\ \dots \\ \boldsymbol{g}_p \end{pmatrix}$$

and with the corresponding linearization terms:

$$J = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_p \end{bmatrix} \text{ and } \Delta\boldsymbol{x} = \begin{pmatrix} \Delta\boldsymbol{x}_1 \\ \dots \\ \Delta\boldsymbol{x}_p \end{pmatrix}$$

where $J_i := d\boldsymbol{x}_i(\boldsymbol{q}) / d\boldsymbol{q}$ and $\Delta\boldsymbol{x}_i = \boldsymbol{g}_i - \boldsymbol{x}_i(\boldsymbol{q})$ are the Jacobian and desired increment of the i$^{\text{th}}$ task, respectively.

According to Section 4.8, a least-squares solution is found. However, this point requires attention: as stated in Section 4.4, the increment $\Delta\boldsymbol{x}$ must be clamped to some threshold. When solving multiple tasks, the individual increments $\Delta\boldsymbol{x}_i$ cannot be clamped independently: their relative magnitude must be preserved in order to converge towards the desired least-squares solution. Hence, a unique global resizing factor is applied to $\Delta\boldsymbol{x}$, which is computed on the basis of the magnitudes $\|\Delta\boldsymbol{x}_i\|$ of its elements.

By definition, the least-squares solution $\boldsymbol{q}^*$ minimizes the total residual error function:

$$e^{tot}(\boldsymbol{q}) = \|\boldsymbol{x}(\boldsymbol{q}) - \boldsymbol{g}\| = \sqrt{\sum_{i=1}^{p} e_i(\boldsymbol{q})^2} \tag{5.2}$$

where $e_i(\boldsymbol{q}) = \|\boldsymbol{x}_i(\boldsymbol{q}) - \boldsymbol{g}_i\|$ is the residual error of the i$^{\text{th}}$ task.

A least-squares solution provides a best approximate solution when the system is over-determined (see Fig. 5.1 and Fig. 5.2). This is a compromise solution that satisfies none of the tasks. From the user point of view, the tasks act like rubber band that tend to attract each end-effector to its goal, by some “force”. Adjusting the goals of the tasks allows to manipulate the posture in an artistic fashion, but then no task can be exactly satisfied.

5.2.1 Weighting the residual errors

The control over the least-squares solution can be improved by assigning a weight to each task. This weight represents the relative importance of each task with respect to the others. For this purpose the total error to be minimized can be redefined as follows:

$$e^{tot}(\boldsymbol{q}) = \sqrt{\sum_{i=1}^{p} w_i e_i(\boldsymbol{q})^2} \tag{5.3}$$

where $w_i > 0$ is the scalar weight associated to the ith task. To solve this weighted problem, the following transformations must be performed

$$\begin{aligned} J_i' &= \sqrt{w_i} J_i \\ \Delta \boldsymbol{x}_i' &= \sqrt{w_i} \Delta \boldsymbol{x}_i \end{aligned} \qquad \text{for } i = 1 \ldots p \tag{5.4}$$

and the problem is solved as before, but with the set of J_i' and $\Delta \boldsymbol{x}_i'$. This transformation is equivalent to using a *weighted least-squares inverse* [BEN 74] instead of a simple least-squares inverse, when solving the linear system of equations.

The weighting procedure has another important application. When tasks with different dimensional units are combined, Eq. (5.2) is meaningless, and the result depends on the choice of the units. This typically occurs when a position task (expressed in millimeters) and an orientation task (expressed in radians) are solved simultaneously. Doty *et al.* [DOT 93] discuss the fallacy of using the least-squares inverse in such situations. Weighting the relative importance of the tasks allows to compensate for the possible different orders of magnitude or different units. Ignoring this results in meaningless solutions, and also in a very poor convergence speed if the magnitudes are different. However, how to choose the weighting coefficients in these situations is not always clear: the only guideline is that all the weighted tasks should be expressed in units of approximately the same magnitude.

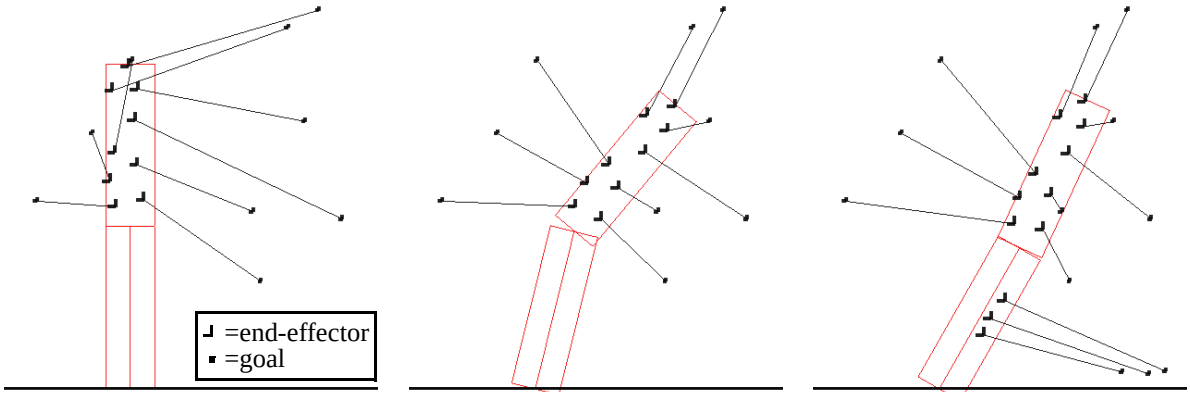


Figure 5.1 Least squares solution to an over-constrained problem. On the left, a 2 DOF chain is in its initial configuration and numerous tasks with equal weight are affected to it. In the middle, the least-squares solution is shown. On the right, three tasks are added to the lower part of the chain, and the resulting new least-squares configuration is shown.

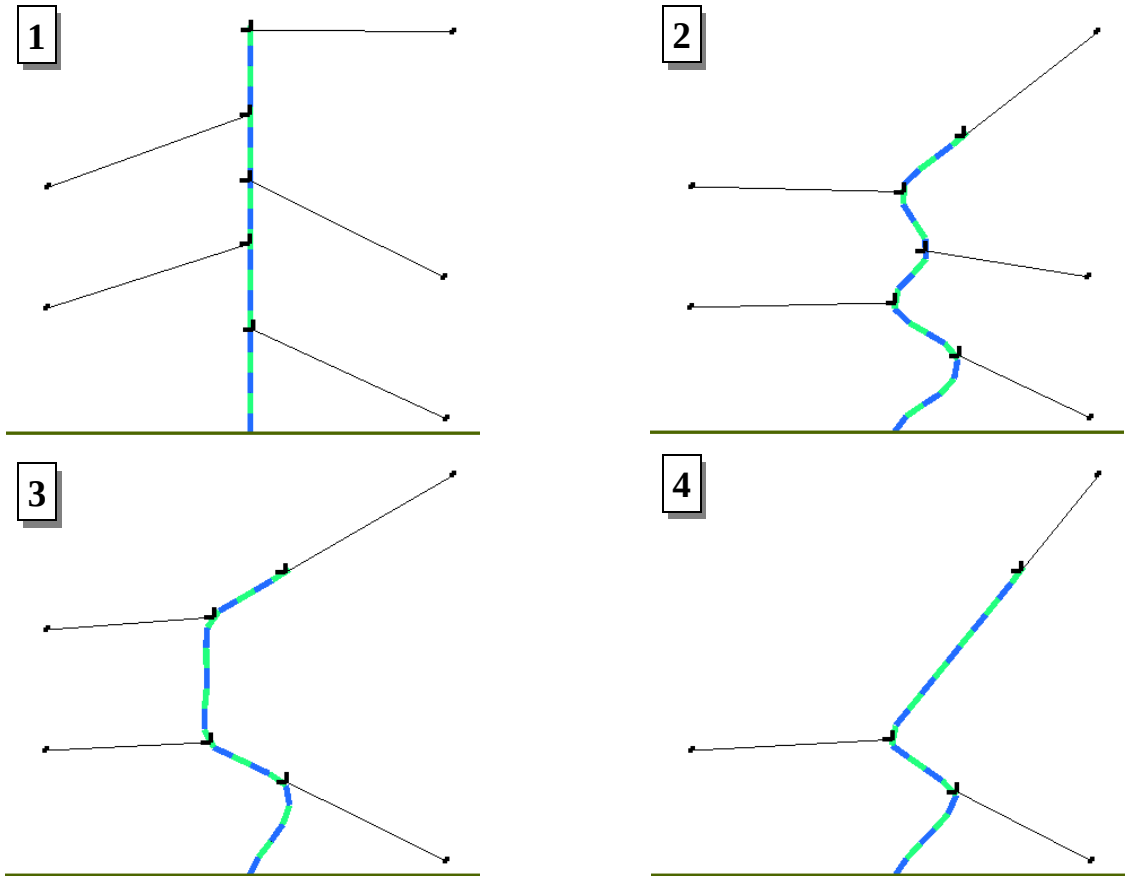


Figure 5.2 In this example, five tasks are affected to a simple 20 DOF chain. No solution exists that satisfies all the tasks. A least-squares solution is shown in the second figure. The last two figures show the configuration that results from the removal of one and two tasks, respectively.

5.2.2 Limitations of the weighting strategy

Ideally, one would like the weights to be related to the residual errors by $w_j e_j(\boldsymbol{q}^*) = w_k e_k(\boldsymbol{q}^*)$, for any pair of conflicting tasks *j* and *k*, as shown in Fig. 5.4-b. This semantics is for example used by Badler *et al*. [BAD 87]. However, in general this is not implied by the minimization of Eq. (5.3): since at an unconstrained minimum its gradient must be zero, the only relationship that holds between the residual errors is

$$\sum_{i=1}^{p} w_i e_i(\boldsymbol{q}^*) \nabla e_i(\boldsymbol{q}^*) = \mathbf{0} \tag{5.5}$$

The presence of a derivative in this equation implies that, in general, no simple relationship can be established between the residual errors of conflicting tasks and their weights. An example is shown in Fig. 5.3: two conflicting tasks with same weight are applied to a simple 2D chain. In the least-squares configuration, the total residual error $e_{tot} = 190$ is unequally distributed on both tasks: the residual errors of the tasks are $e_1 = 75$ and $e_2 = 175$. Therefore, the weights do not have, in general, a clear semantics in task space.

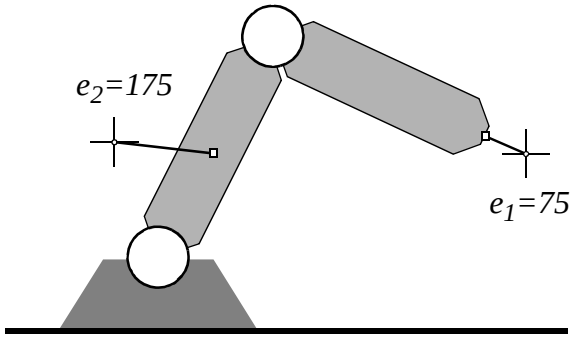


Figure 5.3 A least-squares solution to a simple problem with two conflicting tasks with equal weights ($w_1=w_2$). However, the residual errors e_1 and e_2 are not equal.

Moreover, merely weighting the tasks does not allow for a clear prioritization. When tasks get into conflict, the total residual error is distributed among all of them according to their weight. This implies that no task is exactly satisfied, unless a weight is highly predominant with respect to the others. It may be unpleasant for an animator to obtain a compromise between all tasks (Fig. 5.4-b), instead of strictly achieving at least the most important one(s) (Fig. 5.4-d). The relevance of this distinction becomes evident when one considers a collision avoidance task, which of course has top priority and must be achieved regardless of other tasks. Clearly, the weighting method is not always appropriate to handle multiple tasks. When needed, the animator should be able to impose an order of priority between the tasks.

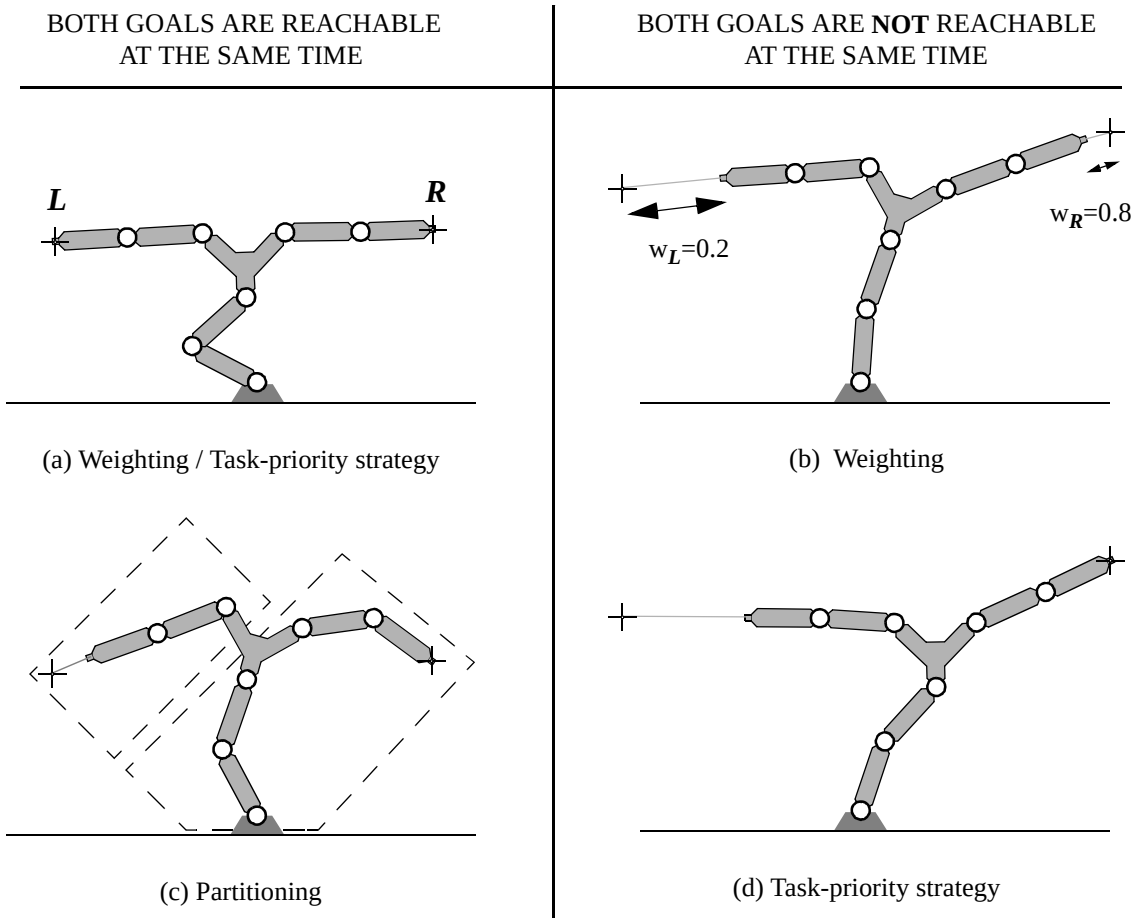


Figure 5.4 Three approaches to the management of task conflicts are illustrated on a seven DOF articulated chain whose left and right tips are controlled by tasks ***L*** and ***R***, respectively. Task ***L*** is considered less important than task ***R***. The goals are represented by crosses. When both are reachable at the same time, the weighting method allows to reach them all (a). Otherwise (b), the residual error is distributed on both tasks according to their weight ($w_{\boldsymbol{L}}$ and $w_{\boldsymbol{R}}$). The concept of priority is introduced in order to satisfy at least the most important task which is ***R***. An approach based on the partitioning of the set of joints is not entirely satisfactory, as perfectly attainable goals may not be reached (in (c), task ***L*** would be achievable). On the contrary, the task-priority strategy always provides satisfying solutions ((a) and (d)).

5.3 The task-priority strategy

To implement a priority mechanism, a simple but crude method suggested by Watt and Watt [WAT 92] (p. 383) is to allocate each joint solely to the task with highest priority among those depending on that joint. While this *partitioning* of the set of joints ensures the respect of priority, it is too restrictive since a valid solution satisfying all tasks may not be found (Fig. 5.4-c). This happens especially when the number of joints shared by several tasks is high.

In robotics, a task-priority strategy has been developed to deal with conflicts directly at the differential level. This task-priority strategy is more flexible, and provides satisfying solutions

whether the tasks are in conflict or not. When they can be achieved simultaneously, all are satisfied (Fig. 5.4-a), otherwise the top priority task reaches its goal without being perturbed, while the residual error of the other task is minimized (Fig. 5.4-d).

The algorithm is a natural extension of the numerical method presented in Chapter 4: a joint variation vector $\Delta\boldsymbol{q}$, hereafter simply called *solution*, is computed at each iteration taking into consideration a set of tasks, now ordered by priority.

First the exploitation of the null space to perform two or more tasks with lower priority is exposed. Then, we introduce a recursive relation to speed-up the process, and the resulting algorithm is summarized.

5.3.1 A formulation for managing a pair of tasks with different priorities

In Section 4.7.4, we have seen how to exploit the redundancy left available by the satisfaction of a task by means of a vector, noted $\boldsymbol{z}$, projected onto the null space of the Jacobian matrix. Another interesting exploitation of this vector is to express a second task with lower priority. The first attempts to do this are due to Hanafusa *et al.* [HAN 81] [NAK 87] and to Maciejewski *et al.* [MAC 85]. Given two tasks characterized by $J_1\Delta\boldsymbol{q} = \Delta\boldsymbol{x}_1$ and $J_2\Delta\boldsymbol{q} = \Delta\boldsymbol{x}_2$, the following solution allows the first task to take priority over the second one when a conflict arises:

$$\Delta\boldsymbol{q} = J_1^{\dagger}\Delta\boldsymbol{x}_1 + P_{N(J_1)}\boldsymbol{z} \tag{5.6}$$

where

$$\boldsymbol{z} = (J_2P_{N(J_1)})^{\dagger}(\Delta\boldsymbol{x}_2 - J_2J_1^{\dagger}\Delta\boldsymbol{x}_1) \tag{5.7}$$

is the minimum-norm vector that, among all the vectors belonging to the null space of J_1, minimizes the tracking error of the second task. This is known as a *constrained least squares* solution [BEN 74]. From this description it is clear that the projector in Eq. (5.6) is redundant, as $\boldsymbol{z}$ already belongs to $N(J_1)$ (this is formally proved in [MAC 85]).

This solution can be rewritten as

$$\Delta\boldsymbol{q} = J_1^{\dagger}\Delta\boldsymbol{x}_1 + \tilde{J}_2^{\dagger}\Delta\hat{\boldsymbol{x}}_2 \tag{5.8}$$

where

$$\begin{aligned} \tilde{J}_2 &= J_2P_{N(J_1)} \\ \Delta\hat{\boldsymbol{x}}_2 &= \Delta\boldsymbol{x}_2 - J_2J_1^{\dagger}\Delta\boldsymbol{x}_1 \end{aligned} \tag{5.9}$$

These two components can be interpreted as follows. First, $\tilde{J}_2$ is the restriction of the linear transformation J_2 to the null space of J_1: hence, $R(\tilde{J}_2^{T})$ is the subspace available for the second task to perform without interfering with the first task. Second, the effective increment $\Delta\hat{\boldsymbol{x}}_2$ is an adjustment of the desired increment for the second task $\Delta\boldsymbol{x}_2$ in order to compensate for the displacement $J_2J_1^{\dagger}\Delta\boldsymbol{x}_1$ due to the achievement of the first task.

It can be noted that the two terms that compose Eq. (5.8) are independent because

$$R(J_1^T) \cap R(\tilde{J}_2^T) = \{\mathbf{0}\} \tag{5.10}$$

This guarantees the respect of the priority of task one over task two.

5.3.2 The occurrence of algorithmic singularities

A serious problem with Eq. (5.8) is not apparent at first sight: the matrix $\tilde{J}_2$ may present singularities in addition to those of J_2, more precisely when

$$rank(\tilde{J}_2) < rank(J_2)$$

In terms of subspaces, these so-called *algorithmic* singularities occur at configurations where the ranges of the single tasks overlap:

$$R(J_1^T) \cap R(J_2^T) \neq \{\mathbf{0}\} \tag{5.11}$$

This means that some component of the solution necessarily affects both tasks and that no solution allows to solve them independently. In other words, a conflict between tasks of different priorities appears as particular singularities of $\tilde{J}_2$.

The undesirable effects discussed in Section 4.7.6 for *kinematic* singularities also appear in the proximity of algorithmic singularities. The ability to deal with algorithmic singularities is essential, since they are inevitable in a conflicting situation. Fortunately, the damped least squares technique, already used in Section 4.7.6 to deal with kinematic singularities, can also be applied to manage algorithmic singularities, hence unifying the management of both kinds of singularities. Therefore, Eq. (5.3) becomes:

$$\Delta \boldsymbol{q} = {J_1}^{\dagger^{\lambda_1}} \Delta \boldsymbol{x}_1 + (J_2 P_{N(J_1)})^{\dagger^{\lambda_2}} (\Delta \boldsymbol{x}_2 - J_2 {J_1}^{\dagger^{\lambda_1}} \Delta \boldsymbol{x}_1) \tag{5.12}$$

5.3.3 Another task-priority formulation

More recently, Chiaverini [CHI 94] [CHI 97] and Boulic *et al.* [BOU 96] have independently proposed an alternative solution that overcomes the effects of algorithmic singularities. The secondary solution ${J_2}^{\dagger} \Delta \boldsymbol{x}_2$ is first evaluated separately, and then projected on $N(J_1)$ to remove the components that would interfere with the high priority task:

$$\Delta \boldsymbol{q} = {J_1}^{\dagger} \Delta \boldsymbol{x}_1 + P_{N(J_1)} ({J_2}^{\dagger} \Delta \boldsymbol{x}_2) \tag{5.13}$$

In [BOU 96], this scheme, called *cascaded control*, is extended with a joint space optimization term, where a vector $\boldsymbol{z}$ is supposed to exploit any redundancy left by the satisfaction of the two Cartesian tasks:

$$\Delta \boldsymbol{q} = {J_1}^{\dagger} \Delta \boldsymbol{x}_1 + P_{N(J_1)} ({J_2}^{\dagger} \Delta \boldsymbol{x}_2 + P_{N(J_2)} \boldsymbol{z}) \tag{5.14}$$

However, it can be shown that the additional term in Eq. (5.14) is not correct. While it

respects the high priority task, it violates the secondary task since:

$$R(P_{N(J_1)}P_{N(J_2)}) \cap R(P_{N(J_1)}J_2^T) \neq \{\mathbf{0}\}$$

The correct solution would be to project the vector $\boldsymbol{z}$ on $N(J_1) \cap N(J_2) = N\begin{pmatrix} J_1 \\ J_2 \end{pmatrix}$, so that no task is perturbed.

5.3.4 Theoretical comparison of both task-priority formulations

The formulation of Chiaverini (Eq. (5.13)) does not experience the problem of algorithmic singularities, since the two least-squares problems are solved independently. Unfortunately, as discussed in [CHI 97] and [BAE 98], this solution poorly tracks the secondary task, thus affecting the convergence of the iterative process. We found more effective to deal with the algorithmic singularities than to loose tracking accuracy.

To clarify the different behaviours of the two formulations, their respective solutions are now visualized with a simple problem in three different situations. Each task is represented as on Fig. 5.5, in a 2D joint variation space.

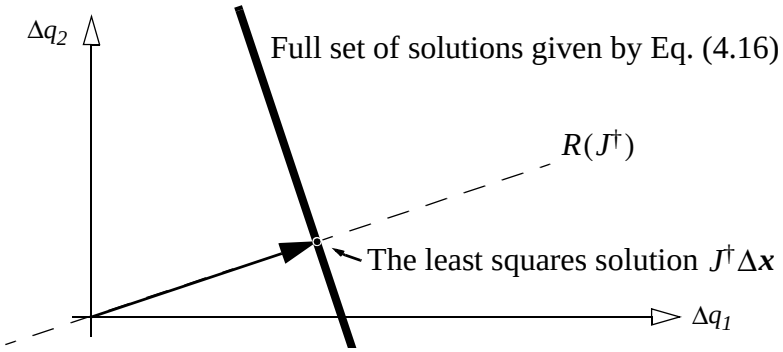


Figure 5.5 Representation of the set of solutions and of a particular, least-squares solution.

A graphical comparison of both formulations is given in Fig. 5.6 and Fig. 5.7. Two tasks, with different priorities, are represented in the 2D joint variation space. The high priority task and the low priority task are represented by black and grey lines respectively. The solution given by the first formulation is noted $\Delta \boldsymbol{q}_A$ (see Eq. (5.12)), while the solution of the second formulation is noted $\Delta \boldsymbol{q}_B$ (see Eq. (5.13)). We compare these solutions in three different situations: in Fig. 5.6, the two tasks are compatible, while in Fig. 5.7, they are nearly conflicting and in Fig. 5.8 they are totally conflicting. In all situations, both solutions lie on the null space of the high priority task: therefore the priorities are always respected. Now let us consider the first and second situations, where no conflict arise. The first formulation provides the intersection of the two solution spaces. This is equivalent to the following solution:

$$\Delta \boldsymbol{q} = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}^\dagger \begin{bmatrix} \Delta \boldsymbol{x}_1 \\ \Delta \boldsymbol{x}_2 \end{bmatrix}$$

However, this apparently optimal solution becomes problematic when an algorithmic singularity is approached: as seen in Fig. 5.7, its norm grows to infinity, since the solution spaces are becoming parallel to each other. Using the damped least-squares inverse allows to keep the norm within a reasonable limit, without violating the priority order.

The solution of the second formulation ($\Delta \boldsymbol{q}_B$) provides the orthogonal projection of the secondary task solution over the solution subspace of the primary task. Hence the problem of algorithmic singularities is avoided, and the solution obtained near conflicts (Fig. 5.7) is acceptable. However, as seen in Fig. 5.6, this solution poorly tracks the secondary task when the tasks are compatible: an unnecessary tracking error is introduced that results in slow convergence towards an optimal solution.

The last situation is shown in Fig. 5.8: we are in a conflicting situation. For the first formulation, we are exactly at an algorithmic singularity.

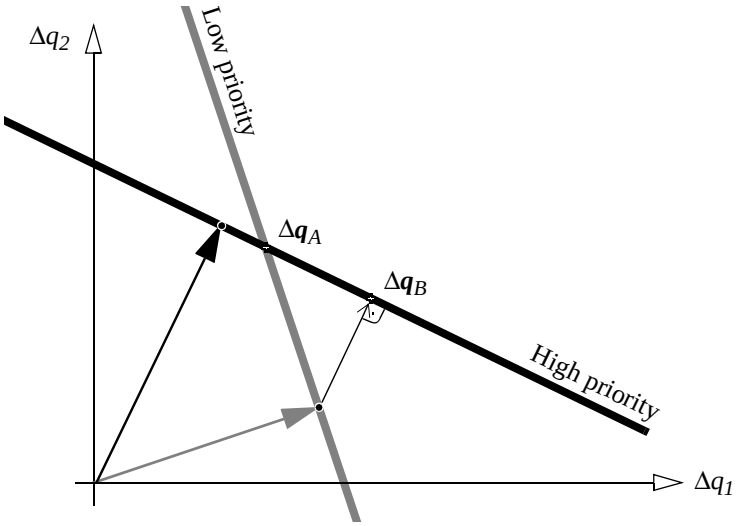


Figure 5.6 Comparison of the two formulations, when the tasks are compatible.

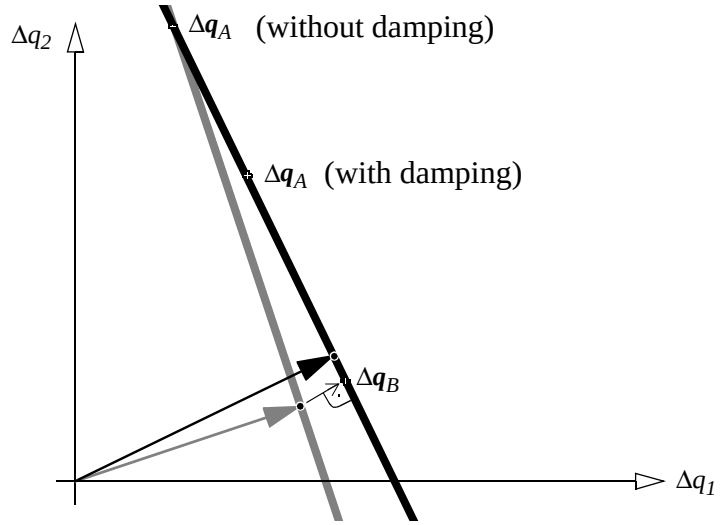


Figure 5.7 Comparison of the two formulations, when the tasks are nearly in conflict.

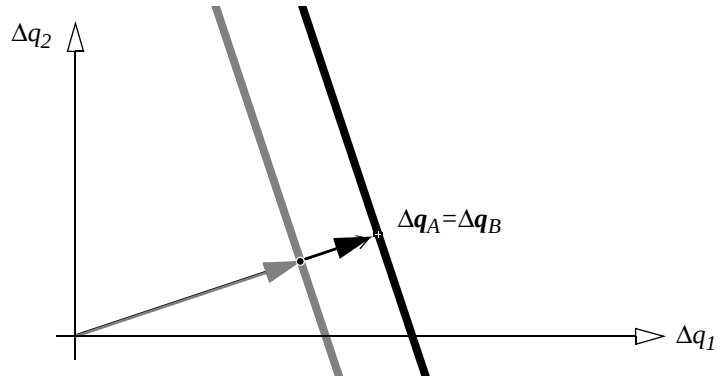


Figure 5.8 Comparison of the two formulations, when the tasks are in conflict.

5.3.5 Practical comparison of both task-priority formulations

To better understand the practical impact of the differences in the two formulations, consider a main and a secondary task that respectively control the position of two end-effectors $\boldsymbol{E_1}$ and $\boldsymbol{E_2}$ of a simple chain (as in Fig. 5.4). We have tested each formulation in the following three situations.

1 - Move $\boldsymbol{E_1}$ and consider the movement of $\boldsymbol{E_2}$

In the first test, the goal of the main end-effector $\boldsymbol{E_1}$ is moved and we consider the movement of the secondary end-effector $\boldsymbol{E_2}$. With the first formulation, $\boldsymbol{E_2}$ 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. In this case, $\boldsymbol{E_2}$ is no more fixed, but still quickly comes back to its initial point. Conversely, the use of the sec-

ond formulation leads to very large movements of E_2, ultimately starting to converge to its initial point only when we stop to move E_1.

2 - Move E_2 and consider the movement of E_1

In the second test, the goal of the secondary end-effector E_2 is moved and we consider the movement of the main end-effector E_1. As can be expected, with both formulations E_1 does not move since it has priority and thus is not perturbed by the secondary task. The difference lies in the way E_2 converges to its goal. As shown in Fig. 5.10, with the second formulation the path is less direct than with the first formulation. Furthermore, the displacements are anisotropic: the convergence is fast in some directions but very slow in others.

3 - Attract E_2 out of its reachable space

In the third test, the goal of the secondary end-effector E_2 is placed outside its reachable space. It is expected that E_2 attains the reachable space boundary and minimizes its distance to the goal g_2. This is what effectively happens with the first formulation. With the second one however, the end-effector first reaches the limit and then moves to a non-optimal position (see Fig. 5.9). This is because the sub-optimal solution computed at the differential level (see Section 5.3.4) affects the final solution to which the iterative process converges.

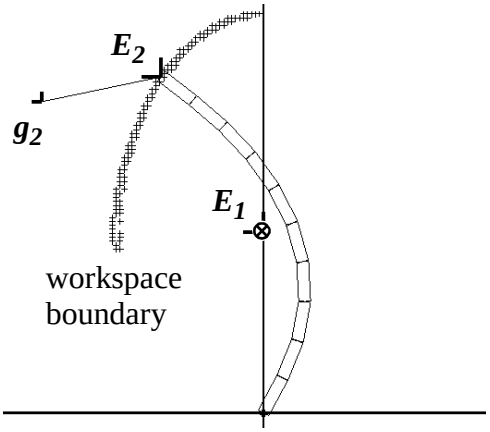


Figure 5.9 Final configuration obtained with the second formulation. This solution is not optimal for the secondary task.

In conclusion, while the second formulation overcomes the delicate problem of algorithmic singularities, the first formulation converges more directly towards a solution, which is a least-squares solution when the goal is unreachable.

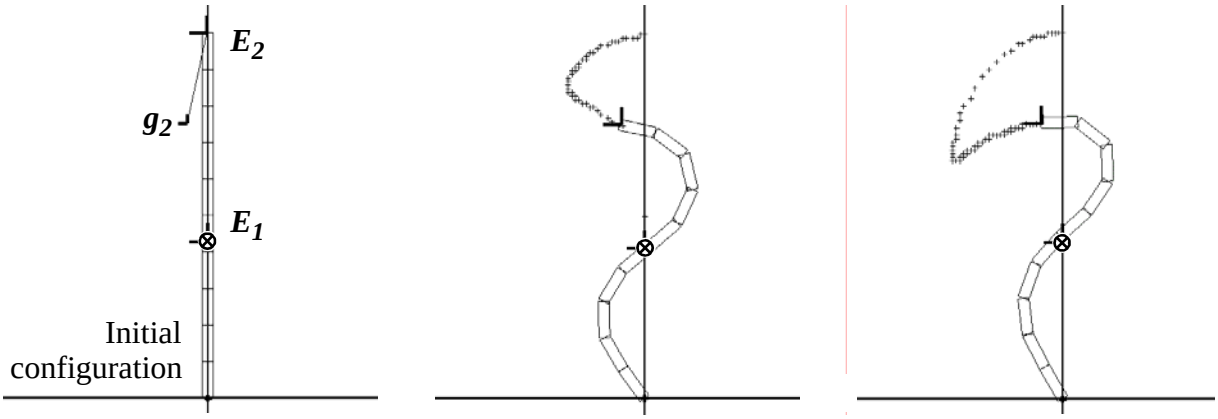


Figure 5.10 Comparison of the path taken by the low priority task (***E*****2**) to reach its goal, under the constraint that the center of mass (***E*****1**) must stay on the vertical line. Clearly, the path taken by the first formulation (middle figure) is shorter than the path taken by the second formulation (right figure). Note that this is a difficult situation for both formulations since the tasks are initially at a kinematic singularity.

5.3.6 Extension to multiple levels of priority

Following the same approach leading to Eq. (5.12), the method can be generalized to an arbitrary number of tasks with multiple levels of priority. Again, let us consider the set of p tasks defined in Section 5.1, now ordered from the highest priority ($i = 1$) to the lowest ($i = p$). We assume that they all have different priorities: this is not a limitation, since two tasks actually having the same priority may be lumped together, to form a single « augmented » task. In this case, weights could be used to set their relative importance (as in Eq. (5.4)).

To support more than two levels of priority, additional terms must be added to Eq. (5.12), one for each level. However, these *partial solutions* become difficult to express explicitly because they depend on the partial solutions of the higher priority tasks. As proposed by Siciliano and Slotine [SIC 91], a recursive scheme more easily expresses the solution $\Delta \boldsymbol{q} = \Delta \boldsymbol{q}_p$, where $\Delta \boldsymbol{q}_p$ is derived from

$$\begin{aligned} \Delta \boldsymbol{q}_i &= \Delta \boldsymbol{q}_{i-1} + \tilde{J}_i^{\dagger \lambda_i} (\Delta \boldsymbol{x}_i - J_i \Delta \boldsymbol{q}_{i-1}) \\ \Delta \boldsymbol{q}_1 &= J_1^{\dagger \lambda_1} \Delta \boldsymbol{x}_1 \end{aligned} \tag{5.15}$$

with

$$\tilde{J}_i := J_i P_{N(J_{i-1}^A)}$$

and where

$$P_{N(J_i^A)} := I_n - J_i^{A\dagger} J_i^A \tag{5.16}$$

is the projector on the null space of the so-called *augmented Jacobian*

$$J_i^A := \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_i \end{bmatrix} \tag{5.17}$$

Restricting each Jacobian J_i to $N(J_{i-1}^A)$ ensures that the partial solution will not perturb any task of higher priority, since $N(J_{i-1}^A) = N(J_1) \cap N(J_2) \cap \ldots \cap N(J_{i-1})$

5.3.7 The cost of the algorithm with multiple priority levels

Besides the cost of constructing Jacobian matrices, which is the same for any inverse kinematics resolution scheme, the dominant operation in terms of computational cost is the inversion of matrices, especially if the SVD is used for this purpose. In the algorithm, there are two matrix inversions at each priority level, the most expensive being the inversion of J_i^A within Eq. (5.16) since the size of this matrix increases from a level to the other. Hence, the computation of $P_{N(J_i^A)}$ represents a significant part of the total cost.

We note that $P_{N(J_i^A)}$ is the only part that is not computed in a recursive fashion: instead, it is computed "from scratch" at each priority level. In the following section we exploit a new recursion relation to incrementally compute the required sequence of projectors. This results in a more efficient algorithm. The benefit also comes from the fact that computing the increment is not expensive at all, as it is obtained from already available results.

5.3.8 Incremental computation of the projection matrices

Greville found a recursive algorithm in order to compute the least-squares inverse $J^\dagger$ of a matrix J, incorporating one column at a time in the partial result [GRE 60]. Actually, based on this well-known result, simple recursive formulas to compute projection matrices are easily obtained. They are quite useful in many application areas of the least-squares inverse [ROS 60] [ALB 66] [BOUL 71] [BEN 74] [KOH 89], especially when one is not only interested in the final result but also in the partial results, moving from one to the other with a minimum of operations. An example is the following recursive formula that computes the orthogonal projector on the null space of an $m \times n$ matrix K:

$$P_{N(K)} = \begin{cases} P_{N(J)} - \dfrac{\tilde{\boldsymbol{j}}\tilde{\boldsymbol{j}}^T}{\tilde{\boldsymbol{j}}^T\tilde{\boldsymbol{j}}} & \text{if } \tilde{\boldsymbol{j}} \neq \boldsymbol{0} \\ P_{N(J)} & \text{otherwise} \end{cases} \qquad \text{with } \tilde{\boldsymbol{j}} := P_{N(J)}\boldsymbol{j} \tag{5.18}$$

where J is an arbitrary $(m-1) \times n$ matrix, $\boldsymbol{j}$ is an arbitrary vector (of size n) and $K = \begin{bmatrix} J \\ \boldsymbol{j}^T \end{bmatrix}$.

The term which is subtracted from the projection matrix $P_{N(J)}$ is a matrix that orthogonally projects on the range of a non-zero vector $\tilde{\boldsymbol{j}}$:

$$P_{R(\tilde{j})} = \frac{\tilde{\boldsymbol{j}}\tilde{\boldsymbol{j}}^T}{\tilde{\boldsymbol{j}}^T\tilde{\boldsymbol{j}}} \tag{5.19}$$

A geometric interpretation of this recursion relation is given in Fig. 5.11.

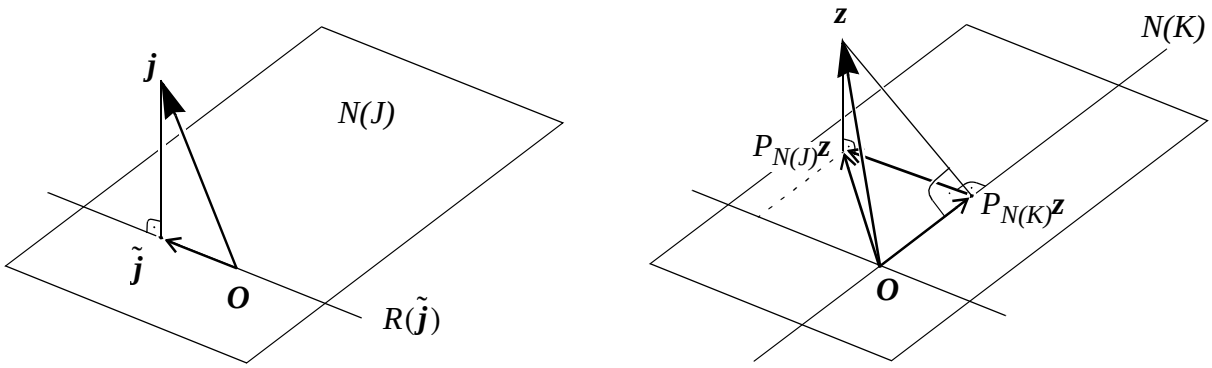


Figure 5.11 Geometric interpretation of Eq. (5.18), in a simple case where $m = 3$ and $n = 3$. On the left, a given three-dimensional vector $\boldsymbol{j}$ is projected on a given two-dimensional subspace *N(J)*. The result, noted $\tilde{\boldsymbol{j}}$, defines a subspace $R(\tilde{\boldsymbol{j}})$ that belongs to *N(J)*, and is the component that has to be removed in order to obtain a projection on subspace *N(K)*. On the right, the projectors on subspaces *N(J)* and *N(K)* are applied to an arbitrary three-dimensional vector $\boldsymbol{z}$. The difference between its projection on *N(J)* and on *N(K)* is its projection on the range of $\tilde{\boldsymbol{j}}$.

This recursion relation can be applied to each line of each Jacobian matrix J_i in order to compute the sequence of projection matrices needed in Eq. (5.15), from the highest priority level ($i=1$) to the lowest ($i=p$). For the same operation there exists a more compact expression that projects the Jacobian matrix in a single step:

$$\begin{aligned} P_{N(J_i^A)} &= P_{N(J_{i-1}^A)} - \tilde{J}_i^{\dagger}\tilde{J}_i \quad \text{with } \tilde{J}_i := J_i P_{N(J_{i-1}^A)} \\ P_{N(J_0^A)} &= I_n \end{aligned} \tag{5.20}$$

When J_i reduces to a single row matrix, Eq. (5.20) reduces to Eq. (5.18). We have proved this identity in Appendix G. As in Eq. (5.19), the increment is also a projection matrix since

$$P_{R(\tilde{J}_i^T)} = \tilde{J}_i^{\dagger}\tilde{J}_i \tag{5.21}$$

and it is efficiently computed from the SVD of $\tilde{J}_i$ (see Section 4.6). This SVD is already available since it is required for the computation of the damped inverse of $\tilde{J}_i$ in Eq. (5.15).

The sequence generated by Eq. (5.20) starts as follows:

$$
\begin{aligned}
P_{N(J_0^A)} &= I_n \\
P_{N(J_1^A)} &= I_n - J_1^\dagger J_1 \\
P_{N(J_2^A)} &= I_n - J_1^\dagger J_1 - (J_2 P_{N(J_1^A)})^\dagger (J_2 P_{N(J_1^A)}) \\
&\dots
\end{aligned}
$$

At each priority level, the joint variation space available for achieving the tasks with lower priority is appropriately reduced to prevent interference with the current task. Of course, at the beginning of the sequence, the whole space is available (hence $P_{N(J_0^A)} = I_n$).

5.3.9 Speedup obtained with the recursive formulas

We now estimate the speedup that results from the use of the recursive formulas (either Eq. (5.18) or Eq. (5.20)) with respect to a computation based on Eq. (5.16). For simplicity, we assume that the Jacobian matrices J_i all have the same dimension $m \times n$.

The cost of computing a projector for an $m \times n$ matrix is $k(m, n) \sim mn^2$. The cost of computing an increment relative to an $m \times n$ matrix J_i is the same, since it is a projector too. Hence, at the ith priority level, the old method costs $k(im, n)$, as $P_{N(J_i^A)}$ must be evaluated. On the other side, the recursive method only costs $k(m, n)$, as $P_{\tilde{R}(J_i^T)}$ must be evaluated.

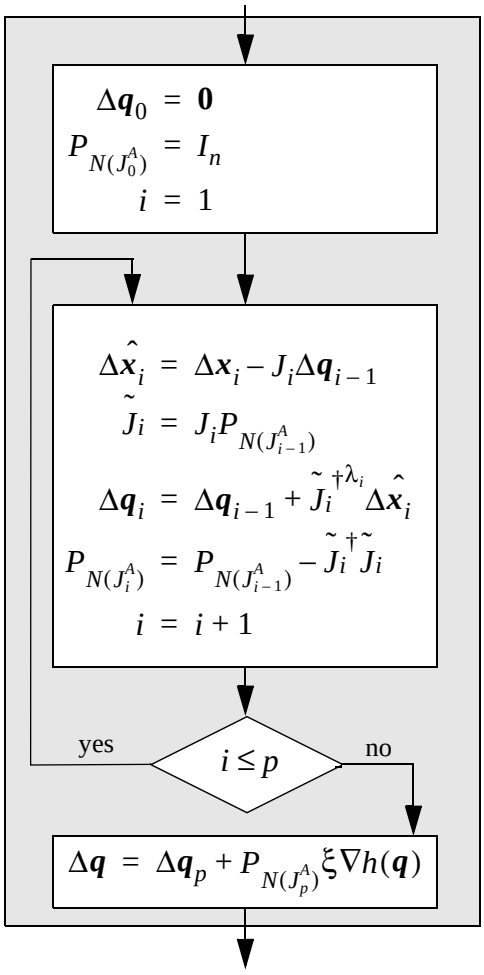
These costs are summed over all priority levels (from 1 to p):

$$
\begin{aligned}
total_{old} &= \sum_{i=1}^{p} k(im, n) = \frac{p(p+1)}{2} mn^2 \\
total_{recursive} &= \sum_{i=1}^{p} k(m, n) = pmn^2
\end{aligned}
$$

Thus the speed-up factor is roughly

$$
\frac{total_{old}}{total_{recursive}} = \frac{p+1}{2}
$$

Of course, for a single priority level (p=1) there is no speedup: the incremental version is truly interesting when many priority levels are used. For example, evaluating the nullspace projectors for five priority levels is about three times faster with the incremental algorithm than with the previous method.



Initialization of:
- solution vector
- projection matrix
- priority level index

Compensation of motion due to higher priority tasks.

Restriction by projection on $N(J_{i-1}^A)$.

Accumulation of the partial solutions.

Incremental update of the projector.

Move to the next lower priority level.

Add a criterion minimization term (with $\xi < 0$).

Figure 5.12 The recursive task-priority algorithm with p priority levels, traversed from the highest to the lowest, and a joint space criterion $h(\boldsymbol{q})$ to be minimized.

5.3.10 Summary of the recursive task-priority algorithm

The algorithm is summarized in Fig. 5.12. The solution $\Delta \boldsymbol{q}$ is an accumulation of the partial solutions evaluated at the different levels of priority, traversed from the highest priority to the lowest. At each such level i, the following operations are performed. First, the desired increment in task coordinates $\Delta \boldsymbol{x}_i$ is adjusted in order to *compensate* for the displacement due to the partial solutions evaluated at the higher priority levels. Second, the partial solution best matching the adjusted increment $\Delta \hat{\boldsymbol{x}}_i$ is found, but the search is *restricted* to the null space of J_{i-1}^A in order not to perturb higher priority tasks. Third, the projection matrix is updated by removing the component relative to the subspace exploited at the current level (which is the range of $\tilde{J}_i^T$), so that it is no longer available for tasks of lower priority. When all levels of priority have been traversed, a criterion minimization term is added in order to take advantage of any remaining redundancy.

5.3.11 Convergence of the algorithm and semantics of the solution

In Section 4.8 we have seen that the algorithm with a single task converges towards a least-squares solution of the residual error function, which is the desired behaviour. Although we have no formal proof, we have empirically verified that the task-priority algorithm also converges towards a solution that satisfies least-squares criteria for all the tasks. Except for the top priority task, the residual error of each task is minimized under the constraint of not affecting higher priority tasks.

For any pair of tasks with different priority, the algorithm produces the behaviours summarized in Fig. 5.13. In practice, they correspond to an intuitive understanding of the concept of priority. In addition to respecting the priority order, the tasks are always satisfied as much as possible (in the sense that their residual error is locally minimized). Also, for two (or more) tasks lying at the same priority level, the algorithm converges towards a (possibly weighted) least-squares solution, which is a compromise solution when a conflict arises (see Section 5.2).

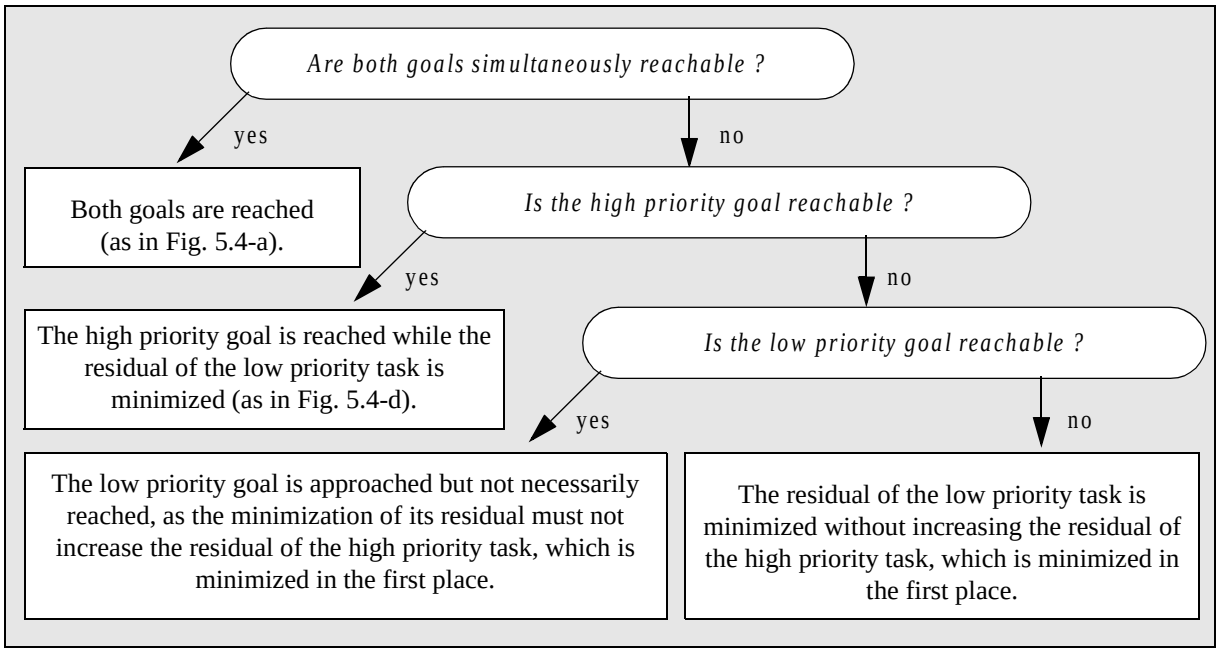


Figure 5.13 Semantics of the task-priority solution with two tasks of different priority.

5.3.12 Speed of the algorithm

In this section we evaluate the computational complexity of the task-priority algorithm. The measurements have been performed on an SGI Octane with an R10000 processor (195 MHz). The algorithm has been implemented in C language. We only consider the time required to compute an increment $\Delta \boldsymbol{q}$, and we ignore the computation of the Jacobian matrices. The time required to find a final configuration $\boldsymbol{q}^*$ is more difficult to evaluate as the number of iteration steps is highly dependent on the initial configuration.

The time required for the evaluation of $\Delta \boldsymbol{q}$ with the task-priority algorithm depends on the number of priority levels (p) and the size of the Jacobian matrices J_i. For simplicity, we assume that all Jacobian matrices have the same size ($m \times n$). Hence, the time required for the evaluation of $\Delta \boldsymbol{q}$ is mainly a function of m, n and p (it may also depend on the content of the Jacobian matrices, but here we consider "average" Jacobian matrices).

Fig. 5.14 shows several evaluations of this function.

- Graph **A** shows how the cost evolves according to the number of priority levels, p. The filled rectangles represent the result obtained with the fully recursive formulation, while the hatched rectangles represent the result obtained with the non-recursive formulation. Clearly, the recursive formulation is faster, by a factor proportional to p: this is in accordance with the theoretical prediction obtained in Section 5.3.9.
- The graphs **B** and **D** show how the computational cost evolves according to the number of degrees of freedom, n. It is a non-linear relationship, close to $O(n \log n)$.
- Graph **C** shows the dependence with respect to the number of constraints m at each priority level (i.e. the "height" of each Jacobian matrix). If $m < n$, the dependence is linear.
- Finally, graph **E** shows that the time required for a problem with a given number of tasks is reduced when they are distributed on several priority levels.

A typical situation is represented by the following parameters: m=15, n=50, p=4. In that case, the time required to compute a step $\Delta \boldsymbol{q}$ is ~ 5 ms: hence the computation can be performed 200 times a second (again, without considering the evaluation of the Jacobian matrices).

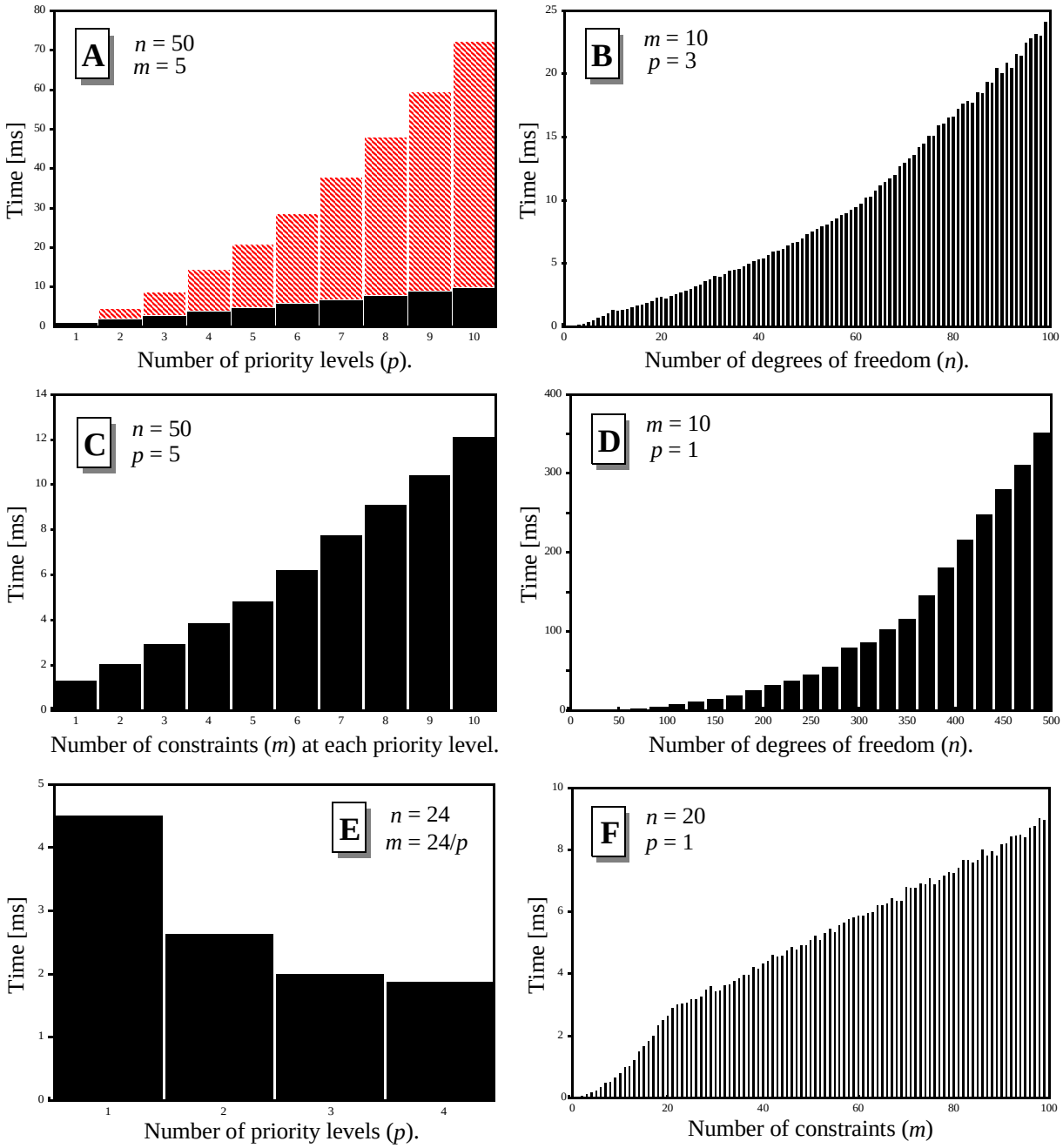


Figure 5.14 Empirical evaluation of the computational load of the task-priority algorithm.

5.4 Constraining the task-priority solution

The task-priority algorithm can be further extended to ensure that a set of equality and inequality constraints are satisfied after each iteration step. This is necessary for "hard" constraints such as anatomical joint couplings or joint limits that should never be violated (see Section 3.8). We consider a set of *linear* equality and inequality constraints:

$$\boldsymbol{c}_i^T \boldsymbol{q} \;=\; b_i \qquad \textit{for } i=1..g \tag{5.22}$$

$$\boldsymbol{c}_i^T \boldsymbol{q} \leq b_i \qquad \text{for } i=g+1..h \tag{5.23}$$

where the $\boldsymbol{c}_i$'s are n-dimensional vectors and the b_i's are scalars. Eq. (5.23) allows simple lower bounds and upper bounds on joint variables, as well as linearly coupled joint limits. Eq. (5.22) permits linearly coupled joint variables. This set of constraints defines a convex space of feasible configurations, which is supposed non-empty. The choice of linearity is due to the higher complexity introduced by non-linear constraints.

The management of these constraints must be integrated within the process that computes the step $\Delta\boldsymbol{q}$. Ignoring the constraints at this stage and then adjusting the resulting configuration in order to satisfy them, leads to undesirable solutions. This has already been noted by Boulic and Mas [BOU 97] [MAS 96] for the management of joint limits: merely truncating the exceeding joint values which are beyond their limit introduces a bias into the solution, since the optimality of the solution holds only for the constraints explicitly taken into account by the algorithm.

In the context of the task-priority algorithm, additional hard constraints can be seen as tasks of "infinite" priority (i.e. of higher importance than any other task). The next sections explain how to integrate them in the algorithm.

5.4.1 Linear equality constraints

We start with the equality constraints. Because of the hypothesis of linearity, they can be ensured in a single iteration step. The set of linear constraints is easily integrated within the task-priority scheme, because they are just a special case of the non-linear task (Eq. (5.1)) already handled by the algorithm. We modify its initialization phase (see Fig. 5.12) as follows:

$$\begin{aligned} \Delta\boldsymbol{q}_0 &= C^{\dagger}(\boldsymbol{b} - C\boldsymbol{q}) \\ P_{N(J_0^A)} &= P_{N(C)} = I_n - C^{\dagger}C \end{aligned} \tag{5.24}$$

where $C = \left[\boldsymbol{c}_1 \ \ldots \ \boldsymbol{c}_g\right]^T$ and $\boldsymbol{b} = \left[b_1 \ \ldots \ b_g\right]^T$.

This ensures that the solution lies in the constraint subspace defined by $C\boldsymbol{q} = \boldsymbol{b}$.

The first term, $\Delta\boldsymbol{q}_0$, is the displacement required to reach the constraint, if this is not already the case. This term is also useful to absorb possible numerical drift away from the constraint, that is accumulated with the iterations. This displacement, which is supposed to be small, is automatically compensated into $\Delta\hat{\boldsymbol{x}}_1$, as if it was due to the achievement of a task (see Fig. 5.12). The second term, $P_{N(J_0^A)}$, ensures that the resulting solution lies on the constraint subspace.

5.4.2 Linear inequality constraints

Now we address the integration of inequality constraints, typically required to ensure joint limits (see Fig. 5.15). Inequality constraints are more difficult to manage than equality constraints. In general, they are dealt with an iterative method, typically an *active set* algorithm [BJO 96]. Such algorithms are well described in the literature of constrained least-squares methods, hence we only summarize the principle.

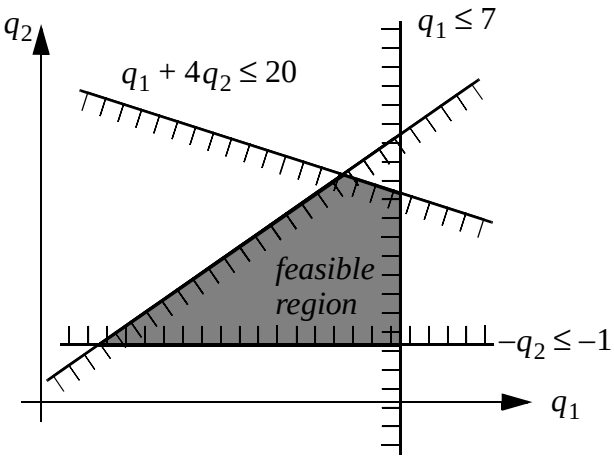


Figure 5.15 Four linear inequality constraints define a feasible region of the joint space.

The principle of an active set algorithm is that at a solution $\boldsymbol{q}$ a certain subset of the inequality constraints will be *active*, i.e. satisfied with equality. Hence, the inequality-constrained problem can be reduced to an equality-constrained problem. However the set of *active* constraints is not known a priori, and must be determined by a sequence of trials that test if a prediction of the active set (called the working set) is valid or not (see Fig. 5.16). Initially, the working set only contains the equality constraints of Eq. (5.22). Starting from an initial feasible configuration, a step $\Delta\boldsymbol{q}$ is computed and the resulting configuration is checked against the inequality constraints. If one or more of them have been violated, a new solution must be evaluated, with an updated working set. Only the violated constraint which is closest to the initial configuration is introduced in the working set, and thus a row is added to the matrix C and to the vector $\boldsymbol{b}$. Hence, both $\Delta\boldsymbol{q}_0$ and $P_{N(J_0^A)}$ must be re-evaluated[1], and a new solution is computed. This iterative process loops until the solution satisfies all inequality constraints. Usually only few iterations are required. An example is shown in Fig. 5.17.

An interesting characteristic of this method is that when a joint limit is overstepped during a discrete displacement $\Delta\boldsymbol{q}$, the joint is precisely set on the boundary thanks to the adjustment $\Delta\boldsymbol{q}_0$, instead of staying at its current position. This is important for joints such as the knee, whose extension limits correspond to a very common posture of the leg.

1. See Appendix H for an efficient recursion relation to incrementally update these two quantities.

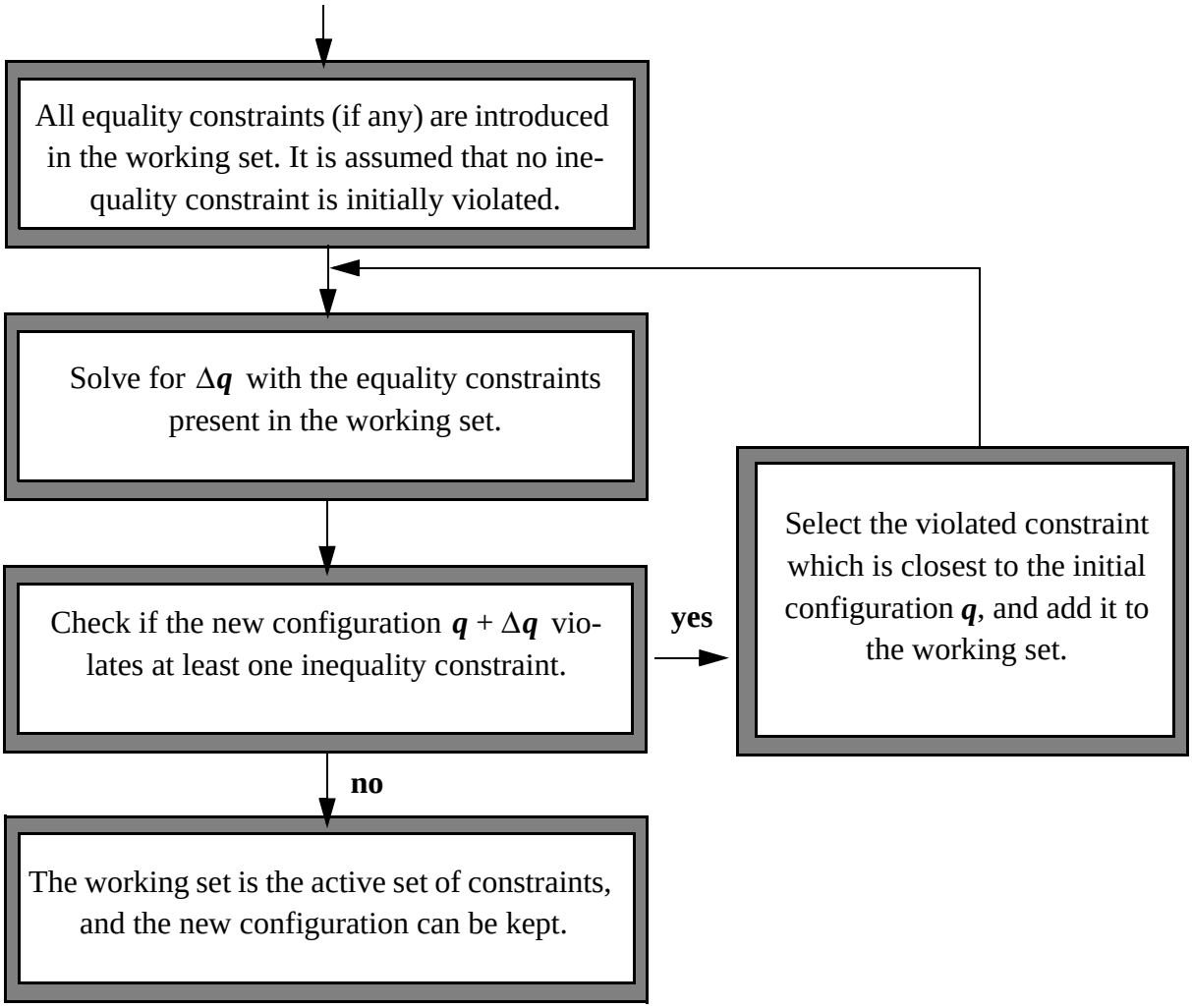


Figure 5.16 Overview of an active set algorithm, which handles linear inequality constraints.

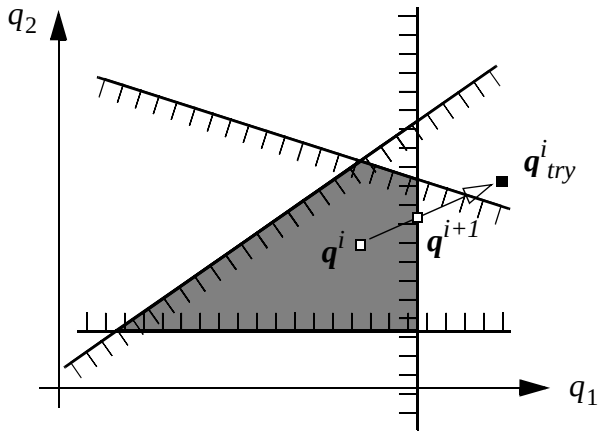


Figure 5.17 An example of two violated constraints, when moving from an initial feasible point q^i to a new (invalid) point q^i_{try}. The next valid configuration is q^{i+1}.

5.4.3 Imposing non-linear hard constraints

Joint limits defined as spherical ellipses or as spherical polygons are not linear with respect to the joint parameters $\boldsymbol{q}$. Non-linear coupling between joint variables is also useful: for example, if a quaternion parametrization is selected for a ball-and-socket joint, the quaternion must be constrained to lie on the unit four-dimensional hypersphere, which is a quadratic constraint. Unfortunately, dealing with non-linear constraints is a difficult problem.

We propose a partial solution inspired by the paper of Rosen on non-linear programming techniques [ROS 61] that deals with non-linear constraints. The constraint is linearized at the current point and considered as a linear constraint in the resolution process. However, after a step, the constraint may have been violated because of its non-linearity. A correction is therefore necessary to bring the configuration back on the constraint. In principle, this correction is small since the step taken from the initial configuration is small. At least, the correction is much smaller than without considering the constraint at all in the optimization process.

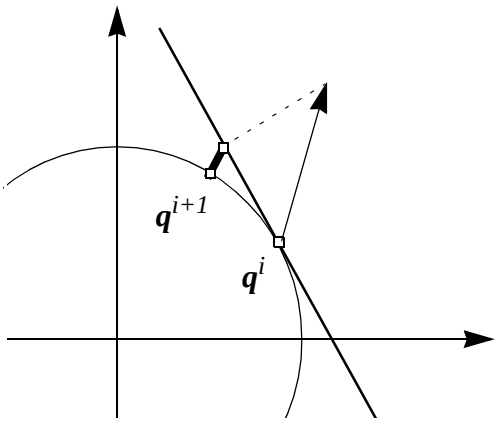


Figure 5.18 A quadratic constraint, and its linearization at a given point $\boldsymbol{q}^i$.

An example is given in Fig. 5.18 for a non-linear equality constraint. It could be a 2D slice of the quaternion space used to parametrize a ball-and-socket joint. The configuration $\boldsymbol{q}$ is constrained to lie on the unit circle, while the step $\Delta\boldsymbol{q}$ is constrained to lie on the tangent space, which is a line passing through the current configuration $\boldsymbol{q}^i$. After the step, the configuration still lies on the tangent space but not on the circle, hence it is projected on the circle in order to obtain a valid configuration $\boldsymbol{q}^{i+1}$. For the next iteration, a new linearization about $\boldsymbol{q}^{i+1}$ must be of course performed.

5.5 Summary and conclusion

In this chapter, two methods for the resolution of task conflicts have been presented: they are complementary techniques, and cohabit in the resolution framework that we have presented. The weighting strategy provides a compromise solution: this is often acceptable when the tasks do not have radically different purposes. In other situations, the weighting strategy can lead to unsatisfying situations because a task is clearly more important than another, and no weights may reflect this relationship. The introduction of an order of priority to arbitrate the conflicts yields a new family of pertinent solutions: this will be illustrated in Chapter 7.

The next chapter discusses a task for the balance control of a figure that exerts and supports forces, in addition to its body weight. This task is of course relevant for animators, because body balance is an important characteristic of human postures. This is an example of task which has priority over less critical functional tasks, because it regards the safety of the figure.

Chapter 6

Posture control with force exertion

6.1 Introduction

In this chapter, we are going to investigate the interaction of a figure with its environment through a set of forces. Since we are dealing with postures and not motions, only static forces are considered. These include body weight, supporting forces, and also forces exerted to perform tasks such as pushing, pulling, holding an object or manipulating a tool. In principle, using forces as a means to control the posture is appealing since there is clearly a relationship between the capability to exert forces and the posture adopted to perform the task (see Fig. 6.1).

Forces are related to the posture mainly for two reasons. First, they are bound by the laws of statics, that determine if a posture is balanced or not, for a given set of forces. The dominant role is played by the weight force, that constrains the position of the center of mass of the figure. Second, the posture is also related to the internal joint torques required to resist to (or to exert) the forces. This makes the relationship between force and posture a highly complex one for human or animal models, since it introduces the intricacies of their musculo-skeletal system.

In ergonomics studies, it is essential to consider the forces required to perform a given task, to evaluate its feasibility. An example of analytical tool to evaluate postures that require the exertion of forces is the *Postural Stability Diagram* (PSD) of Grieve [GRI 79] [GRI 79b]: personal and environmental constraints on the exertion of forces are reported on a 2D diagram, and can be studied for a posture, in a static context. Moreover, an important information for ergonomists is the ability to determine the maximal force that an operator is capable of exerting in a given direction: this also strongly depends on the selected posture and on available strength, as shown by the experiments of Haslegrave [HAS 90].

Of course, roboticians are also highly concerned with the control of the contact forces occurring between a robot manipulator and its environment [CRA 89] [MUR 94]. Moreover, for multi-legged robots a correct distribution of their weight over the supporting sites is required to ensure their stability.

In the context of articulated figures manipulation, the specification of the interaction forces can be exploited to automatically ensure the balance of the figure for the task being performed, hence improving the realism of the posture. In this chapter, we present a few mechanisms for this purpose, that nicely fit within the previously established inverse kinematics framework.

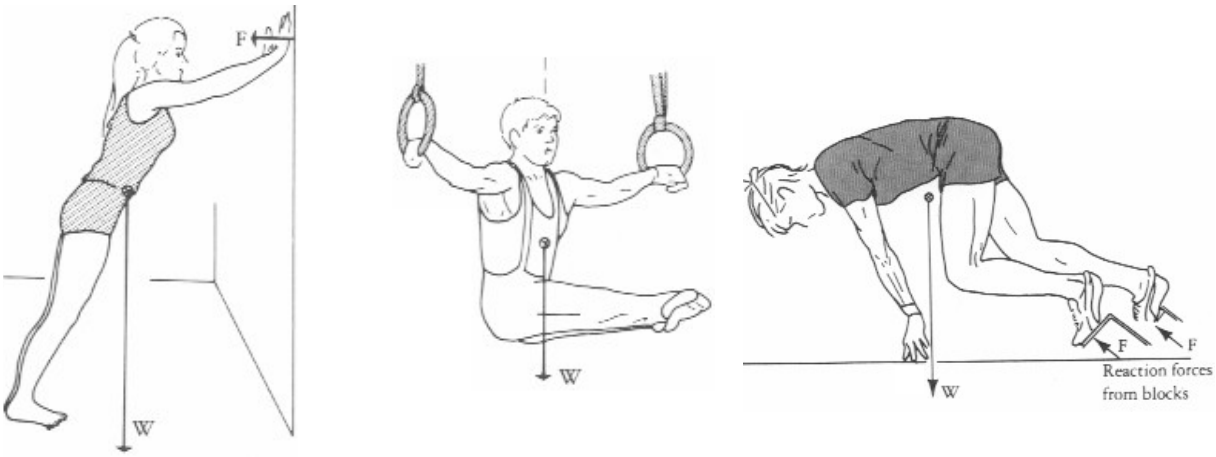


Figure 6.1 Human body postures with force exertion (source: [KRE 90]).

6.2 Force interaction between the figure and its environment

Let us consider the external forces acting on the figure[1]. Besides its weight $\boldsymbol{w}$ applied at the center of mass $\boldsymbol{G}_{tot}$, the force interaction with the environment occurs at "interfaces" where the figure applies a force on the environment and, by the action-reaction principle, the environment applies the opposite force on the figure.

The number of interfaces, hereafter called *sites*, is noted *f*. We can distinguish between *supporting sites* (typically the feet) that bear a significant part of body weight, and *task sites* (typically the hands) where a force (such as push or pull) is exerted to perform a task. To be consistent with the weight force, we only think in terms of forces exerted by the environment upon the figure. Thus, we introduce a force $\boldsymbol{f}_i$ and a torque $\boldsymbol{t}_i$ exerted by the environment at a point $\boldsymbol{E}_i$, which is the centroid of the site area. We also introduce the *wrench* $\widehat{\boldsymbol{f}_i}$ associated to this force exertion, that groups both linear and angular components [MUR 94]:

$$\widehat{\boldsymbol{f}_i} := \begin{bmatrix} \boldsymbol{f}_i \\ \boldsymbol{t}_i \end{bmatrix} \in \Re^6$$

1.Note that here we do not consider the internal Cartesian forces exerted by a body part over another body part. Such forces do not affect the balance conditions since, by the action-reaction principle, they cancel each other out. However, for completeness, they should be considered for the computation of the total joint torques in Section 6.4 (this has not been done).

6.2.1 The laws of statics

Since we are dealing with postures and not motions, the principles of statics can be applied. The figure is in static equilibrium when the following two conditions hold:

$$\boldsymbol{w} + \sum_{i=1}^{f} \boldsymbol{f}_i = \boldsymbol{0} \tag{6.1}$$

$$\overrightarrow{\boldsymbol{OG}_{tot}} \times \boldsymbol{w} + \sum_{i=1}^{f} (\overrightarrow{\boldsymbol{OE}_i} \times \boldsymbol{f}_i + \boldsymbol{t}_i) = \boldsymbol{0} \tag{6.2}$$

An example of such a situation is given in Fig. 6.2.

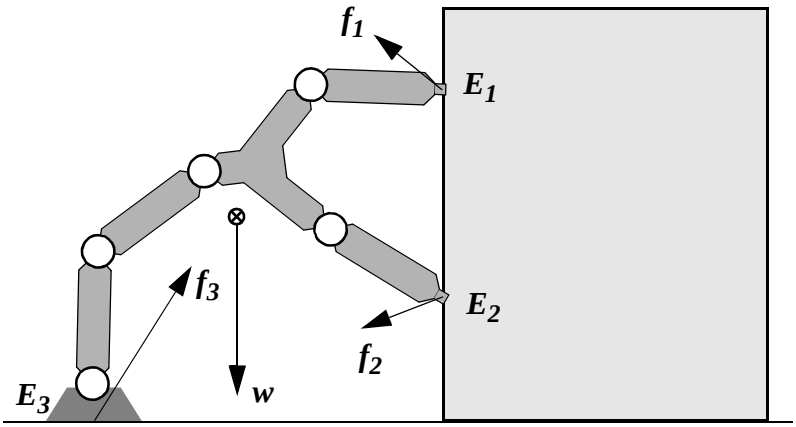


Figure 6.2 An articulated figure in static equilibrium, while pushing a heavy object.

6.2.2 Specifying the forces acting on the figure

Each force acting on the figure must be specified by a vector and a point of application, called end-effector. The position and orientation of an end-effector is often controlled by inverse kinematics, but this is not mandatory: a force can also be applied to an end-effector which is unconstrained or only partially constrained. Thus the kinematic control and the force control of an end-effector are independent, and all combinations are possible. The center of mass is also seen as an end-effector on which the weight vector ***w*** is applied.

While the weight vector ***w*** is usually kept constant, the other forces may be modified by the user at any time, to study their impact on the posture. Because of the first equilibrium condition Eq. (6.1), a modification of a force at a site must be compensated at another site. Our solution is to select the site located at the root of the hierarchy, which is fixed by definition and hence does not need to be constrained by inverse kinematics. Still, it is generally a supporting site (as site $\boldsymbol{E}_3$ in Fig. 6.2). Hence, the associated force, rather than being modified directly by the user, can be automatically updated in order to keep Eq. (6.1) satisfied (the satisfaction of Eq. (6.2)

will be considered in Section 6.3).

To ease the force specification process, two additional simple mechanisms can be introduced.

The first one, also used by Boulic *et al.* [BOU 97b] [MAS 96], is to proportionally distribute the vertical weight force on all supporting sites, according to a ratio s_i assigned to each site. The sum of all those ratios must be equal to 1, but there is no restriction on their value. A negative ratio indicates that an additional vertical force is applied (for example, to integrate the weight of an external object being carried). Of course, the horizontal component of the exerted force is still directly controlled by the user. The advantage of this mechanism is that the specification of the vertical and horizontal components are decoupled, and thus allows a more precise control of the weight distribution over the sites.

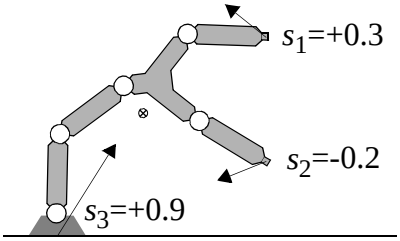


Figure 6.3 Weight distribution example (see Fig. 6.2).

The second mechanism constrains the force exerted by the supporting surface to lie within a so-called *friction cone* whose axis is perpendicular to the surface, and whose angle of aperture α (typically less than 45^o) is related to the static coefficient of friction $\mu = \tan(\alpha)$ used in the Coulomb friction model. To some extent, this mechanism prevents the user from specifying unrealistic forces, since limits on friction are an important constraint for horizontal force exertion such as push and pull [KRO 74].

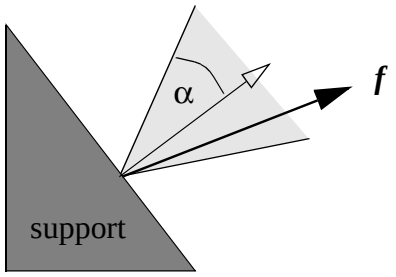


Figure 6.4 The friction cone (side view).

6.3 Ensuring the static equilibrium conditions

Provided that a set of forces satisfying Eq. (6.1) has been specified, Eq. (6.2) still has to be satisfied in order to have a static equilibrium situation. This condition on the moments depends on the configuration $\boldsymbol{q}$ of the figure, since the positions of the application points $\boldsymbol{E}_i$ and of the center of mass $\boldsymbol{G}_{tot}$ depend on $\boldsymbol{q}$. Hence, a constraint must be imposed on the configuration in order to satisfy Eq. (6.2) as well, when this is possible.

To solve this problem, Aydin *et al.* [AYD 99b] [AYD 99c] introduced a new task (along with its Jacobian matrix) to control the total torque about a fixed point, typically the main point of support (i.e. that bears the most weight). Constraining this torque to be zero ensures that the system is in static equilibrium.

We decided to use the same method to ensure the static equilibrium condition. However, we give an alternative derivation of the Jacobian matrix for this task type, that deals with the weight force in a particularly simple manner thanks to the kinetic Jacobian matrix.

6.3.1 Derivation of the Jacobian matrix for torque control

First, the task function $\boldsymbol{t}(\boldsymbol{q})$ representing the Cartesian torque about the origin $\boldsymbol{O}$ of some fixed reference frame is stated:

$$\boldsymbol{t}(\boldsymbol{q}) = \overrightarrow{\boldsymbol{OG}_{tot}(\boldsymbol{q})} \times \boldsymbol{w} + \sum_{i=1}^{f} (\overrightarrow{\boldsymbol{OE}_i(\boldsymbol{q})} \times \boldsymbol{f}_i + \boldsymbol{t}_i) \tag{6.3}$$

All quantities are expressed in the reference frame. To control this function with the inverse kinematics technique described previously, we need to derive its $3 \times n$ Jacobian matrix:

$$J_{\boldsymbol{t}}(\boldsymbol{q}) := \frac{d\boldsymbol{t}(\boldsymbol{q})}{d\boldsymbol{q}} \tag{6.4}$$

Hence:

$$J_{\boldsymbol{t}}(\boldsymbol{q}) = \frac{d}{d\boldsymbol{q}}(\overrightarrow{\boldsymbol{OG}_{tot}(\boldsymbol{q})} \times \boldsymbol{w}) + \sum_{i=1}^{f} \frac{d}{d\boldsymbol{q}}(\overrightarrow{\boldsymbol{OE}_i(\boldsymbol{q})} \times \boldsymbol{f}_i + \boldsymbol{t}_i) \tag{6.5}$$

Now, we exploit the following vector identity:

$$\frac{d}{d\boldsymbol{q}}(\boldsymbol{a}(\boldsymbol{q}) \times \boldsymbol{b}) = [-\boldsymbol{b}\times]\frac{d\boldsymbol{a}(\boldsymbol{q})}{d\boldsymbol{q}}$$

This identity allows us to rewrite Eq. (6.5) as:

$$J_{\boldsymbol{t}}(\boldsymbol{q}) = [-\boldsymbol{w}\times]\left(\frac{d}{d\boldsymbol{q}}\overrightarrow{\boldsymbol{OG}_{tot}(\boldsymbol{q})}\right) + \sum_{i=1}^{f} [-\boldsymbol{f}_i\times]\left(\frac{d}{d\boldsymbol{q}}\overrightarrow{\boldsymbol{OE}_i(\boldsymbol{q})}\right)$$

Finally, by using the Jacobian matrices J_G and J_T derived in Section 4.5, it can be reduced to:

$$J_{\boldsymbol{t}}(\boldsymbol{q}) = [-\boldsymbol{w}\times]J_G(\boldsymbol{q}) + \sum_{i=1}^{f} [-\boldsymbol{f}_i\times]J_{T_i}(\boldsymbol{q}) \tag{6.6}$$

Of course, this result is equivalent to that of Aydin. However, we would like to point out a few differences. First, and contrary to the derivation method proposed by Aydin, the weight is not considered as any other external force applied at the center of gravity of each single rigid body, but it is treated as a single force acting at the center of mass $\boldsymbol{G}_{tot}$ of the figure. Hence, to deal with the weight, only one Jacobian must be computed instead of one for each rigid body constituting the figure. This certainly results in a faster evaluation of $J_{\boldsymbol{t}}$, since the number of exerted forces *f* is usually small with respect to the number of rigid bodies. Second, Eq. (6.6) is independent on the joint types, since the Jacobian matrices already integrate this aspect. Hence, no new derivation is required if new joint types are added to the system.

The range of the Jacobian matrix of each force component is equal to the set of vectors per-

pendicular to that force, and thus in a 3D environment its dimension is 2. Hence each force constraint leaves one degree of freedom to the torque function $\boldsymbol{t}(\boldsymbol{q})$. For example, if a vertical force is exerted (such as the weight), then the vertical component of $\boldsymbol{t}(\boldsymbol{q})$ is necessarily zero and is left unconstrained. However, in general, multiple linearly independent forces are present in Eq. (6.6): in that case the complete Jacobian $J_{\boldsymbol{t}}$ becomes full rank (in other words, all the components of $\boldsymbol{t}(\boldsymbol{q})$ need to be constrained).

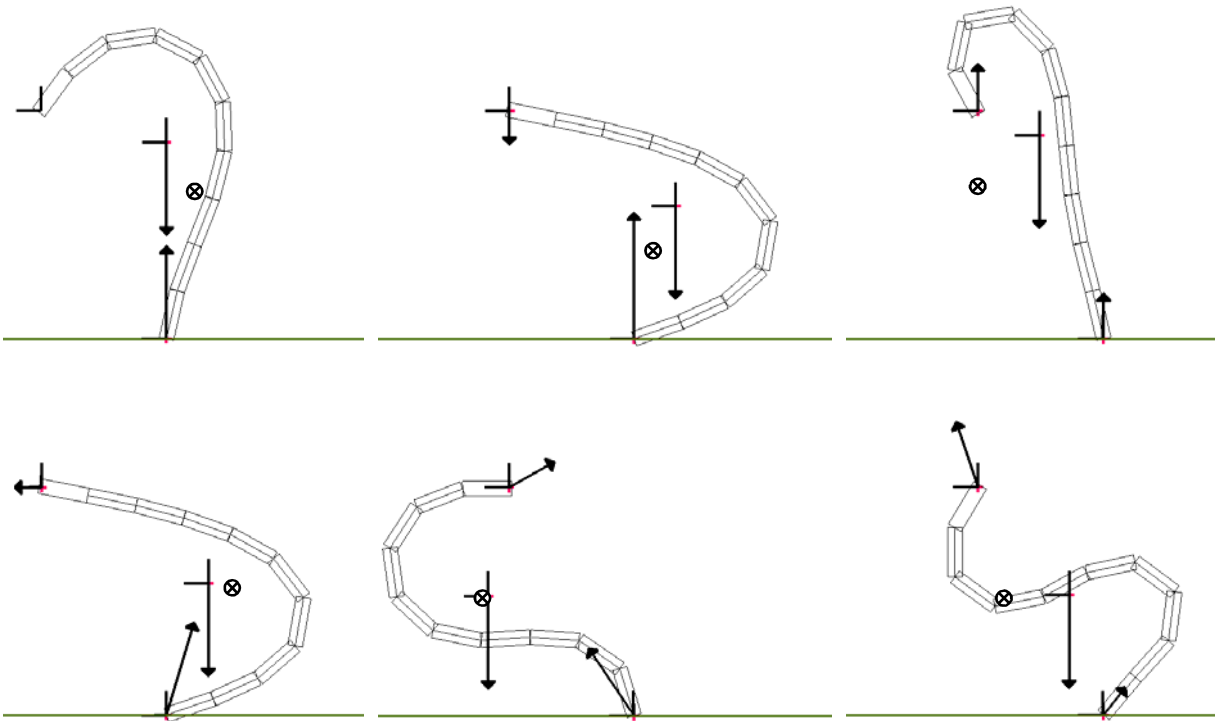


Figure 6.5 Simulation examples of configurations in static equilibrium, under the action of the weight and of two other forces (one for support applied at the base and another applied on the tip of the chain). The user changes the force exerted on the tip, and the system adapts the posture in order to satisfy the static equilibrium condition.

6.4 Computing the internal joint torques due to the external forces

The joint torques required to balance the forces perceived by the structure can be computed in several ways. For example, recursive methods propagate the forces and torques from the extremities to the root of the hierarchy [CRA 89]. Here instead, we make use of the Jacobian transpose relationship, which is well-known in robotics [CRA 89] [MUR 94]. An advantage of this method is that it is based on the same Jacobian matrices associated to the end-effectors for solving the inverse kinematics problem. Often, forces are applied at end-effectors whose location is constrained: hence the Jacobian matrix can be used for these two different purposes.

6.4.1 The use of Jacobian matrices in the static force domain

Let us define the following $6 \times n$ Jacobian matrix:

$$J_{E_i} := \begin{bmatrix} J_{T_i} \\ J_{R_i} \end{bmatrix}$$

where J_{T_i} and J_{R_i} are used to control the position and orientation of an end-effector E_i (see Section 4.5). Remarkably, J_{E_i} also linearly relates a static wrench $\widehat{\boldsymbol{f}_i}$ applied at the origin of the end-effector frame to the generalized joint torques $\boldsymbol{\tau}_{E_i}$ that must act to keep the system in static equilibrium:

$$\boldsymbol{\tau}_{E_i}(\boldsymbol{q}) = J_{E_i}^T(\boldsymbol{q})\,\widehat{\boldsymbol{f}_i} \tag{6.7}$$

This is known as the Jacobian transpose relationship, and can be proved from the equivalence of work performed by the wrench $\widehat{\boldsymbol{f}_i}$ and torque $\boldsymbol{\tau}_{E_i}$ over arbitrary infinitesimal displacements [MUR 94]. Interestingly, the singularities of the Jacobian matrix (discussed in Section 4.5) also affect the computation of the joint torques: in the singular directions, the torques required to exert a force are null, since the mechanical advantage integrally compensates that force. Moreover, parametric singularities (typically due to the choice of an Euler angles parametrization) may lead to wrong force considerations: at such a singularity, it may result from Eq. (6.7) that a force in a singular direction requires no joint torque to be exerted, while this is erroneous and merely results from the choice of parametrization.

The Jacobian transpose relationship can also be used to relate the weight of each single rigid body of the figure to the joint torques that balance that force. This requires to compute a Jacobian matrix for the center of mass of each rigid body, and finally to sum up their contributions. Fortunately, as shown in the next section, this can be expressed in a simpler way with the kinetic Jacobian matrix previously used for the control of the center of mass.

6.4.2 The kinetic Jacobian matrix in the force domain

The joint torques $\boldsymbol{\tau}_G$ required to resist to a static Cartesian force $\boldsymbol{w}$ exerted at the center of mass G_{tot} of the structure are

$$\boldsymbol{\tau}_G(\boldsymbol{q}) = J_G^T(\boldsymbol{q})\boldsymbol{w} \tag{6.8}$$

This result possesses the same form of Eq. (6.8) and can be proved by the same argument, as follows. The work due to the joint torques acting along an arbitrary infinitesimal displacement $d\boldsymbol{q}$ must equal the work due to the force $\boldsymbol{w}$ acting along an infinitesimal displacement $d\boldsymbol{x}_G$ of the center of mass:

$$d\boldsymbol{q}^T\boldsymbol{\tau}_G(\boldsymbol{q}) = d\boldsymbol{x}_G^T\boldsymbol{w} \tag{6.9}$$

Replacing Eq. (4.7) into Eq. (6.9) results in

$$d\boldsymbol{q}^T \boldsymbol{\tau}_G(\boldsymbol{q}) = d\boldsymbol{q}^T J_G^T(\boldsymbol{q}) \boldsymbol{w} \tag{6.10}$$

Finally, Eq. (6.8) results from the fact that Eq. (6.10) must hold for any $d\boldsymbol{q}$.

6.4.3 Computing the total joint torques

Thanks to the additive property of forces, their contributions can be summed up. Hence the total generalized torques due to all forces are

$$\boldsymbol{\tau}(\boldsymbol{q}) = \boldsymbol{\tau}_G(\boldsymbol{q}) + \sum_{i=1}^{f} \boldsymbol{\tau}_{E_i}(\boldsymbol{q}) \tag{6.11}$$

It is also important to note that, while the Jacobian matrices depend on the choice of a root in the figure, Eq. (6.11) does not depend on it, at least when the static equilibrium conditions are respected. This is not surprising, since the location of the root has no physical meaning in a situation with multiple supports (as in Fig. 6.7).

6.4.4 Visualizing the joint torques

Once the joint torques have been computed with Eq. (6.11), they can be visualized directly on the figure, to inform the user about the distribution of the load over the joints. The torque visualization may depend on the type of the joint. For a simple revolute joint (one degree of freedom), a torque can be visualized as an arc of a circle centered around the axis of rotation, and whose length is proportional to the amount of torque (see Fig. 6.6 and Fig. 6.7). This is certainly more intuitive than drawing the Cartesian torque vector itself. The arc of circle can be oriented to indicate the direction of the torque vector, but this is meaningful only in a single support situation. For multiple-DOF joint types, the question of torque visualization is more complex: while we did not implemented the following idea, it seems reasonable to decouple the torque into meaningful components, independently of the parametrization of the joint. For example, for the elbow or knee, the torque for the flexion/extension component can be decoupled from the twist component. Similarly, for a ball-and-socket joint, the torque component required to resist to a change in the swing component (direction) can be decoupled from the twist component. Each single component can then be visualized again with an arc of circle.

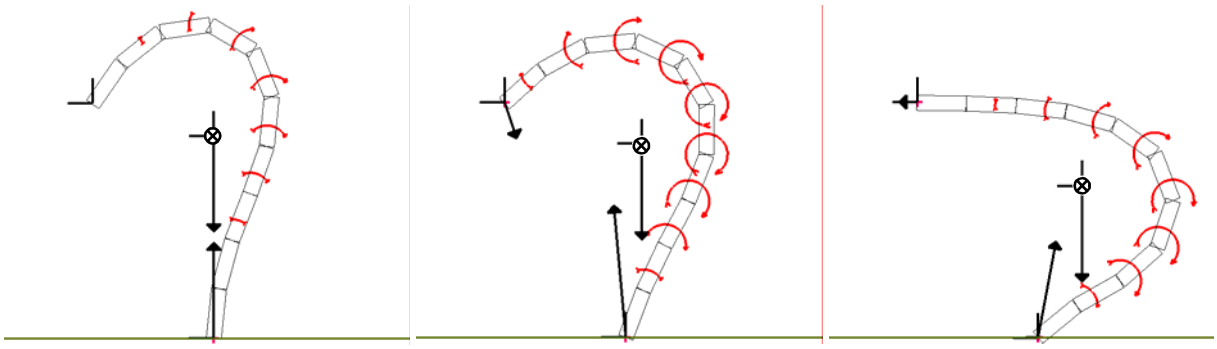


Figure 6.6 Visualization of the joint torques due to the Cartesian forces applied on a simple 2D chain with 10 revolute joints. Clearly, the posture on the left globally requires less joint torques than the two other postures, since no force is applied at the tip.

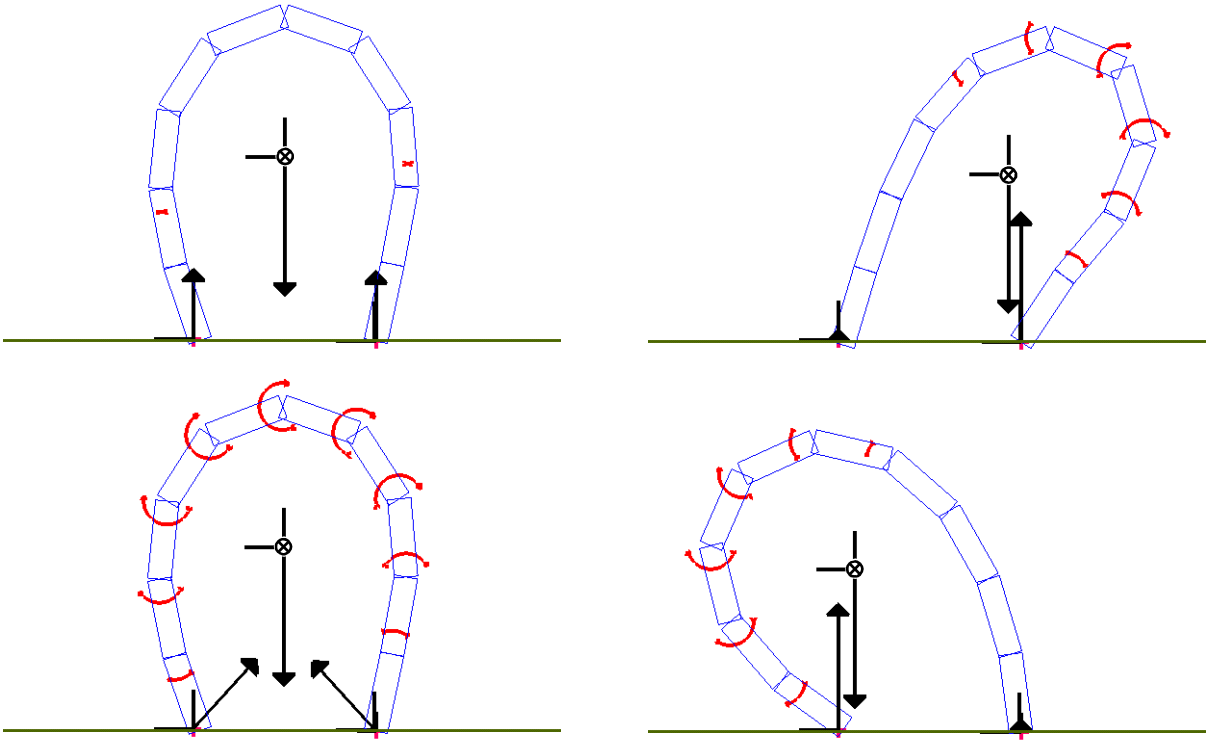


Figure 6.7 Different configurations of a simple chain in double support, resulting from different reaction forces specified by the user. The root of the hierarchy may be indifferently set on either supporting site, since the torque computation is not affected by this choice.

6.5 Minimization of joint torques

The torques exerted by the joints provide an information about the effort performed by the figure (i.e. by its muscles or motors). Of course it depends on the interaction forces, but also on the posture of the figure. Hence, while keeping the static equilibrium and other kinematic con-

straints satisfied, the posture can be adjusted to reduce the effort required to produce the forces. This should lead to more efficient, more comfortable, and hopefully more natural, postures.

For this purpose, a norm of the joint torques vector $\boldsymbol{\tau}(\boldsymbol{q})$ can be minimized:

$$h_{\boldsymbol{\tau}}(\boldsymbol{q}) = \|\boldsymbol{\tau}(\boldsymbol{q})\|^2 \tag{6.12}$$

A constrained minimization of $h_{\boldsymbol{\tau}}(\boldsymbol{q})$ leads to a posture allowing to exert the required forces with a minimum amount of joint torques. The gradient of this function, which is necessary for its optimization, is:

$$\nabla h_{\boldsymbol{\tau}}(\boldsymbol{q}) = 2J_{\tau}(\boldsymbol{q})^T\boldsymbol{\tau}(\boldsymbol{q}) \tag{6.13}$$

where $J_{\tau}(\boldsymbol{q}) := \frac{d\boldsymbol{\tau}(\boldsymbol{q})}{d\boldsymbol{q}}$ is the $n \times n$ Jacobian matrix of the function $\boldsymbol{\tau}(\boldsymbol{q})$. Its j$^{\text{th}}$ column is:

$$\frac{\partial\boldsymbol{\tau}(\boldsymbol{q})}{\partial\boldsymbol{q}_j} = \frac{\partial\boldsymbol{\tau}_G(\boldsymbol{q})}{\partial\boldsymbol{q}_j} + \sum_{i=1}^{f}\frac{\partial\boldsymbol{\tau}_{E_i}(\boldsymbol{q})}{\partial\boldsymbol{q}_j}$$

where

$$\frac{\partial\boldsymbol{\tau}_G(\boldsymbol{q})}{\partial\boldsymbol{q}_j} = \frac{\partial}{\partial\boldsymbol{q}_j}(J_G^T(\boldsymbol{q})\boldsymbol{w}) \tag{6.14}$$

and

$$\frac{\partial\boldsymbol{\tau}_{T_i}(\boldsymbol{q})}{\partial\boldsymbol{q}_j} = \frac{\partial}{\partial\boldsymbol{q}_j}(J_{E_i}^T(\boldsymbol{q})\widehat{\boldsymbol{f}_i}) \tag{6.15}$$

To develop further on, the type of the j$^{\text{th}}$ joint must be considered. For example, for a revolute joint, Eq. (6.15) can be developed as follows:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{q}_j}(J_{E_i}^T(\boldsymbol{q})\widehat{\boldsymbol{f}_i}) &= ([\boldsymbol{a}_j\times]J_{T_i}(\boldsymbol{q}))^T\boldsymbol{f}_i + ([\boldsymbol{a}_j\times]J_{R_i}(\boldsymbol{q}))^T\boldsymbol{t}_i \\ &= J_{E_i}^T(\boldsymbol{q})\begin{pmatrix}\boldsymbol{f}_i\times\boldsymbol{a}_j\\ \boldsymbol{t}_i\times\boldsymbol{a}_j\end{pmatrix}\end{aligned}$$

A similar result can be obtained for Eq. (6.14):

$$\frac{\partial}{\partial\boldsymbol{q}_j}(J_G^T(\boldsymbol{q})\boldsymbol{w}) = J_G^T(\boldsymbol{q})(\boldsymbol{w}\times\boldsymbol{a}_j)$$

6.5.1 Examples and analysis

Two simple examples with joint torque minimization are given in Fig. 6.8 and Fig. 6.9, on a simple chain. Two tasks with high priority are created to control the position of the tip and to ensure the balance condition, while the torque minimization is performed with a lower priority. Clearly, the links tend to align themselves in the direction of force exertion, since this behav-

iour reduces the joint torques. The erect posture of human beings is an example of posture aligned with the weight force, to minimize the gravity torques.

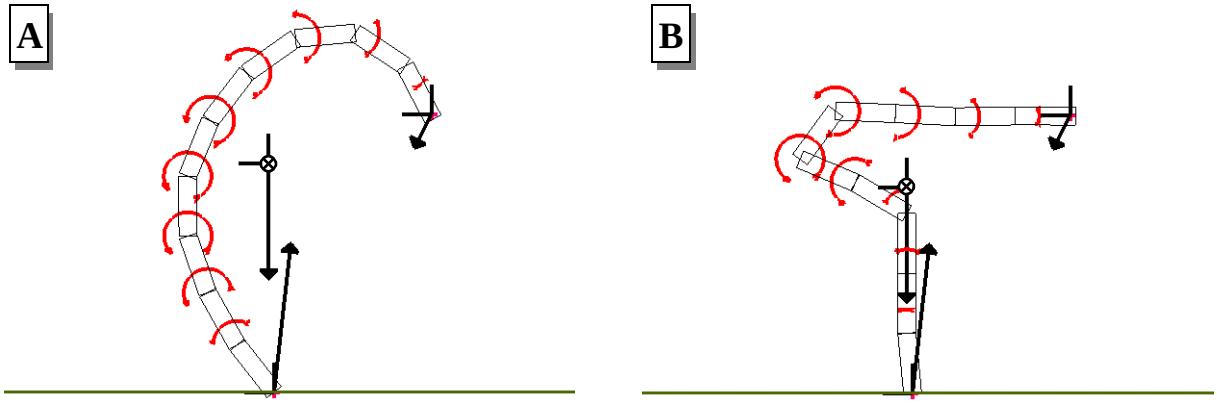


Figure 6.8 Illustration of the minimization of joint torques, under the static equilibrium constraint and a tip position constraint. Posture $\boldsymbol{q}_B$ (on the right) is obtained by minimization of $h_{\boldsymbol{\tau}}(\boldsymbol{q})$, and thus is more efficient than posture $\boldsymbol{q}_A$ ($\|\boldsymbol{\tau}(\boldsymbol{q}_B)\| / \|\boldsymbol{\tau}(\boldsymbol{q}_A)\| = 0.72$).

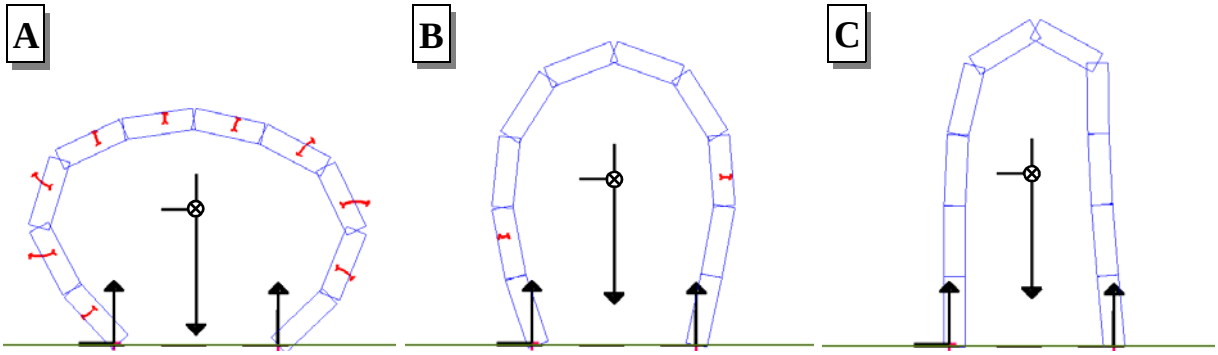


Figure 6.9 Three different configurations ($\boldsymbol{q}_A$, $\boldsymbol{q}_B$, $\boldsymbol{q}_C$) of a chain in double support. Posture $\boldsymbol{q}_C$ (on the right) is obtained by minimization of $h_{\boldsymbol{\tau}}(\boldsymbol{q})$, and thus is the most efficient ($\|\boldsymbol{\tau}(\boldsymbol{q}_C)\| / \|\boldsymbol{\tau}(\boldsymbol{q}_A)\| = 0.65$ while $\|\boldsymbol{\tau}(\boldsymbol{q}_B)\| / \|\boldsymbol{\tau}(\boldsymbol{q}_A)\| = 0.81$).

6.5.2 Biomechanical human joint strength models

The criterion $h_{\boldsymbol{\tau}}(\boldsymbol{q})$ does not take into consideration possible differences between joint capabilities to exert torques. For example, the human wrist can certainly exert less torque than the elbow or the shoulder. In the ergonomics literature, the maximum achievable joint torque is called the *strength*, and it is a valuable information to integrate in a posture optimization process. A number of biomechanical studies (such as [AYO 81]) has been performed to measure strength data, and strength models have been developed [SCH 72]. Some computer simulations, such as those of Garg and Chaffin for ergonomics evaluation [GAR 75] [CHA 99] or Lee *et al.* for motion simulation [LEE 90], rely on such models. This strength information is however very complex to measure and to model, since it is highly variable from a person to

another, and depends on many parameters (such as the direction of exertion, the muscles involved in the exertion, the position and velocity of the joint and of the adjacent joint, the fatigue of the person, and so on). This is due to the high complexity of the musculo-skeletal system, and is a major obstacle to the use of strength models in full body posture optimization, especially for ergonomics applications that require a high degree of accuracy, since the simulations are done for prediction purposes.

For the posture manipulation task with no ergonomic evaluation goal, a practical compromise solution is to weight the joint torques in the criterion function $h_{\boldsymbol{\tau}}(\boldsymbol{q})$, and to determine the weights in a qualitative way in order to respect at least the orders of magnitude. For example, the wrist roughly has ten times less strength than the shoulder.

6.5.3 Limitation of the optimization method

Minimizing the joint torques $\|\boldsymbol{\tau}(\boldsymbol{q})\|$, even with a strength model, still does not guarantees that the resulting “optimal” posture is feasible in reality. The posture is simply optimal for the given set of forces. However, if the forces exceed the capability of the structure, the resulting posture (and actually any other posture) won’t be comfortable or even feasible in reality.

This is particularly apparent at configurations where the Jacobian matrix associated to a force exertion is singular: at such points, an arbitrarily large force can be supported in the singular direction: no joint torques are induced since the force is integrally supported by the component of the structure which is not parametrized by $\boldsymbol{q}$. Hence, internal forces such as compression or tension in the back are not considered. This may provide unrealistic or unfeasible postures if too high forces are exerted on it, and this is of course inacceptable for an ergonomic evaluation tool, but is enough for a posture manipulation tool.

6.6 Conclusion

Posture control via force specification and torque control nicely fits within the inverse kinematics framework described in the previous chapters. The main reason is that the Jacobian matrices used to ensure kinematic constraints can be exploited to control the torque acting on the figure (hence ensuring the static equilibrium condition) and also to compute the generalized joint torques required to balance those forces (with the Jacobian transpose relationship). This is amplified by the fact that, as we have shown, the kinetic Jacobian matrix used to control the center of mass, also allows to deal with the weight force in a particularly simple and elegant way.

In the next chapter, more elaborate examples are shown, that result from the combination of force control with kinematic control of the end-effectors of the figure.

Chapter 7

An application for the manipulation of articulated figures

7.1 Introduction

An important application of inverse kinematics is the manipulation of articulated figures for posture design. In this chapter, we briefly present an application, called *BALANCE*, based on the techniques described before. It allows us to illustrate the usefulness of task priorities, combined with balance control.

7.2 Presentation of the BALANCE application

The *BALANCE* application deals with general open-chain structures with joint limits (see Chapter 3), and is based on the framework of the task-priority algorithm (see Chapter 5). Hence, a figure can be manipulated by a set of tasks with variable priorities and weights. Forces can also be applied at end-effectors, to control the balance of the figure (see Chapter 6). Finally, a set of optimization criteria are proposed in order to select a posture, when multiple solutions exist.

7.2.1 Overview of the application

The application is centered around a window displaying the scene. Additional windows can be created to look at the scene from different viewpoints: this is useful to understand how the posture takes place in the 3D space (see Fig. 7.1). Of course each viewpoint can be modified. In the main window, a “STATUS” field informs about the satisfaction of the tasks, with a color code (*green* if all tasks are satisfied, *orange* if at least the top priority task is satisfied, and *red* if no task is satisfied).

Body models can be loaded from description files, and edited directly in the application: the topology, size of the segments, and joint models can be edited and then saved for future use.

Two panels are available for posture control:

- A panel for the management of tasks (in Cartesian space)
- A panel for the selection of optimization criteria (in joint space)

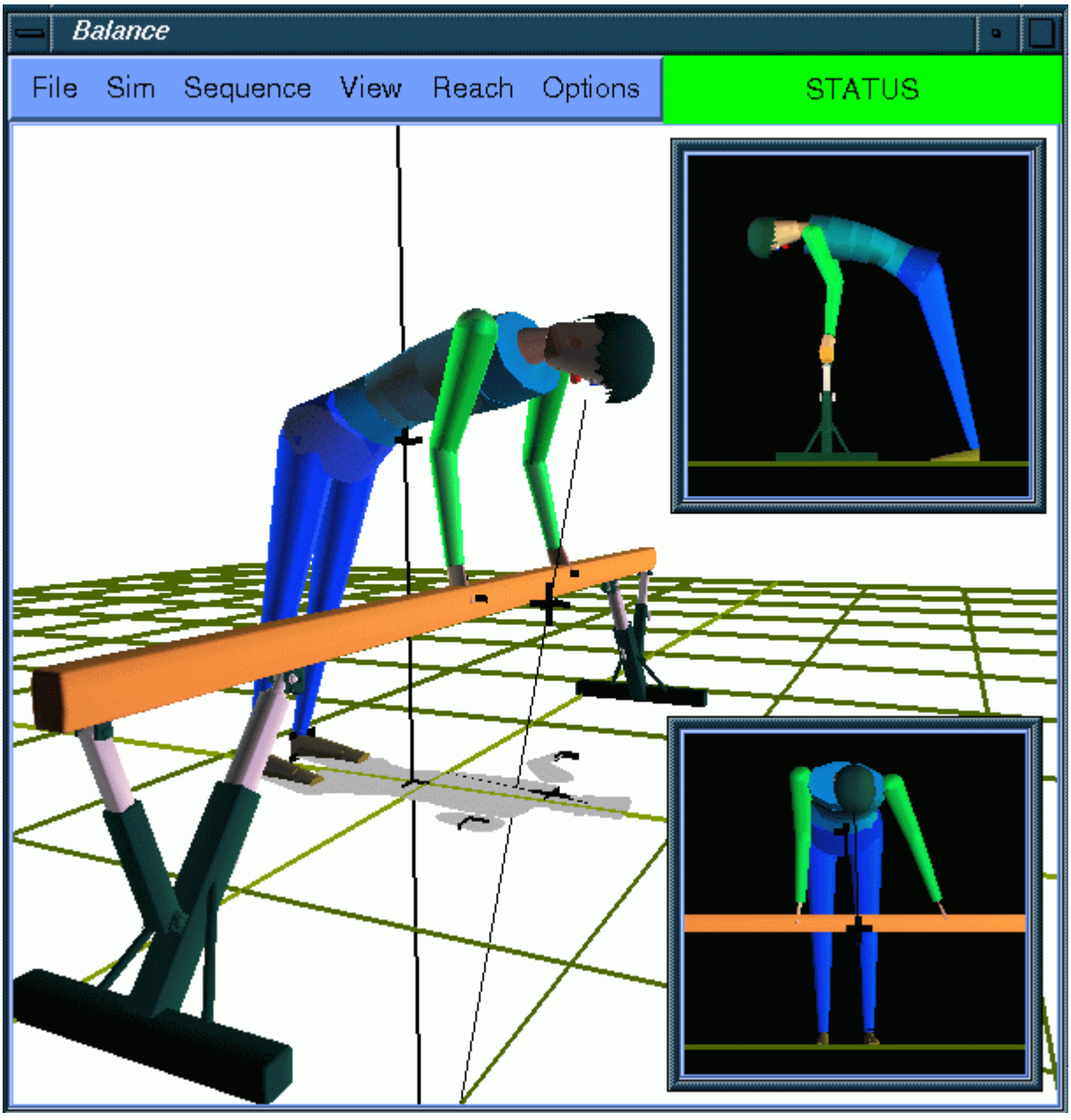


Figure 7.1 Main window of the BALANCE application. Two additional small windows provide different viewpoints, to better appreciate the spatial configuration of the figure (the shadow of the beam is not shown).

7.2.2 Task specification

The panel devoted to the management of tasks is shown in Fig. 7.2. The list of currently active tasks is given, with a name to identify each task and its type. When a new task is created, its type must be specified. Typical task types allow to control

- the position (-P) or orientation (-O) of an end-effector,
- the position of the center of mass of the figure,
- the relative position or orientation of two end-effectors,
- the value of the torque function (for equilibrium control).

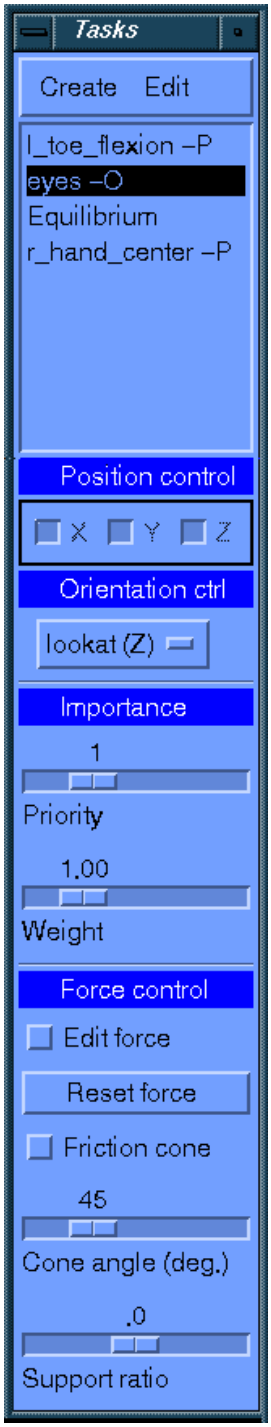


Figure 7.2 Panel for task specification.

There are two ways to specify an end-effector, which is represented by a frame rigidly attached to a node of the hierarchy. First, the end-effector may be selected among a list of predefined frames (or *sites*) available in a separate window. This is a simple and accurate way to select standard end-effectors such as the center of the hands, or the vision frame in order to control the gazing direction of the figure. If the frame is not predefined, an alternative and more direct solution is to pick a point on the surface of a body part with the mouse, in the scene window. The point is simply computed by casting a ray from the camera in a direction given by the mouse pointer, and then intersecting it with the geometric primitives that constitute the figure. This allows to control any visible part of the figure just by clicking on it.

The parameters of the currently selected task are shown in the remaining part of the panel, and can be modified at any time during the interaction without having to stop the convergence process. Some of the parameters depend on the task type, while other are general. For a position task, it is possible to select which of the X/Y/Z dimensions must be effectively controlled, in order to specify point-on-line or point-on-plane constraints. For an orientation task, several modes are available to constrain the whole or part of it: for example, with a *look-at* constraint, only the direction of an axis of the end-effector frame is controlled. The relative weight and priority of the task can also be specified in this panel: the weight is only used when other tasks lie at the same priority level. Finally, for an end-effector position task, a sub-panel is available for the specification of the force exerted by the environment upon that end-effector. The support ratio (i.e. the relative amount of supported body weight) can be controlled. To constrain the force, a friction cone can also be specified, if necessary. The control of torques applied on an end-effector has not been implemented.

The goal for the task and the optional force vector are not specified in this panel but directly in the window displaying the scene, using standard input devices (mouse and keyboard).

7.2.3 Restricting the set of joints for the satisfaction of a given task

By default, all joints that potentially affect the satisfaction of a task are controlled. This is not always desirable, since this may unnecessarily modify the current posture. Instead, a subset of these joints can be used. This reduces the chances of the task of being satisfied, but may result in potentially useful behaviours. This is especially important for designers, that do not want to see their carefully designed posture to be unnecessarily modified by the IK engine.

For example, a task that controls the position of a hand affects the joints lying between the hand and the root of the hierarchy: this includes the arm and the spine. However, modifying all these joints results in unrealistic final postures. If the goal is easily reachable, the joints of the spine should not be recruited by the task.

For the control of the center of mass, it is interesting to restrict the set of joints to those that really have an impact on the center of mass. A side effect of considering all joints is that body parts that have little or no impact are still adjusted. On a human articulated figure, manipulating the neck and wrist joints for the purpose of balance control is not a good idea, since the orientation of the head and hands is dominated by more important tasks such as vision and grasping. This goes against the reasonable principle that changes in joint configuration should only happen when they are truly necessary.

More generally, the user should be able to select the set of joints that participate to the satisfaction of a task, for example by overriding the default choice which is to use all joints. Welman [WEL 93] discusses several modes for selecting the set of active joints: for example, the set of joints may be restricted to the nearest branch where the end-effector lies. In the case of the control of the hand of a human figure, only the joints of the arm would be controlled, and not those of the spine.

7.2.4 Temporary and automatic tasks

Besides permanent tasks that appear in the *Tasks* panel, other tasks may be used in the resolution process.

First, a temporary, anonymous position task is created simply by picking on a body part, then moving its goal and finally releasing the task, all with a single mouse click. This “drag-and-drop” possibility proves to be very useful to perform small adjustments of the posture. Since it does not appears in the *Tasks* panel, its parameters cannot be modified, but appropriate default values are used. For example, the priority is set to the lowest possible level, so that the current tasks are not affected.

Second, to ensure collision avoidance with surrounding objects, the program automatically manages a set of top priority tasks to repel the possibly colliding body parts outside the objects.

Similarly, self-collision avoidance could be ensured with a set of tasks controlling the relative position of pairs of body parts colliding with each other. This is a *collision response* approach comparable to that adopted in [ZHA 94b]. Self-collision avoidance has not been implemented in our system.

7.2.5 Selecting optimization criteria

A set of criteria expressed at the joint level is available in a separate panel. Note that some criteria are general and can be applied to any articulated structure, while others are specific to the human body. The following criteria have proved to be particularly useful:

- *distance to initial posture*: each joint angle is kept as close as possible to its initial value;
- *magnitude of joint torques*: the torques due to the weight and to the exerted forces are minimized (see Section 6.5);
- *amount of flexion* / *extension*: the joints whose axes are perpendicular to the sagittal plane are rotated in order to flex or extend the whole body (especially the hip, knee and ankle joints).

Any such criterion can be activated or deactivated at any time during the interactive session. When multiple criteria are active simultaneously, their gradients are simply summed together to form the final gradient $\nabla h(\boldsymbol{q})$. By construction, these criteria are always minimized with the lowest priority and hence do not interfere with the other tasks.

7.2.6 Improving the user interface

Certainly, the user interface of the application could be further improved. For example, in the *Tasks* panel, the priority of the currently selected task is displayed and modified through a slider widget. However, there is no global view of how the tasks are related to each other: the user must browse through the entire list of tasks to have the global picture. The priority order should be made more explicit: for example, the tasks could be sorted by order of priority, and indented proportionally to their level of priority.

7.3 Simulation examples

Now a number of examples illustrate the task-priority mechanism and the balance task.

7.3.1 The use of top priority tasks to ensure anatomical or structural constraints

This example shows how an anatomical constraint may be ensured with higher priority than a functional constraint. A kinematic model of the shoulder has been developed by Maurel *et al.* [MAU 00]: as shown in Fig. 7.3-A, the scapula is constrained to slide on the thorax modelled as an ellipsoid, in order to prevent unrealistic positioning of the bones (Fig. 7.3-B). Hence, a point of the hierarchy is constrained to slide on the ellipsoid. Of course, such an anatomical constraint must be ensured with the highest priority, with respect to other tasks like hand position control. Doing this with a weighting technique would not be satisfying, since the anatomi-

cal constraint could be violated.

Another example of anatomical constraint is given in Fig. 7.4 by the coupling of the knee and hip joints discussed in Section 3.8.2. This time, it is a linear inequality constraint that must be ensured.

The last example concerns the satisfaction of loop constraints. Loops are structural constraints for parallel robot manipulators: as shown in Fig. 7.5, a loop constraint can be ensured with the highest priority while other tasks modify the posture with a lower priority.

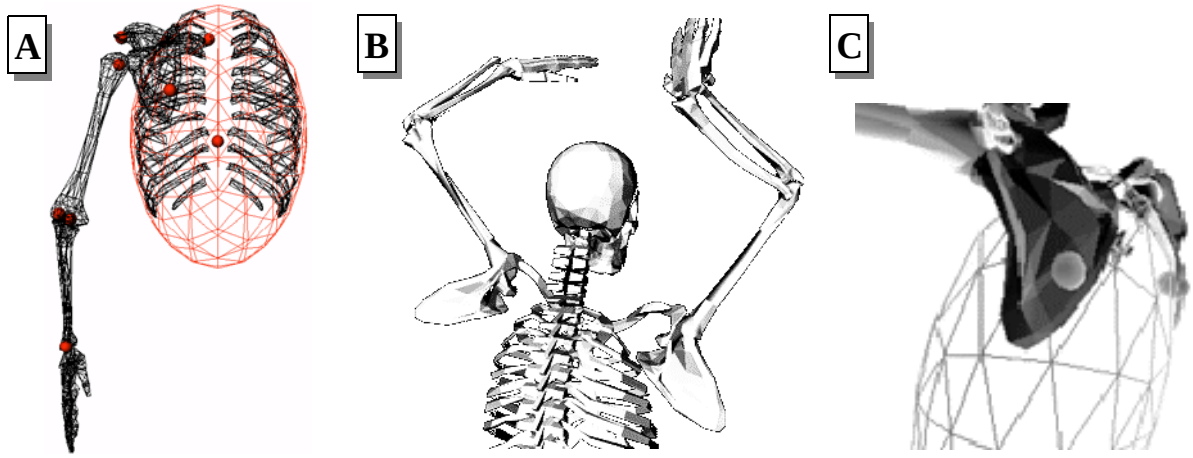


Figure 7.3 The Scapulo-Thoracic (ST) constraint. Wireframe rendering of the thorax modeled as an ellipsoid, and of the bones of the right arm (**A**), and polygonal rendering of the skeleton showing the consequence of ignoring the ST constraint (**B**). Close-up on the ST constraint (**C**). Figures are due to W. Maurel.

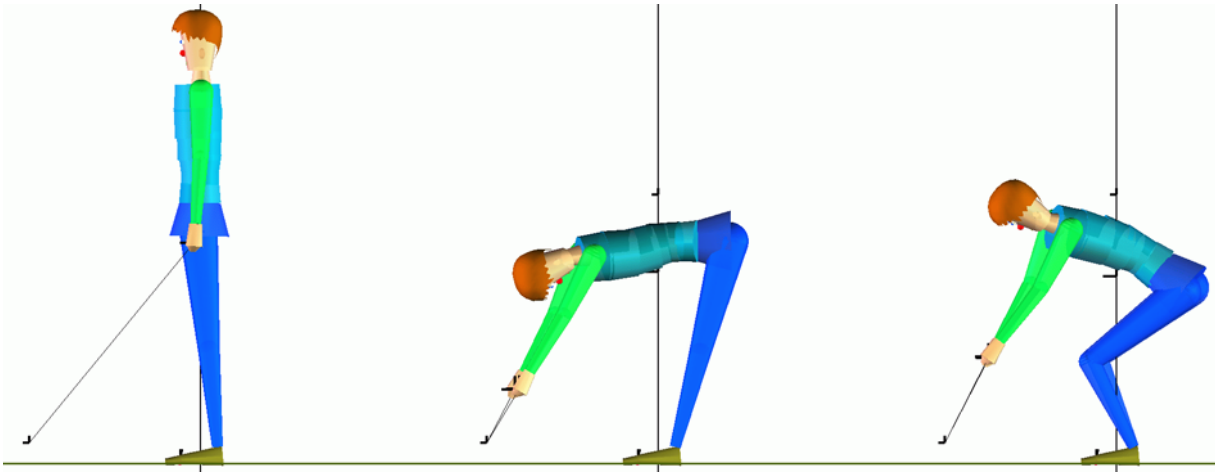


Figure 7.4 An illustration of the coupling between the hip and knee joints. A reaching task is specified (*left*), and the resulting posture is somewhat uncomfortable (*center*). On the right, the coupling constraint is activated: as a consequence, the knees are flexed to satisfy the constraint.

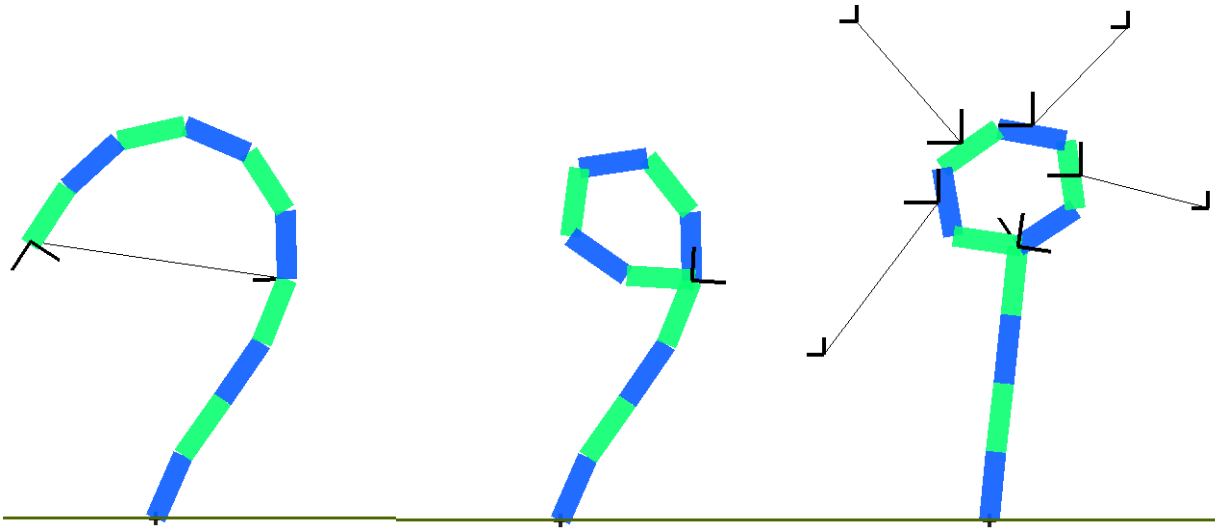


Figure 7.5 A loop constraint is applied to a simple chain. It is given the highest priority so that other end-effector tasks (shown on the right figure) do not break the loop.

7.3.2 Collision detection with an external obstacle

This example takes place in a plane, and illustrates the collision detection mechanism that creates a set of high priority tasks to repel any part of the chain lying inside the fixed obstacle. At the same time, the tip of the chain is controlled with a task of lower priority. In Fig. 7.6-A, a goal is specified on the tip, and it collides with the obstacle (Fig. 7.6-B): hence, a new task is automatically created to move the tip out of the obstacle (the plane tangent to the obstacle is shown). The tip slides on the surface of the obstacle, and finally reaches its goal (Fig. 7.6-C). In the last figures, the goal is moved around the sphere, in order to tighten the chain up to its limit.

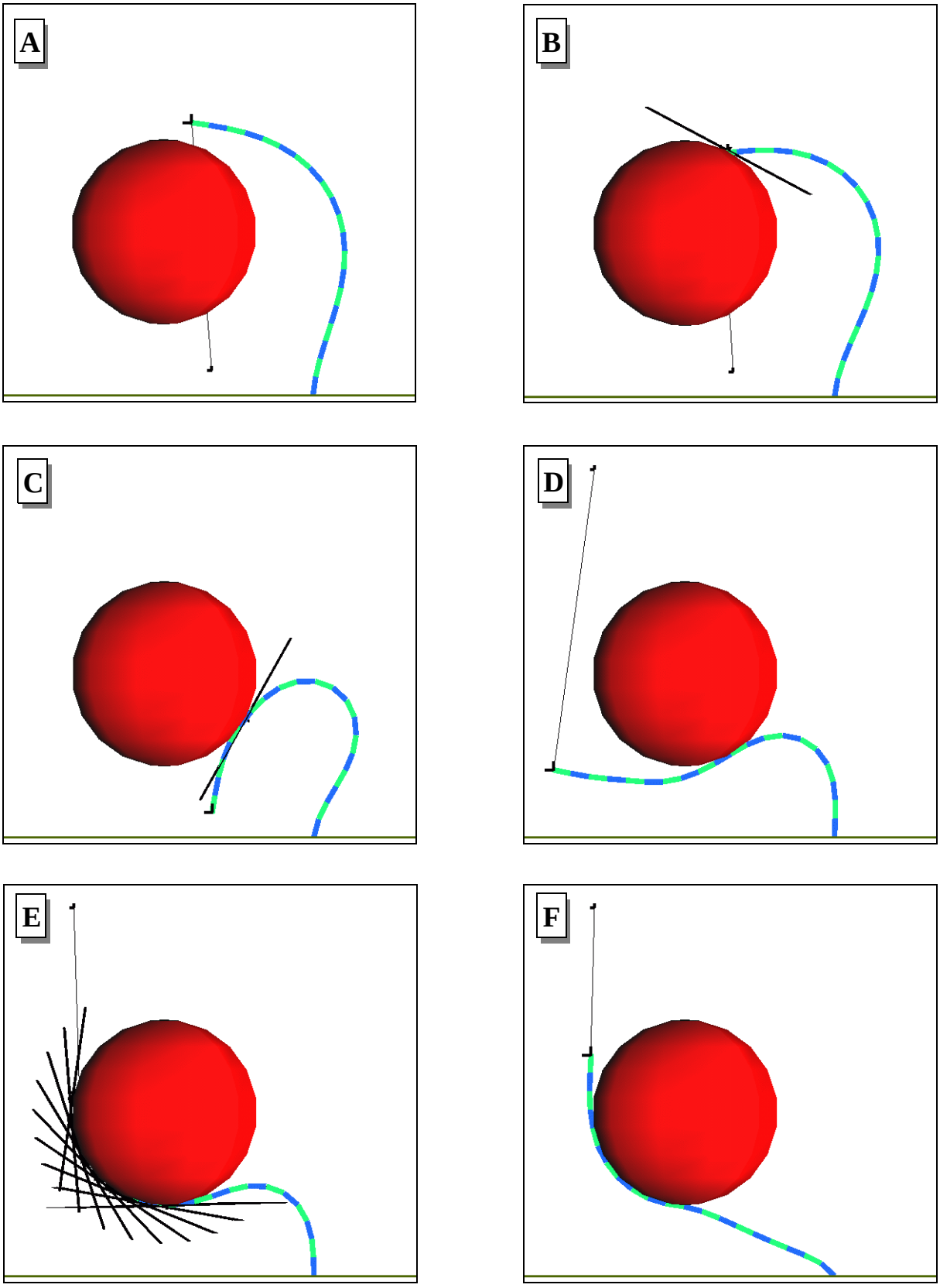


Figure 7.6 Collision detection and response of a simple chain with a fixed sphere.

7.3.3 Control of the moment of inertia

In [BAE 00] we have extended the control of mass properties of an articulated body to its moment of inertia about an axis passing through the center of mass, for the control of the mass distribution about that axis. A moment of inertia is graphically represented by a thin ring centered at the center of mass, lying in a plane orthogonal to the axis, and whose radius is the radius of gyration $r = \sqrt{inertia/m_{tot}}$. We now give two simple examples that use this kind of task, together with other tasks.

As a first example, consider a 9-DOF chain lying in a plane, and whose links all have identical mass properties. Its initial configuration and its center of mass are shown in Fig. 7.7-A. Two constraint tasks are first assigned to the chain: a *loop* task, to bind its tip and base together, and a *center of mass* position task. A configuration satisfying both tasks is shown in Fig. 7.7-B. Then, a *moment of inertia* task is added, with lower priority that the others. In Fig. 7.7-B, the current moment of inertia is visualized as a thin ring, while the desired moment of inertia is visualized as a thick ring (both are centered at the center of mass). In Fig. 7.7-C, a solution to this updated problem is shown. Both rings do not coincide because the desired moment of inertia is unreachable under the given constraint tasks. Now, if the center of mass is left free to move in the vertical direction (i.e. one degree of freedom is recovered), the chain adopts a configuration with a higher moment of inertia (Fig. 7.7-D): it is “stretched” up to its maximum, without violating the two constraint tasks.

As a second example, we control the center of mass and moment of inertia of a simple 2D human body model (with only 12 DOFs and joint limits). The rigid parts of the body are represented by simple geometric shapes, and their mass is proportional to their volume (i.e. the density is constant). To ensure static balance, the vertical projection of the center of mass on the floor is kept fixed at the center of the area of support. For this purpose, a task with high priority is assigned to the center of mass, but its vertical component is left unconstrained. In Fig. 7.8-A, the initial posture is shown, along with a ring representing the moment of inertia about a lateral axis passing through the center of mass. This central moment of inertia is controlled by a second task of lower priority. Its goal, shown as a thick ring in Fig. 7.8-B, is made exaggeratedly big, so that the posture with greatest moment of inertia is found. As it can be expected, the solution is an elongated posture (Fig. 7.8-B). From this configuration whose moment of inertia is maximal, a sequence of postures with decreasing inertia is shown in Fig. 7.8-C, D and E, until the body is completely curled up (Fig. 7.8-F).

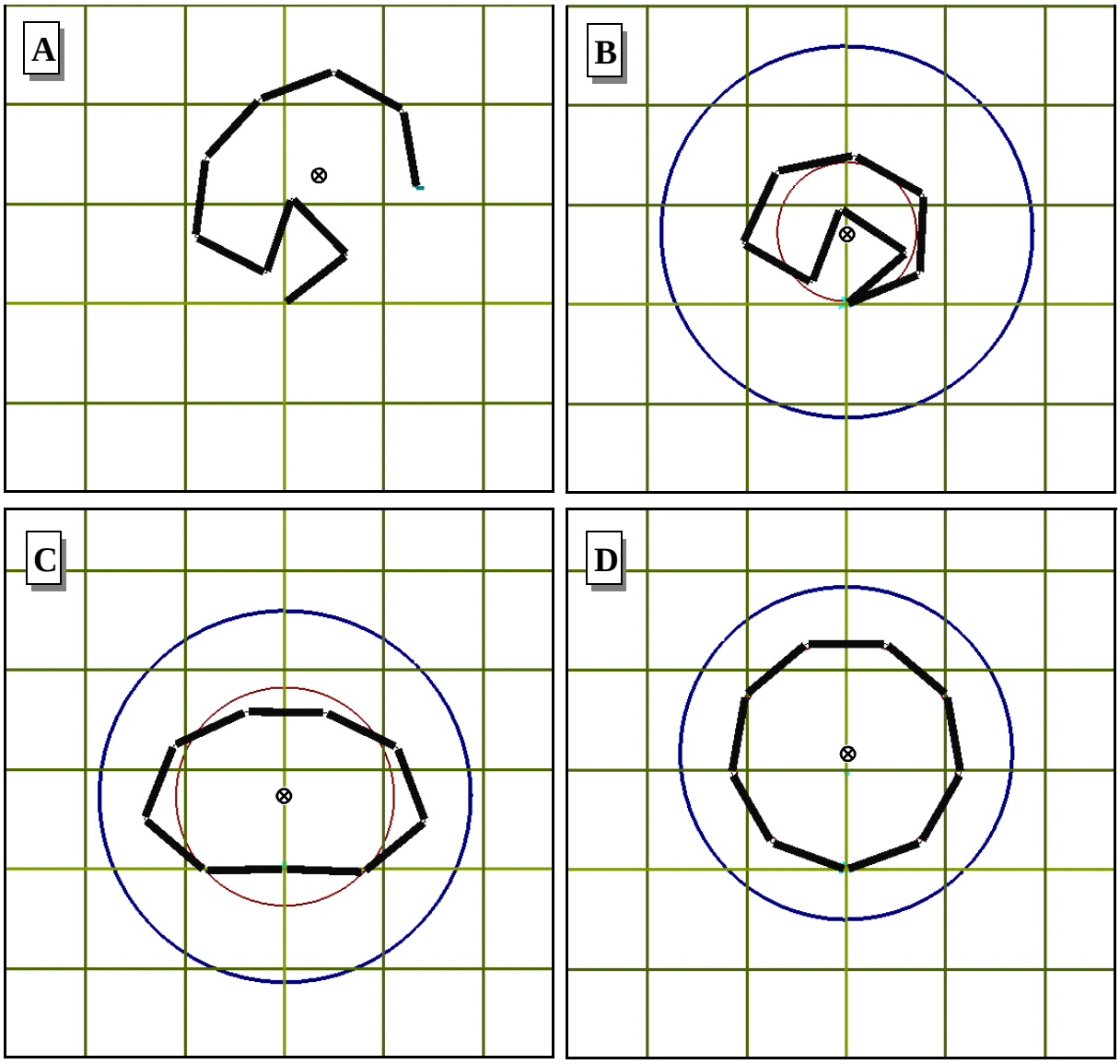


Figure 7.7 The moment of inertia about the center of mass (⊗) is maximized under a set of constraint tasks (see text).

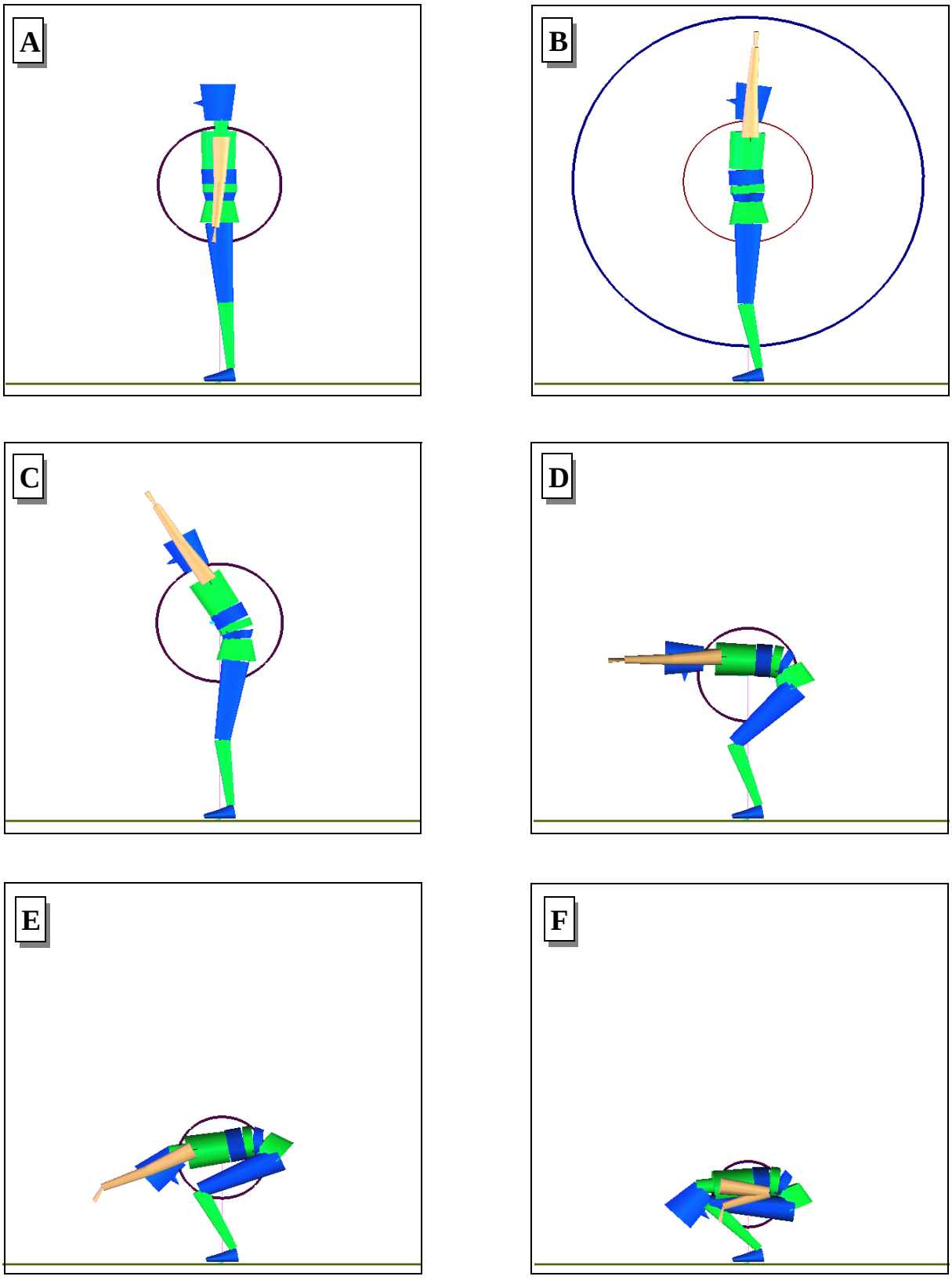


Figure 7.8 Lateral view of a simple 2D human figure trying to match prescribed moments of inertia while keeping balance.

7.3.4 Accessibility study: interactive evaluation of the reachable space

The use of different priority levels allows to evaluate the reachable space of an end-effector under a set of constraints. We interactively explore the reachable space by moving the end-effector in all directions and leaving a trace on its path. In Fig. 7.9, this procedure is applied to the tip of a simple chain, while in Fig. 7.10 it is applied to the hands of a human figure. Both figures are constrained to keep balance, with a high priority task assigned to the center of mass. The hands of the human figure are also required to stay together, with a high priority loop task.

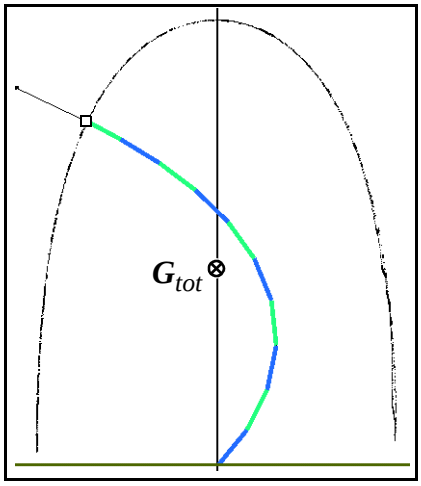


Figure 7.9 Boundary of the reachable space of the tip of a 10 DOF chain. Its position is controlled by a task of low priority. To keep the chain in static equilibrium, its center of mass is constrained to lie on the vertical line by a task of higher priority.

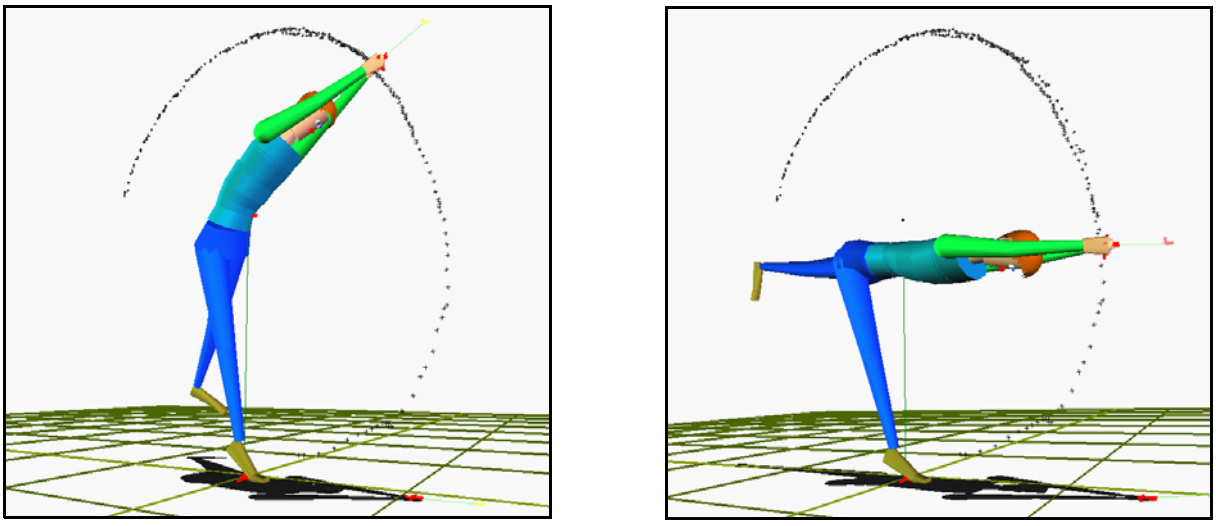


Figure 7.10 Accessibility study of the hands, constrained to stay together. Only the result in the sagittal plane is shown.

7.3.5 Looking at a fly while standing on a beam

A sequence of postures of a figure standing on a beam is shown in Fig. 7.11 and Fig. 7.12. The root is set on one foot, while the other foot is fixed both in position and in orientation by a high priority task. The center of mass is also constrained on a vertical line passing inbetween the feet, for balance control. In **B** and **C**, the legs are flexed with an optimization criterion, hence without affecting the tasks. In **D**, a new task with low priority is created to control the gazing direction specified by a cross. The figure adapts its posture to follow up the cross, while it is moved around. In **F**, this is not possible and thus the figure adopts the closest possible posture. In **I**, another task is created on the right hand to try to reach the cross: a low priority is set also for this task. Hence the hand only points towards the cross, as if it was trying to reach it. In **J**, the body weight is entirely shifted on the left foot, to obtain the posture that results in the shortest distance of the cross for the hand. This posture is of course highly unstable.

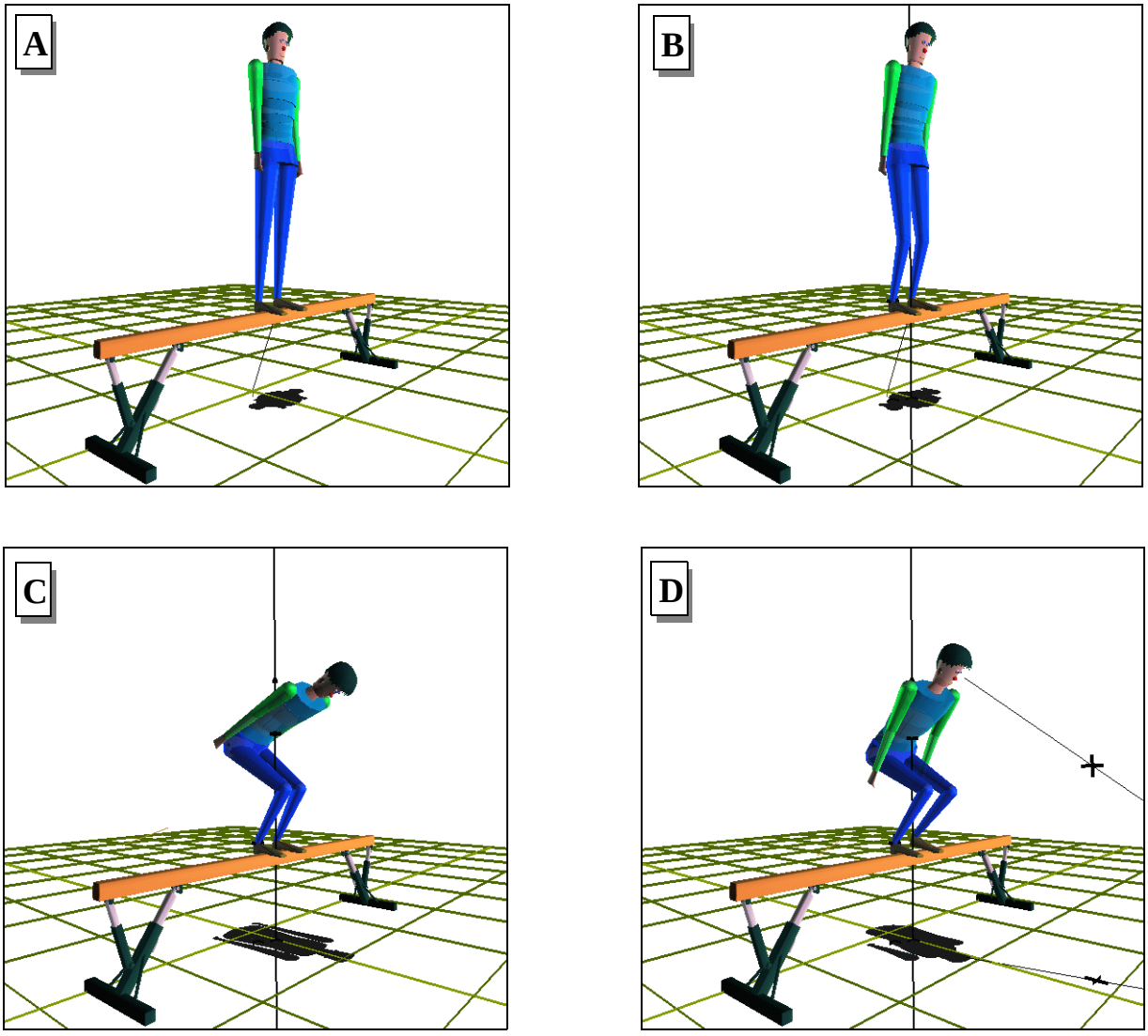


Figure 7.11 Interactive manipulation of a human figure standing on a beam.

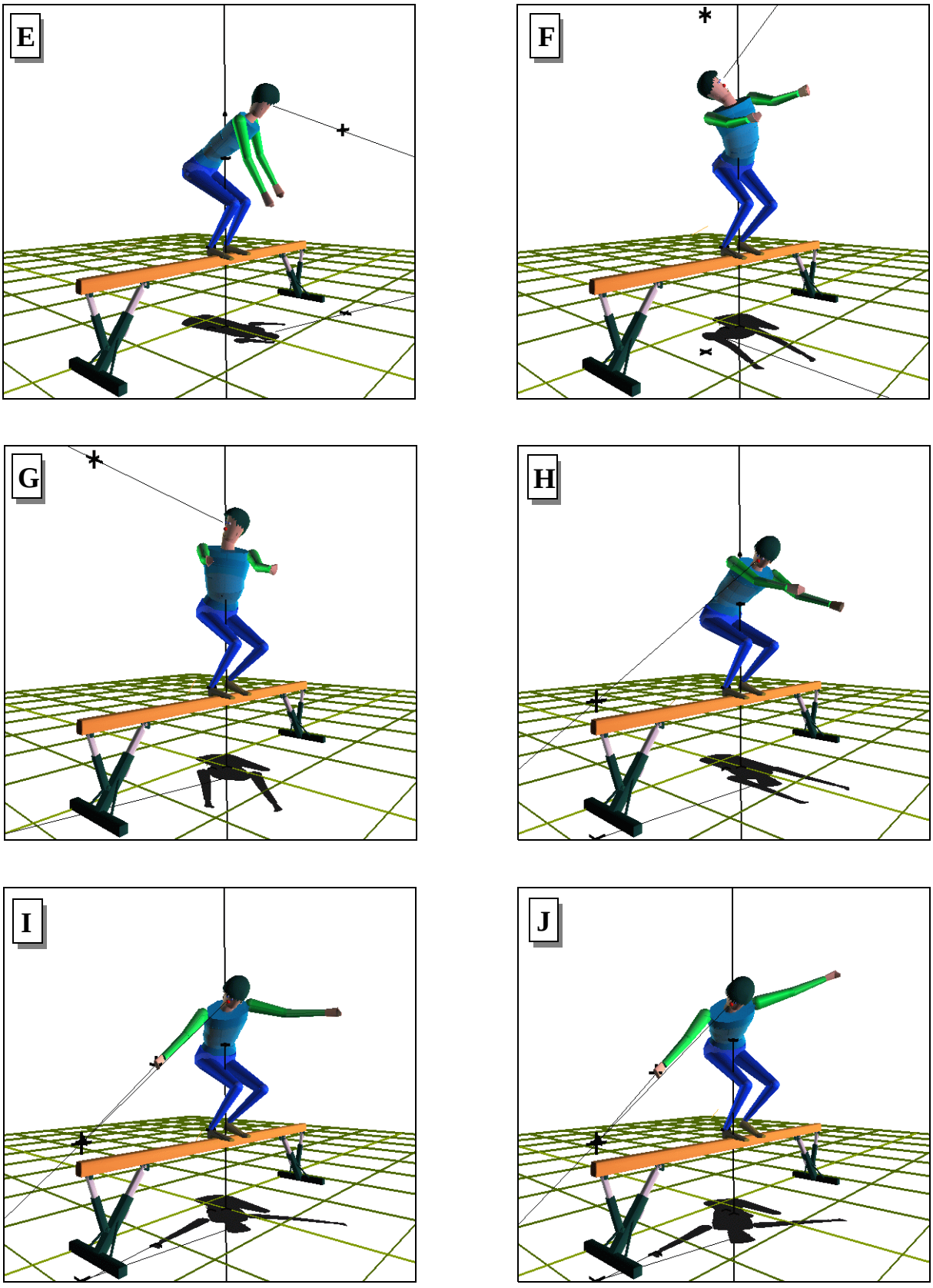


Figure 7.12 Interactive manipulation of a human figure standing on a beam (cont.).

7.3.6 A full hierarchy of tasks to control a human figure

The sequence of Fig. 7.13 illustrates the priority mechanism with a large number of levels. In the first image, the initial posture of the figure is shown, along with a ball and a cylinder (which is an obstacle). It is required that both feet stay firmly planted on the floor. For this purpose, the figure hierarchy is rooted on one foot, while the position and orientation of the other foot is constrained with two tasks. The user wants the figure to look at the ball, to reach it with the right hand, and to keep static balance as well. According to the positions of the ball and of the cylinder, modifiable at any time, this set of tasks may give rise to conflicts. To manage this possibility, the user establishes the order of priority given in Table 7.1. 1n the second image, a posture satisfying all tasks is shown. Then, the user moves the ball away from the figure. On the third image, a conflict between the balance task and the reach task appears: the figure can no longer reach the ball without losing balance, despite an attempt to move back the body mass. Of course, the balance is ensured thanks to its higher priority. To conclude, the user moves the cylinder towards the figure: the waist moves back, in order to avoid the collision, up to a point where the balance task becomes incompatible with the collision avoidance task and, as shown in the fourth image, cannot be satisfied anymore. On the other hand, the foot constraint is still achieved, because it is not in conflict with the collision avoidance task and because it has priority over the balance task. If the balance task had priority over the foot constraint, the figure would first have tried to keep balance by moving the leg forward, hence sacrificing the foot constraint.

Priority level	Task specification (and its *type*)	Valency
1 (top priority)	Handle collisions with the cylinder (*position*)	variable
2	Keep foot on the floor (*position* + *orientation*)	6 DOF
3	Balance control (*center of mass position*)	2 DOF
4	Look at the ball (*orientation*)	2 DOF
5 (lowest priority)	Reach the ball with the right hand (*position*)	3 DOF

Table 7.1 The order of priority established for the sequence shown in Fig. 7.13.

7.3.7 Application of optimization criteria

Another example (Fig. 7.14) illustrates the activation of the *flexion* criterion, to flex the body without interfering with a set of constraint tasks, which are to keep balance, to keep both feet fixed on the floor, to hold object with both hands and to look at it. Hence, the figure flexes its legs without losing balance, and the neck is automatically adjusted in order to keep the eyes on the object.

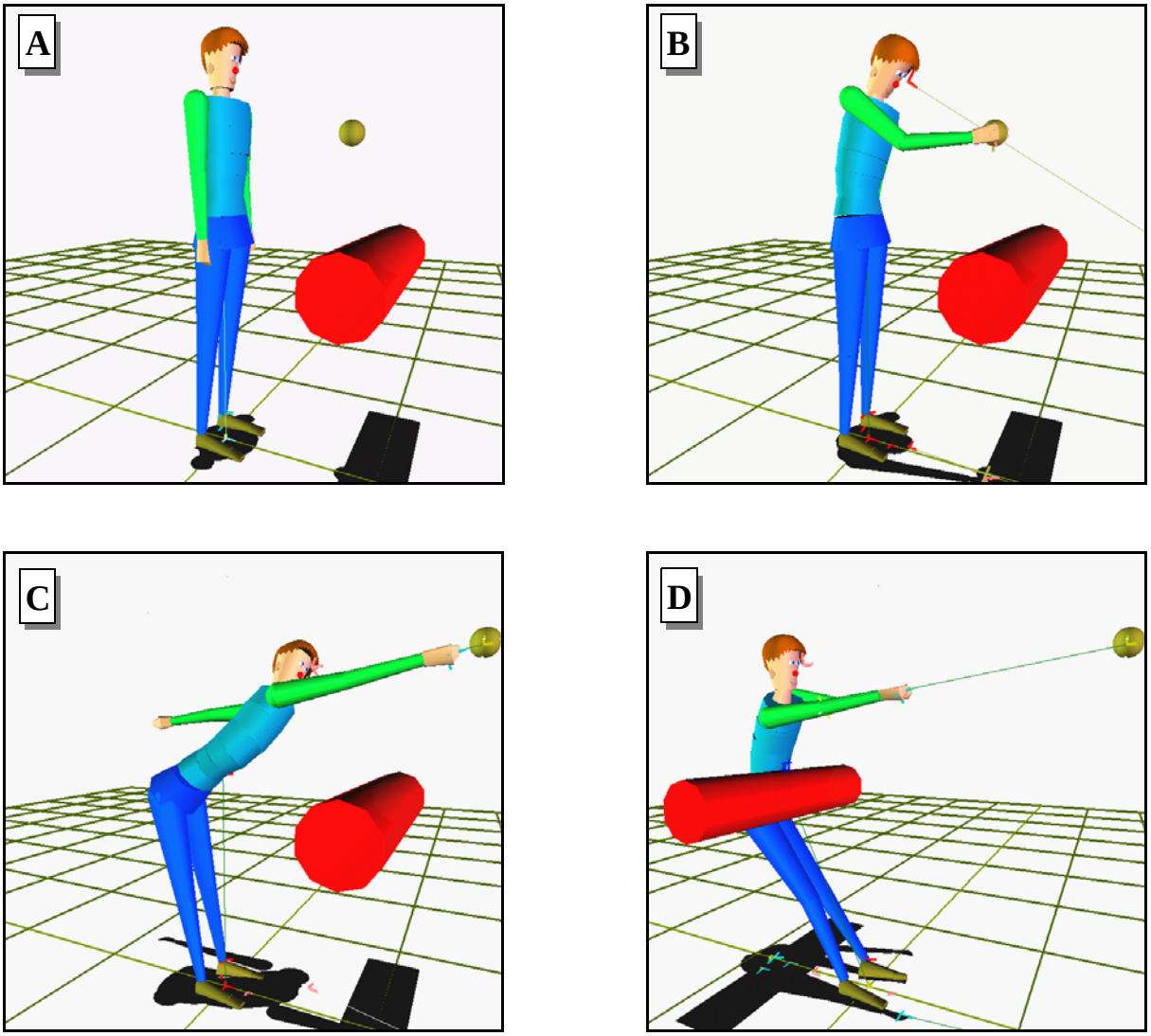


Figure 7.13 A sequence of postures respecting the priority order established in Table 7.1.

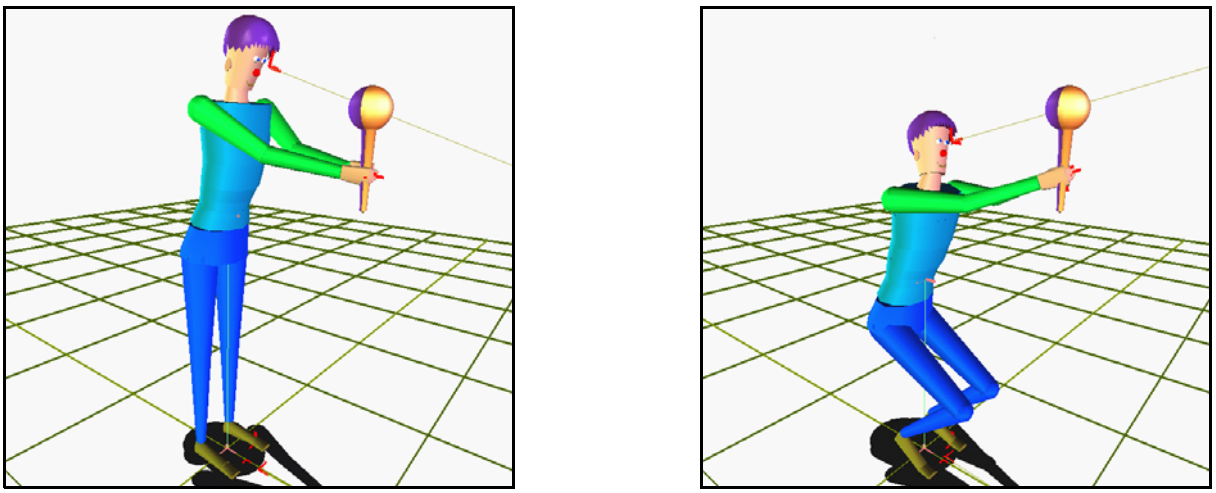


Figure 7.14 Flexing the legs while ensuring a set of constraint tasks.

7.3.8 Force exertion examples with a human figure

In this section we show how forces applied at selected end-effectors can be used to constrain the posture, with the technique described in Chapter 6. In Fig. 7.15 a simple example shows how a change in the distribution of body weight over the feet affects the posture. The weight distribution is controlled through the supporting ratios described in Section 6.2.2, and accessible in the *Tasks* panel (see Fig. 7.2). On the left, the total weight is equally supported by the feet while, on the right, it is mainly supported by the right foot: this results in a displacement of the center of mass over that foot.

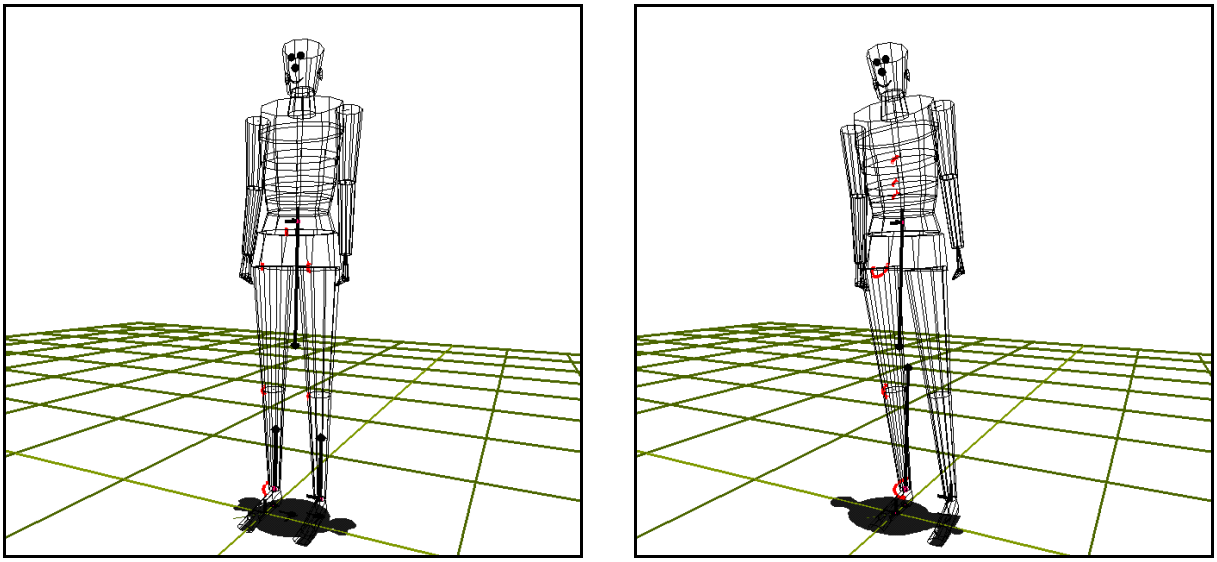


Figure 7.15 Two different distributions of body weight over the two supporting sites.

The second example (Fig. 7.16) shows the same figure, still standing on the right foot, while the right hand is constrained to lie on a vertical plane. Initially no force is applied at that end-effector (**A**), but then a vertical force is applied (**B**) - for example to indicate the weight of an object. In (**C**), an horizontal force is applied, as if the figure was trying to pull something. The hand is always free to move on the plane, so its position has changed. In (**D**), the figure is constrained to look forward on the hand, in order to obtain a more realistic posture.

The third example (Fig. 7.17) shows a figure trying to push a heavy object. Both feet are constrained on the floor, and the hands are constrained on the corners of the object. Two equal forces are applied at the hands. Thus the center of mass heads towards the object.

In the last example (Fig. 7.18), the figure tries to twist a heavy object. The hands are constrained to lie on the corners of the object, and two opposite forces are exerted on them. This would rotate the object anti-clockwise if it was lighter. The gazing direction is also constrained to obtain a more natural posture.

The last example also shows a difficulty with the proposed method: it is not always obvious to

specify the forces acting at the interface with the environment. Here, the forces acting between the feet and the floor must be specified so that they produce a torque that compensates the torque due to the forces applied by the hands on the object. Of course, this is not obvious to do manually. A solution would be to specify only the vertical components of the reaction forces acting at the feet, and leaving to the system the duty of computing the horizontal (frictional) components that realize the zero torque requirement.

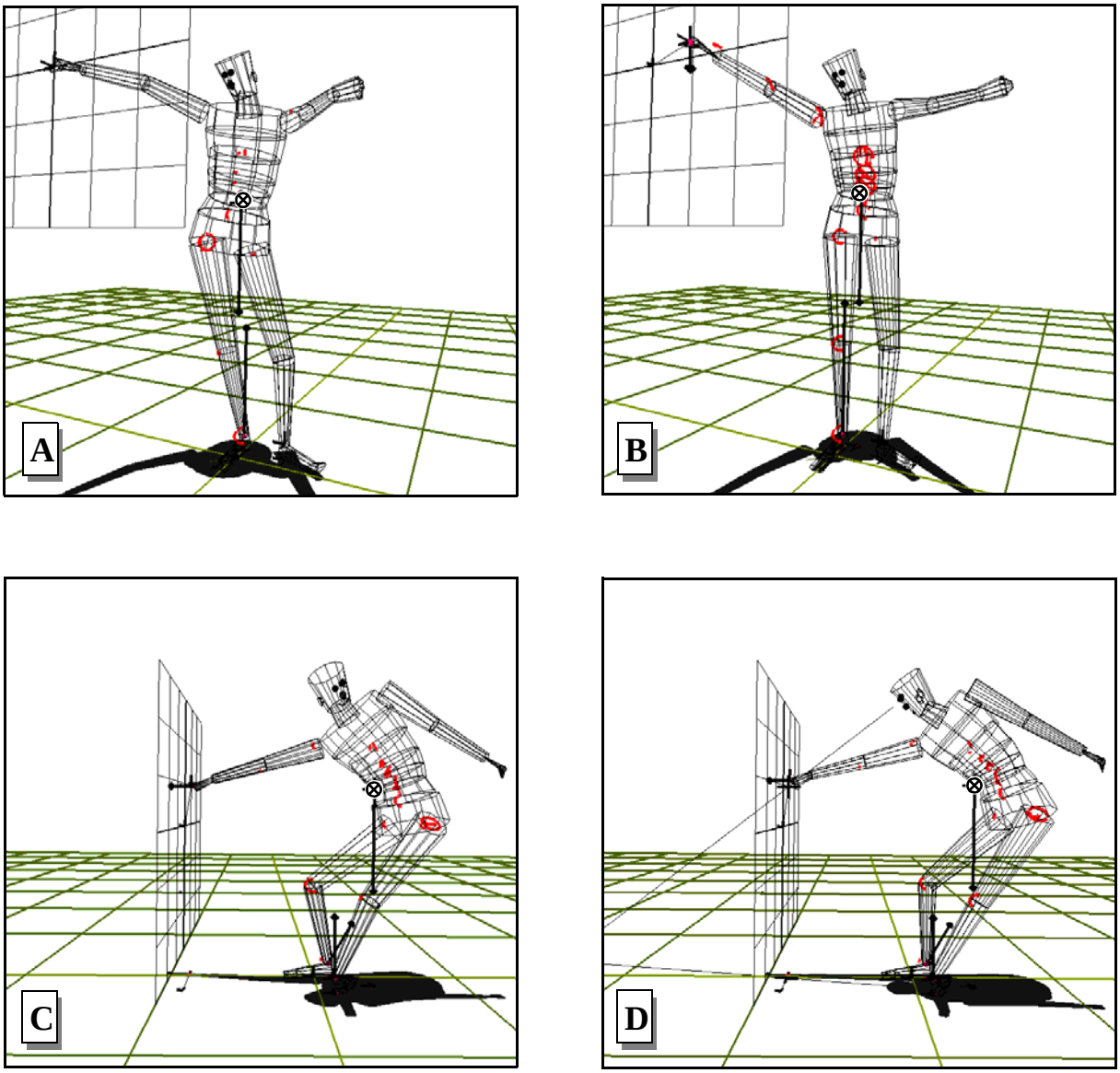


Figure 7.16 Example with force exertion, with a constraint on the position of the right hand.

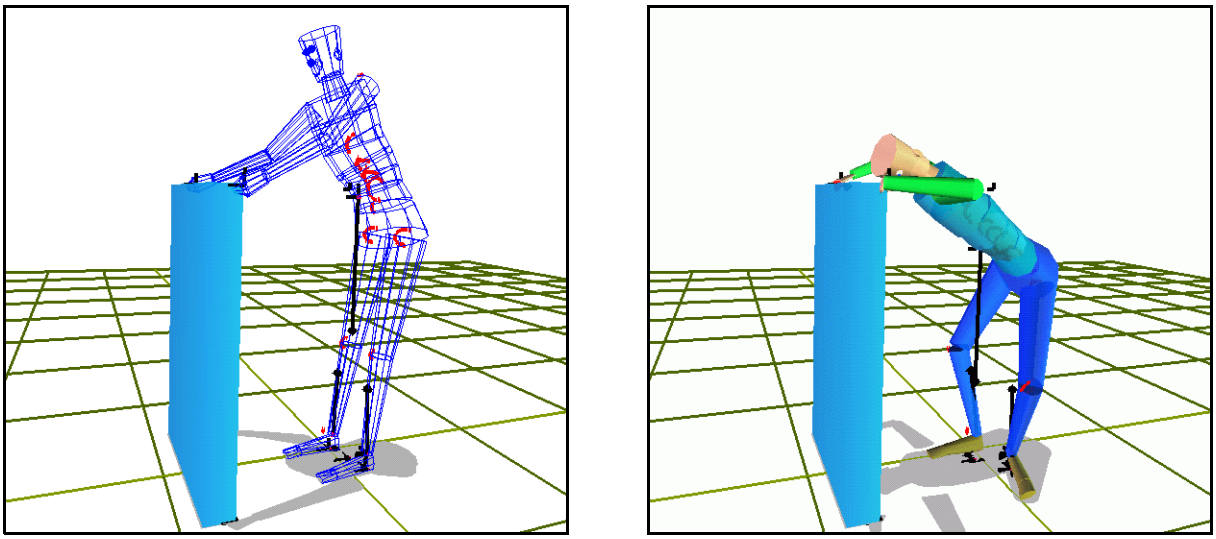


Figure 7.17 Trying to push a heavy object with both hands.

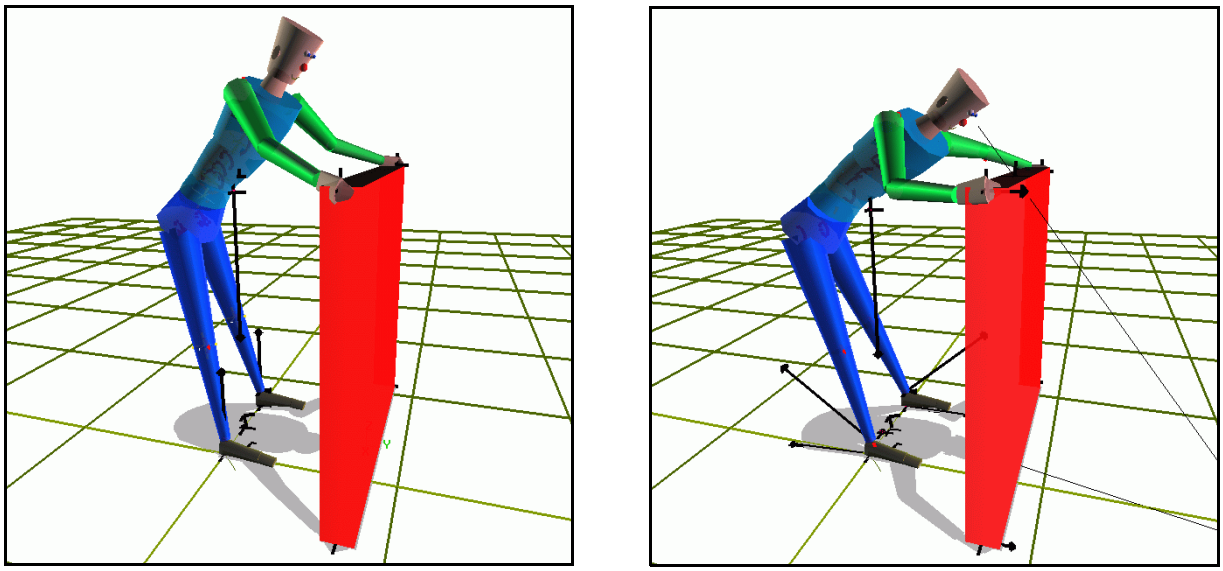


Figure 7.18 Trying to twist a heavy object. On the left, no force is applied on the object. On the right, two opposite forces with equal magnitude are applied by the hands on the object.

7.4 Conclusion

In this chapter, we have given several examples that illustrate the use of a task-priority strategy to deal with conflicting tasks. The concept is interesting mainly for two reasons:

- It is an intuitive concept. Everybody has a clear idea of what “priority” means: it occurs in many situations of everyday life. The concept of weight also seems intuitive but its exact meaning is not always clear (see Section 5.2.2). Despite this limitation, weights are useful

to smoothly modify the posture (i.e. to “interpolate” between solutions, in task space), and also to normalize the units of tasks that have different orders of magnitude (such as millimeters and radians).

- It arbitrates conflicts with a policy that suits to many situations, and which is often more appropriate than simply weighting the residual errors. The solution strictly satisfies the most important task.

An important feature of the algorithm is that while respecting the priority order, it also provides least-squares solutions for each task, which means that the solution is optimal (for example, it minimizes the distance between the end-effector and its goal).

Except for “hard” constraints (such as joint limits), every task may be seen as desirable but not necessarily achievable because it would not be feasible alone or because it conflicts with a task of higher priority. With the task-priority strategy, these two causes are unified, in the sense that, when a task is not achievable whatever the reason, a least-squares solution for that task is found. This is in contrast with the weighting strategy, that finds a least-squares solution to the whole set of tasks.

Finally, the computational complexity of the algorithm may be an obstacle to real-time interaction. However, even the most complex of the above-mentioned examples have not suffered of undesirable slow-downs, at least on modern machines. The use of recursion relations and appropriate numerical techniques certainly contributes to this result.

Chapter 8

Conclusion

8.1 Discussion

While investigating the problem of interactive posture control by means of inverse kinematics techniques, we were confronted to the problem of resolving conflicting situations between tasks. The most obvious solution was to find a compromise solution, but then no task was exactly satisfied. This was a problem because some tasks had to be ensured at any cost: for example, there was no reason for loosing balance just to allow a reaching task to approach its goal. Moreover, it was not clear how to specify weights to indicate the higher importance of the balance task, especially when there were several degrees of importance. To remove this limitation, a clear task prioritization had to be introduced.

We have come up with a numerical resolution framework capable of solving multiple tasks simultaneously, and that supports both task-priority and weighting strategies for the resolution of conflicts. Joint limits and joint coupling are also integrated in the framework, at least if they are modeled as linear equations.

The algorithm inherits the well-known limitation of iterative numerical resolution methods: the solution is only locally optimal. This is not a major problem in an interactive environment, because the user can often resolve these situations by hand.

The proposed framework is well-suited for real-time applications such as the interactive manipulation of articulated figures. The complexity of the algorithm with respect of the number of priority levels is linear, thanks to a recursive relation that we have proposed.

In conclusion, the effectiveness of our framework has been demonstrated for the interactive manipulation of complex articulated figures. Moreover, its use is certainly not limited to articulated figures and it may be applicable to other modelling or manipulation systems based on the resolution of a set of possibly conflicting tasks.

8.2 Contributions

Our main contributions are the following:

- We have shown the attractiveness, for the interactive manipulation of articulated figures, of a task-priority strategy to resolve certain task conflicts.
- We have compared and evaluated two task-priority algorithms based on the null space projection operator.
- A recursive formula has been proposed to speed-up the computation of the task-priority algorithm: its computational complexity, with respect to the number of priority levels p, has been reduced from $O(p^2)$ to $O(p)$.
- We have proposed an inverse kinematics framework that handles both priority and weighting strategies for the resolution of task conflicts, and that also respects linear joint limits and joint couplings.
- We have shown that the kinetic Jacobian matrix, used for the differential control of the center of mass, also possesses a meaning in the static force domain:
  - by the Jacobian transpose relationship, it relates the weight of a structure in static equilibrium to the generalized joint torques required to resist to that force;
  - at the differential level, it can be used to compute the variation of the Cartesian torque vector acting on the figure due to a variation of the joint coordinates. Within the inverse kinematics framework, this is exploited to ensure the static equilibrium condition.

8.3 Future topics

To conclude, we suggest a few directions for future research.

- The introduction of a floating base in the body model would remove the need of selecting a root for the hierarchy and also of re-rooting. To achieve this, the inverse kinematics engine should be extended to control the floating base.
- A more realistic human spine model would enhance the realism of the postures.
- A self-collision avoidance mechanism would greatly enhance the utility of the tool. This is a difficult problem to solve with the real-time requirement.
- The postures of hand and feet at contact sites with the environment is presently not satisfying. A contact site is not merely a point, but it is a surface or an object on which the body extremity must be realistically connected.
- Improvements must be done in the presence of redundancy. A joint should not be recruited to participate to the achievement of a task if this is not natural. For example, the position of the hand may be changed by moving the arm or the back, but the back should not move if the goal is close to the body. A possibility would be to assign weights to each joint, in order to represent its degree of involvement in a given task. Hence, for that example, arm joints would have higher weight than joints of the back.

- An interesting experience would be to exploit the computed joint torques to drive an anatomic-based rendering of the skin of a human or animal figure: the skin shape depends on the activation (and hence on the exerted torque) of the underlying muscles. Since this information is usually not available, the shape of the muscles are often based only on the angles of the proximal joint(s), which is clearly not sufficient for an accurate representation of muscle shape.

List of symbols and definitions

Frames

$\boldsymbol{O}_E$	Origin of a frame named *E*.
$\boldsymbol{O}_j$	Origin of the frame associated to the jth joint.

Jacobian matrices

J_i	Jacobian associated to the i^{th} task.
J_i^A	Augmented Jacobian (made of Jacobians J_1 ... J_i with same number of columns).
J_T	Translation Jacobian - to control the origin of an end-effector frame.
J_R	Rotation Jacobian - to control the orientation of an end-effector frame.
J_E	End-effector Jacobian - to control both position and orientation of its frame.
J_G	Kinetic Jacobian - to control the position of the center of mass.

Mass properties

m_{tot}, $\boldsymbol{G}_{tot}$	Total mass and center of mass of the structure ($m_{tot} = m_j^L + m_j^U, \forall j$).
m_j^U, $\boldsymbol{G}_j^U$	Mass and center of mass of the upper body (or augmented body) of the jth joint.
m_j^L, $\boldsymbol{G}_j^L$	Mass and center of mass of the lower body of the jth joint.

Operators

$\boldsymbol{a} \times \boldsymbol{b}$	The cross product between two vectors $\boldsymbol{a}, \boldsymbol{b} \in \Re^3$.
$[\boldsymbol{a} \times]$	The unary cross operator, satisfying $[\boldsymbol{a} \times]\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$ (see Appendix A).
∇	The gradient operator: $\nabla h(\boldsymbol{q}) := (\partial h(\boldsymbol{q}) / \partial q_1, \ldots, \partial h(\boldsymbol{q}) / \partial q_n)^T$, with $\boldsymbol{q} \in \Re^n$
$R_{\boldsymbol{a}}(\theta)$	Rotation about an axis $\boldsymbol{a}$ by an angle θ.

$R_D(\boldsymbol{a}, \boldsymbol{b})$	Rotation that transforms a unit vector $\boldsymbol{a}$ into a unit vector $\boldsymbol{b}$, with the smallest angle of rotation. It is called a *direct rotation*. Undefined if $\boldsymbol{b} = -\boldsymbol{a}$.
M^{-1}	The usual inverse of a square matrix M (exists if and only if the matrix is not singular).
$M^{\dagger}$	The least-squares inverse of a matrix M, also known as Moore-Penrose pseudoinverse.
$M^{\dagger\lambda}$	The damped least-squares inverse (λ is the damping factor).
$\|\boldsymbol{v}\|$	Euclidean norm of vector $\boldsymbol{v}$: $\|\boldsymbol{v}\| := \sqrt{\boldsymbol{v}^T\boldsymbol{v}}$
$\|\boldsymbol{v}\|_W$	Weighted norm of vector $\boldsymbol{v}$: $\|\boldsymbol{v}\|_W := \sqrt{\boldsymbol{v}^T W \boldsymbol{v}}$
sinc *(x)*	sin*(x)/x*, or 1 if *x* is zero.
atan2 *(y,x)*	Returns the polar angle of point $\boldsymbol{P}(x, y)$, in the range [-π..+π]

Linear subspaces

S	A generic subspace.
$S^{\perp}$	The orthogonal complement of subspace S, defined by $S^{\perp} := \{\boldsymbol{z} \in \Re^m \mid \boldsymbol{y}^T\boldsymbol{z} = 0, \forall \boldsymbol{y} \in S\}$, where m is the dimensionality of S.
$R(J)$	The *range* of an $m \times n$ matrix J is the subspace defined by $R(J) := \{\boldsymbol{v} \in \Re^m \mid \exists \boldsymbol{w} \in \Re^n, J\boldsymbol{w} = \boldsymbol{v}\}$.
$N(J)$	The *null space* of an $m \times n$ matrix J is the subspace defined by $N(J) := \{\boldsymbol{v} \in \Re^n \mid J\boldsymbol{v} = \boldsymbol{0}\}$
rank (J)	The *rank* of matrix *J*, which is the dimensionality of *R (J)*.
null (J)	The *nullity* of matrix *J*, which is the dimensionality of *N (J)*.
B_S	A matrix whose columns are mutually orthonormal, and that span subspace *S*.
P_S	Orthogonal projector on subspace *S*. P_S is a symmetric and idempotent matrix.

Constants

$\boldsymbol{0}$	A zero vector.
I_n	The $n \times n$ identity matrix.

Appendices

Appendix A: Properties of the unary cross operator.

The unary cross operator of a vector $\boldsymbol{v}$ is defined as follows:

$$[\boldsymbol{v}\times]:=\begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

Here are its main properties. Given two vectors $\boldsymbol{a}, \boldsymbol{b} \in \Re^3$ and a scalar k:

$$\begin{aligned} \boldsymbol{a}\times\boldsymbol{b} &= [\boldsymbol{a}\times]\boldsymbol{b} \\ rank([\boldsymbol{a}\times]) &= 2 \qquad if\ \ \boldsymbol{a}\neq\boldsymbol{0} \\ k[\boldsymbol{a}\times] &= [(k\boldsymbol{a})\times] \\ [\boldsymbol{a}\times]+[\boldsymbol{b}\times] &= [(\boldsymbol{a}+\boldsymbol{b})\times] \\ [\boldsymbol{a}\times]^T &= -[\boldsymbol{a}\times] \\ [(\boldsymbol{a}\times\boldsymbol{b})\times] &= [\boldsymbol{a}\times][\boldsymbol{b}\times]-[\boldsymbol{b}\times][\boldsymbol{a}\times] = \boldsymbol{b}\boldsymbol{a}^T-\boldsymbol{a}\boldsymbol{b}^T \\ [(R\boldsymbol{a})\times] &= R[\boldsymbol{a}\times]R^T \quad \text{where R is an orthonormal matrix} \end{aligned}$$

Appendix B: Basics of quaternion algebra.

Quaternions generalize complex numbers, and are very useful to deal with 3D rotations.

- Quaternion definition:

  A quaternion $\boldsymbol{e}$ is made of a vectorial part $\boldsymbol{e_v} = (e_x, e_y, e_z)$ and of a scalar part e_s. It is noted $\boldsymbol{e} = (\boldsymbol{e_v}, e_s)$.

- Quaternion norm:

  The norm of a quaternion $\boldsymbol{e}$ is $\|\boldsymbol{e}\| = \sqrt{\boldsymbol{e_v}^T\boldsymbol{e_v}+e_s^2}$. A quaternion whose norm is 1 is called a unit quaternion.

- Quaternion multiplication between two quaternions $\boldsymbol{e}$ and $\boldsymbol{f}$ (non-commutative):

$$\boldsymbol{e}\boldsymbol{f} = (e_s\boldsymbol{f_v}+f_s\boldsymbol{e_v}+\boldsymbol{e_v}\times\boldsymbol{f_v},\ e_sf_s-\boldsymbol{e_v}^T\boldsymbol{f_v}) \tag{B.1}$$

- Conjugate quaternion:

  Given a quaternion $\boldsymbol{e}$, its conjugate is defined as $\tilde{\boldsymbol{e}}:=(-\boldsymbol{e_v}, e_s)$

- Unit quaternions and rotations:

  For each 3D rotation defined by its axis of rotation $\boldsymbol{n}$ and its angle of rotation θ, an equivalent unit quaternion can be defined: $\boldsymbol{e} = (\boldsymbol{n}\sin(\theta/2), \cos(\theta/2))$, and the four components are also known as Euler parameters. Note that the opposite quaternion $-\boldsymbol{e}$ represents the same rotation. Quaternion multiplication is equivalent to the combination of rotations.

- Rotation of a vector $\boldsymbol{p}$ by a unit quaternion $\boldsymbol{e} = (\boldsymbol{e_v}, e_s)$:

$$(\boldsymbol{p}', 0) = \boldsymbol{e}\boldsymbol{f}\tilde{\boldsymbol{e}} = (e_s - \boldsymbol{e_v}^T\boldsymbol{e_v})\boldsymbol{p} + 2(e_s(\boldsymbol{e_v} \times \boldsymbol{p}) + (\boldsymbol{p}^T\boldsymbol{e_v})\boldsymbol{e_v}) \quad \text{(B.2)}$$

  where $\boldsymbol{f} = (\boldsymbol{p}, 0)$ is a quaternion and $\boldsymbol{p}'$ is the rotated vector.

Sources of information about quaternions are [SCH 85] [MUR 94] [WAT 92] [GRA 98].

Appendix C: Swing and twist decomposition of an orientation.

Problem: Let a given orientation be described by a unit quaternion $\boldsymbol{e} = ((e_x, e_y, e_z), e_w)$. We want to decompose it into a swing, represented by the axis-angle $\begin{bmatrix} s_x & s_y & 0 \end{bmatrix}^T$, followed by a twist by an angle τ about the $\boldsymbol{z}$ axis, represented by the axis-angle $\begin{bmatrix} 0 & 0 & \tau \end{bmatrix}^T$.

Solution: If both e_z and e_w are zero, the orientation is at the singularity of the swing component: thus, an infinity of swings and twists exist that match the orientation so that one may arbitrarily choose a twist. Otherwise, the twist is $\tau = 2\operatorname{atan2}(e_z, e_w)$.

Once the twist component is known, the swing component is computed as follows:

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \frac{2}{\operatorname{sinc}(\beta)} \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix}$$

where $\gamma = \tau/2$ and $\beta = \operatorname{atan2}(\sqrt{e_x^2 + e_y^2}, \sqrt{e_z^2 + e_w^2})$.

Note that $0 \le \beta \le \pi/2$, and that $-2\pi \le \tau \le 2\pi$.

Appendix D: Projection of a point over an ellipse.

Problem: Given an axis-aligned ellipse centered at the origin, with semi-axes lengths r_x and r_y, find the point on the ellipse which has the smallest Euclidean distance to a given arbitrary point $\boldsymbol{P}$ of coordinates P_x and P_y.

Solution: First find the real solutions λ_i to the following polynomial equation of the 4th degree:

$P_y\lambda^4 + (f-e)\lambda^3 + (f+e)\lambda - P_y = 0$, where $e = 2(r_y - r_x^2/r_y)$ and $f = 2P_x r_x / r_y$.

This can be done analytically. Then compute the distance between each candidate point $Q_i\left(r_x\frac{1-\lambda_i^2}{1+\lambda_i^2}, r_y\frac{2\lambda_i}{1+\lambda_i^2}\right)$ and the given point P, and keep the candidate with the smaller distance.

Method: This result is obtained by the minimization of the distance between point ***P*** and an arbitrary point $Q(r_x\cos(\alpha), r_y\sin(\alpha))$ on the ellipse. Instead of solving for α, we solve for $\lambda = \tan(\alpha/2)$: this substitution allows to transform a transcendental equation into a simpler polynomial equation.

Appendix E: Basic rules of vectorial differentiation.

$$\frac{d}{d\boldsymbol{a}}(\boldsymbol{a}\times\boldsymbol{b}) = -[\boldsymbol{b}\times]$$

$$\frac{d}{d\boldsymbol{q}}(\boldsymbol{a}(\boldsymbol{q})\times\boldsymbol{b}(\boldsymbol{q})) = \frac{d\boldsymbol{a}(\boldsymbol{q})}{d\boldsymbol{q}}\times\boldsymbol{b}(\boldsymbol{q}) + \boldsymbol{a}(\boldsymbol{q})\times\frac{d\boldsymbol{b}(\boldsymbol{q})}{d\boldsymbol{q}}$$

$$\frac{d}{d\boldsymbol{a}}((\boldsymbol{a}^T\boldsymbol{a})\boldsymbol{b}) = 2\boldsymbol{b}\boldsymbol{a}^T$$

$$\frac{d}{d\boldsymbol{a}}((\boldsymbol{a}^T\boldsymbol{b})\boldsymbol{a}) = \boldsymbol{a}\boldsymbol{b}^T + (\boldsymbol{a}^T\boldsymbol{b})I_3$$

$$\frac{d}{d\theta}(R_{\boldsymbol{a}}(\theta)) = [\boldsymbol{a}\times]R_{\boldsymbol{a}}(\theta)$$

$$\frac{d}{d\alpha}([\boldsymbol{a}(\alpha)\times]) = [\frac{d\boldsymbol{a}(\alpha)}{d\alpha}\times]$$

$$d\frac{(\boldsymbol{a}/\|\boldsymbol{a}\|)}{d\boldsymbol{a}} = \frac{\|\boldsymbol{a}\|^2 I_3 - \boldsymbol{a}\boldsymbol{a}^T}{\|\boldsymbol{a}\|^3}$$

More results can be found in [LUE 96].

Appendix F: Mapping from axis-angle variations to quaternion variations.

Given an axis-angle $\boldsymbol{k} \in \Re^3$, we define:

$$\theta = \|\boldsymbol{k}\| \text{ and } \boldsymbol{a} = \begin{cases} \boldsymbol{k}/\theta & \text{if } \theta > 0 \\ \text{otherwise undefined} \end{cases}$$

Let $\boldsymbol{e} = (\boldsymbol{e}_v, e_s) = (\boldsymbol{a}\sin(\theta/2), \cos(\theta/2))$ be the quaternion equivalent to axis-angle $\boldsymbol{k}$.

We can compute the 4×3 Jacobian matrix that relates differential changes of axis-angle $\boldsymbol{k}$ to differential changes of quaternion parameters $\boldsymbol{e}$:

$$J_{\boldsymbol{k}}^{\boldsymbol{e}} = \frac{d\boldsymbol{e}}{d\boldsymbol{k}} = \frac{\sin(\theta/2)}{\theta}\begin{bmatrix} I_3 + \dfrac{\cot(\theta/2)/(2\theta) - 1}{\theta^2}(\boldsymbol{k}\boldsymbol{k}^T) \\ -\boldsymbol{k}^T/2 \end{bmatrix} \text{ if } \theta > \varepsilon \qquad \text{(F.1)}$$

When θ is small, the result (obtained by computing the limit of Eq. (F.1) when $\theta \to 0$) is:

$$J_{\boldsymbol{k}}^{\boldsymbol{e}} = \frac{1}{2}\begin{bmatrix} I_3 \\ -\boldsymbol{k}^T/2 \end{bmatrix}$$

Appendix G: Recursion relation for nullspace projectors.

Proposition: Given an $m \times n$ matrix J_i^A partitioned as $\begin{bmatrix} J_{i-1}^A \\ J_i \end{bmatrix}$, the following identity holds:

$$I_n - J_i^{A\dagger} J_i^A = (I_n - J_{i-1}^{A\ \dagger} J_{i-1}^A) - \tilde{J}_i^{\dagger} \tilde{J}_i$$

$$\text{where } \tilde{J}_i = J_i(I_n - J_{i-1}^{A\ \dagger} J_{i-1}^A)$$

Demonstration: A result due to R. Cline [CLI 64] [BOUL 71] yields the pseudoinverse of a partitioned matrix. It is applied on J_i^A:

$$J_i^{A\dagger} = (J_{i-1}^{A\ \dagger} - T_i J_i J_{i-1}^{A\ \dagger}, T_i)$$

where $T_i = \tilde{J}_i^{\dagger} + X(I - \tilde{J}_i \tilde{J}_i^{\dagger})$ and X is a complex term not useful here.

The identity can now be established:

$$\begin{aligned} I_n - J_i^{A\dagger} J_i^A &= I_n - (J_{i-1}^{A\ \dagger} - T_i J_i J_{i-1}^{A\ \dagger}, T_i)\begin{bmatrix} J_{i-1}^A \\ J_i \end{bmatrix} \\ &= I_n - J_{i-1}^{A\ \dagger} J_{i-1}^A - T_i J_i (I_n - J_{i-1}^{A\ \dagger} J_{i-1}^A) \\ &= I_n - J_{i-1}^{A\ \dagger} J_{i-1}^A - T_i \tilde{J}_i \\ &= I_n - J_{i-1}^{A\ \dagger} J_{i-1}^A - (\tilde{J}_i^{\dagger} + X(I - \tilde{J}_i \tilde{J}_i^{\dagger}))\tilde{J}_i \\ &= (I_n - J_{i-1}^{A\ \dagger} J_{i-1}^A) - \tilde{J}_i^{\dagger} \tilde{J}_i \end{aligned}$$

In the last step, the pseudoinverse property $JJ^{\dagger}J = J$ is used.

Appendix H: Another useful recursion relation.

Proposition: Given an $m \times n$ matrix $C_m^A = \left[\boldsymbol{c}_1\ \boldsymbol{c}_2\ \ldots\ \boldsymbol{c}_m\right]^T$ where the $\boldsymbol{c}_i$'s are n-dimensional, linearly independent vectors, and an m-dimensional vector $\boldsymbol{b}_m^A = \left[b_1\ b_2\ \ldots\ b_m\right]^T$, the sequence of $\boldsymbol{q}_i = C_i^{A\dagger}\boldsymbol{b}_i^A$ and of $P_{N(C_i^A)}$, with $i = 1 \ldots m$, can be computed with the following recursion relations:

$$\boldsymbol{q}_i = \boldsymbol{q}_{i-1} + \frac{(b_i - \boldsymbol{c}_i^T \boldsymbol{q}_{i-1})}{\tilde{\boldsymbol{c}}_i^T \tilde{\boldsymbol{c}}_i}\tilde{\boldsymbol{c}}_i$$

$$P_{N(C_i^A)} = P_{N(C_{i-1}^A)} - \frac{\tilde{\boldsymbol{c}}_i \tilde{\boldsymbol{c}}_i^T}{\tilde{\boldsymbol{c}}_i^T \tilde{\boldsymbol{c}}_i}$$

where

$$\tilde{\boldsymbol{c}}_i = P_{N(C_{i-1}^A)}\boldsymbol{c}_i$$

and

$$\boldsymbol{q}_0 = \boldsymbol{0}$$

$$P_{N(C_0^A)} = I_n$$

Demonstration: see [ALB 66] or [BOUL 71].

Bibliography

[ALB 66] A. Albert, R.W. Sittler, "**A Method for Computing Least Squares Estimators that Keep Up with the Data**", *Journal of Soc. Industrial Applied Mathematics Control*, Ser. A, Vol. 3, No. 3, pp. 384 - 417, 1966.

[ALI 98] Maya Alias/Wavefront, "**Maya User Manual**", Using Maya : Animation, Version 1.0, Chapter 12, 1998.

[ANA 92] G. Anandalingam, T.L. Friesz, "**Hierarchical Optimization**", *Annals of Operations Research*, J.C. Baltzer AG, Scientific Publishing Company, Basel, Vol. 34, No. 1, 1992.

[AND 96] R. Anderl, R. Mendgen, "**Modelling with Constraints: Theoretical Foundation and Application**", *Computer-Aided Design*, Vol. 28, No. 3, pp. 155 - 168, 1996.

[AYD 99] Y. Aydin, M. Nakajima, "**Database guided computer animation of human grasping using forward and inverse kinematics**", *Computers & Graphics*, Vol. 23, No. 1, pp. 145 - 154, 1999.

[AYD 99b] Y. Aydin, M. Nakajima, "**Realistic Articulated Character Positioning and Balance Control in Interactive Environments**", *Proceedings of Computer Animation '99*, pp. 160 - 168, 1999.

[AYD 99c] Y. Aydin, M. Nakajima, "**Balance Control and Mass Centre Adjustment of Articulated Figures in Interactive Environments**", *The Visual Computer*, pp. 113 - 123, 1999.

[AYO 81] M. Ayoub, C. Gidcumb, M. Reeder, M. Beshir, H. Hafez, F. Aghazadeh,, N. Bethea, "**Development of an Atlas of Strength and establishment of an appropriate Model Structure**", Technical Report, Texas Tech University, 1981.

[AYO 87] M. Ayoub, A. Mital, "**Manual Materials Handling**", Taylor & Francis, 1987.

[BAD 80] N. Badler, J. O'Rourke, B. Kaufman, "Spe**cial problems in human movement simulation**", *Computer Graphics (SIGGRAPH 80)*, Vol. 14, No. 3, pp. 189 - 197, 1980.

[BAD 87] N.I. Badler, K.H. Manoochehri, G. Walters, "**Articulated Figure Positioning by Multiple Constraints**", *IEEE Computer Graphics & Applications*, Vol. 7, No. 6, pp. 28 - 38, June 1987.

[BAD 90] N. Badler, K. Manoochehri, G. Walters, "**Articulated Figure Positioning by Multiple Constraints**", *SIGGRAPH 90 Course notes (Human figure animation : approaches and applications)*, May 1990.

[BAD 91] N. Badler, "**Human Factors Simulation Research at the University of Pennsylvania**", *SIGGRAPH 91 Course notes (Advanced techniques in human modeling, animation, and rendering)*, 1991.

[BAD 93] N.I. Badler, C.B. Phillips, B.L. Webber, "**Simulating Humans, Computer Graphics Animation and Control**", New York, Oxford University Press, 1993.

[BAD 93b] N.I. Badler, M.J. Hollick, John P. Granieri, "**Real-Time Control of a Virtual Human Using Minimal Sensors**", *Presence*, Vol. 2, No. 1, pp. 82 - 86, 1993.

[BAE 00] P. Baerlocher, R. Boulic, "**Kinematic Control of the Mass Properties of Redundant Articulated Bodies**", *Proceedings of IEEE Conference ICRA 2000*, San Francisco, CA, pp. 2557 - 2562, Apr. 2000.

[BAE 98] P. Baerlocher, R. Boulic, "**Task-Priority Formulations for the Kinematic Control of Highly Redundant Articulated Structures**", *Proceedings of IEEE Conference "Intelligent Robots and Systems" IROS '98*, Victoria (Canada), pp. 323 - 329, Oct. 1998.

[BAI 85] J. Baillieul, "**Kinematic Programming Alternatives for Redundant Manipulators**", *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 722 - 728, 1985.

[BAR 88] R. Barzel, A. Barr, "**A Modeling System Based on Dynamic Constraints**", *SIGGRAPH 88*, pp. 179 - 188, 1988.

[BEC 92] D.J. Beck, D.B. Chaffin, "**An evaluation of inverse kinematics models for posture prediction**", *Computer Applications in Ergonomics, Occupational Safety and Health (CAES '92)*, Mattila M., Karwowski W. (eds), North-Holland, pp. 329 - 336, 1992.

[BEN 65] A. Ben-Israel, "**A modified Newton-Raphson method for the solution of systems of equations**", *Israel Journal of Mathematics*, Vol. 3, pp. 94 - 99, 1965.

[BEN 66] A. Ben-Israel, "**A Newton-Raphson method for the solution of systems of equations**", *Journal of Mathematical Analysis and Applications*, Vol. 15, pp. 243 - 252, 1966.

[BEN 74] A. Ben-Israel, T.N.E. Greville, "**Generalized inverses: theory and applications**", John Wiley & Sons, 1974.

[BJO 96] A. Björck, "**Numerical Methods for Least-Squares Problems**", SIAM, Philadelphia, 1996.

[BOU 90] R. Boulic, N.M. Thalmann, D. Thalmann, "**A global human walking model with real-time personnification**", *The Visual Computer*, Vol. 6, pp. 344 - 358, 1990.

[BOU 92] R. Boulic, D. Thalmann, "**Combined direct and inverse kinematic control for articulated figures motion editing**", *Computer Graphics Forum*, Vol. 2, No. 4, 1992.

[BOU 94] R. Boulic, R. Mas, D. Thalmann, "**Inverse Kinetics for Center of Mass Position Control and Posture Optimization**", *Race Workshop on "Combined real and synthetic image processing for broadcast and video production (Monalisa Project)"*, Hamburg, Y. Parker & S. Wilbur Edt, Workshop in Computing Series, Springer-Verlag, 1994.

[BOU 96] R. Boulic, R. Mas, D. Thalmann, "**A Robust Approach for the Center of Mass Position Control with Inverse Kinetics**", *Journal of Computers and Graphics*, Vol. 20, No. 5, pp. 693 - 701, 1996.

[BOU 97] R. Boulic, R. Mas, D. Thalmann, "**Interactive Identification of the Center of Mass Reachable Space for an Articulated Manipulator**", *Proc. of International Conference of Advanced Robotics ICAR '97*, Monterey, pp. 589 - 594, July 1997.

[BOU 97b] R. Boulic, R. Mas, D. Thalmann, "**Complex Character Positioning Based on a Compatible Flow Model of Multiple Supports**", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 3, No. 3, pp. 245 - 261, Sept. 1997.

[BOUL 71] T. Boullion, P.L. Odell, "**Generalized Inverse Matrices**", John Wiley and Sons, New York, 1971.

[CAM 90] S. Cameron, "**Collision Detection by Four-Dimensional Intersection Testing**", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 3, pp. 291 - 302, June 1990.

[CHA 91] D.B. Chaffin, M. Erig, "**Three-Dimensional Biomechanical Static Strength Prediction Model Sensitivity to Postural and Anthropometric Inaccuracies**", *IIE Transactions*, Vol. 23, No. 3, pp. 215 - 227, 1991.

[CHA 99] D.B. Chaffin, G.B.J. Andersson, B.J. Martin, "**Occupational biomechanics**", New York, Wiley, 3rd edition, 1999.

[CHI 91] P. Chiacchio, S. Chiaverini, L. Sciavicco, B. Siciliano, "**Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators with Task Space Augmentation and Task Priority Strategy**", *Int. Journal of Robotics Research*, Vol. 10, No. 4, Aug. 1991.

[CHI 92] G.S. Chirikjian, J.W. Burdick, "**A Geometric Approach to Hyper-Redundant Manipulator Obstacle Avoidance**", *ASME Journal of Mechanical Design*, Vol. 114, pp. 580 - 585, Dec. 1992.

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

[CHI 94b] G.S. Chirikjian, J.W. Burdick, "**A Modal Approach to Hyper-Redundant Manipulator Kinematics**", *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 3, pp. 343 - 354, June 1994.

[CHI 97] S. Chiaverini, "**Singularity-Robust Task-Priority Redundancy Resolution for Real-Time Kinematic Control of Robot Manipulators**", *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 3, pp. 398 - 410, June 1997.

[CLE 90] K. Cleary, "**Incorporating Multiple Criteria in the Operation of Redundant Manipulators**", *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 618 - 624, 1990.

[CLI 64] R.E. Cline, "**Representations for the Generalized Inverse of a Partitioned Matrix**", *Journal of Soc. Industrial Applied Mathematics*, XII, pp. 588 - 600, 1964.

[CRA 89] J.J. Craig, "**Introduction to Robotics: mechanics and control**", 2nd edition, Addison-Wesley, 1989.

[DAA 94] B.J. Daams, "**Human Force Exertion in User-Product Interaction**", Delft University, 1994.

[DAS 88] H. Das, J-J. Slotine, T.B. Sheridan, "**Inverse kinematic algorithms for redundant systems**", *Proceedings of IEEE ICRA (International Conference on Robotics and Automation)*, pp. 43 - 48, 1988.

[DEO 97] A.S. Deo, I.D. Walker, "**Minimum Effort Inverse Kinematics for Redundant Manipulators**", *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 5, pp. 767 - 775, 1997.

[DOT 93] K.L. Doty, C. Melchiorri, C. Bonivento, "**A Theory of Generalized Inverses Applied to Robotics**", *The International Journal of Robotics Research*, Vol. 12, No. 1, pp. 1 - 19, Feb. 1993.

[DUB 88] R. Dubey, J. Luh, "**Redundant Robot Control Using Task Based Performance Measures**", *Journal of Robotic Systems*, Vol. 5, No. 5, pp. 409 - 432, 1988.

[EGE 87] O. Egeland, "**Task-space tracking with redundant manipulators**", *IEEE Journal of Robotics and Automation*, Vol. RA-3, pp. 471 - 475, 1987.

[EGE 91] O. Egeland, J.R. Sagli, I. Spangelo, S. Chiaverini, "**A Damped Least-Squares Solution to Redundancy Resolution**", *Proc. IEEE Int. Conference on Robotics and Automation*, Sacramento, CA, pp. 945 - 950, 1991.

[EME 00] L. Emering, R. Boulic, T. Molet, D. Thalmann, "**Versatily Tuning of Humanoid Agent Activity**", *Computer Graphics Forum*, Vol. 19, No. 4, pp. 231 - 242, 2000.

[ENG 89] A.E. Engin, S.T. Tuemer, "**Three dimensional kinematic modelling of human shoulder complex - Part I: Physical model and determination of joint sinus cones**", *Journal of Biomechanical Engineering*, Vol. 111, pp. 107 - 112, 1989.

[ESP 85] B. Espiau, R. Boulic, "**Collision Avoidance for Redundant Robots with Proximity Sensors**", *3rd Int. Symposium on Robotics Research*, Gouvieux, France, Oct. 1985.

[ESP 97] B. Espiau, "**BIP: a Joint Project for the Development of an Anthropomorphic Biped Robot**", *Int. Conference on Advanced Robotics*, ICAR '97, 1997.

[FEA 86] R. Featherstone, "**Robot Dynamics Algorithms**", Kluwer Academic Publishers, 1986.

[FLU 98] L. Flückiger, "**Interface pour le pilotage et l'analyse des robots basée sur un générateur de cinématiques**", Thêse No 1897, EPFL, Lausanne, 1998.

[GAR 75] A. Garg, D.B. Chaffin, "**A Biomechanical Computerized Simulation of Human Strength**", *AIIE Transactions*, pp. 1 - 15, March 1975.

[GAR 94] A. Garcia-Alonso, N. Serrano, J. Flaquer, "**Solving the Collision Detection Problem**", *IEEE Computer Graphics and Applications*, pp. 36 - 43, May 1994.

[GAR 94b] R. Garziera, "**Recursive Formulation for the Inverse Kinematics of Redundant Robots Performing Tasks with Priority Order**", *ASME Journal of Mechanical Design*, Vol. 116, No. 1, pp. 337 - 338, March 1994.

[GER 85] J.S. Gero, "**Design Optimization**", Notes and Reports in Mathematics, in Science and Engineering, Academic Press, Inc., 1985.

[GIR 85] M. Girard, A.A. Maciejewski, "**Computational Modeling for the Computer Animation of Legged Figures**", *Computer Graphics*, Vol. 19, No. 3, pp. 263 - 270, July 1985.

[GIR 87] M. Girard, "**Interactive Design of 3D Computer-Animated Legged Animal Motion**", *IEEE Computer Graphics and Applications*, pp. 39 - 51, June 1987.

[GLE 94] M. Gleicher, "**A Differential Approach to Graphical Interaction**", Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, 1994.

[GLE 98] M. Gleicher, "**Retargetting Motion to New Characters**", *Computer Graphics Proceedings*, pp. 33 - 42, 1998.

[GOL 69] D. Goldfarb, "**Extension of Davidon's Variable Metric Method to Maximisation under Linear Inequality and Equality Constraints**", *SIAM Journal Appl. Math.*, Vol. 17, pp. 739 - 764, 1969.

[GOL 85] A. Goldenberg, B. Benhabib, R.G. Fenton, "**A Complete Generalized Solution to the Inverse Kinematics of Robots**", *IEEE Journal of Robotics and Automation*, Vol. RA-I, No. 1, pp. 14 - 20, 1985.

[GOL 96] G.H. Golub, C.F. Van Loan, "**Matrix computations**", Third edition, Johns Hopkins University Press, 1996.

[GRA 98] F.S. Grassia, "**Practical Parameterization of Rotations using the Exponential Map**", *Journal of Graphics Tools*, Vol. 3, No. 3, pp. 29 - 48, 1998.

[GRE 59] T.N.E. Greville, "**The Pseudoinverse of a Rectangular or Singular Matrix and its Application to the Solution of Systems of Linear Equations**", *SIAM Review*, Vol. 1, No. 1, Jan. 1959.

[GRE 60] T.N.E. Greville, "**Some Applications of the Pseudoinverse of a Matrix**", *SIAM Review*, Vol. 2, No. 1, pp. 15 - 22, 1960.

[GRI 79] D.W. Grieve, "**The Postural Stability Diagram : personal constraints on the static exertion of forces**", *Ergonomics*, Vol. 22, No. 10, pp. 1155 - 1164, 1979.

[GRI 79b] D.W. Grieve, "**Environmental constraints on the static exertion of force: PSD analysis in task-design**", *Ergonomics*, Vol. 22, No. 10, pp. 1165 - 1175, 1979.

[GRO 89] M.R. Grosso, R.D. Quach, N.I. Badler, "**Anthropometry for Computer Animated Human Figures**", State-of-the-art in Computer Animation, *Proceedings of Computer Animation '89*, pp. 83 - 96, 1989.

[HAN 81] H. Hanafusa, T. Yoshikawa, Y. Nakamura, "**Analysis and Control of Articulated Robot with Redundancy**", *IFAC, 8th Triennal World Congress*, Vol. 4, pp. 1927 - 1932, 1981.

[HAS 90] C. Haslegrave, "**Postural constraints on force exertion**", PhD Thesis, Nottingham University, 1990.

[HIR 96] M. Hirose, G. Deffaux, Y. Nakagaki, "**Development of an effective motion capture system based on data fusion and minimal use of sensors**", *VRST 96*, July 1996.

[HIR 98] K. Hirai, M. Hirose, Y. Haikawa, T. Takenaka, "**The development of the Honda humanoid robot**", *Proceedings of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, pp. 1321 - 1326, 1998.

[HOD 95] J. Hodgins, W.L. Wooten, D.C. Brogan, J.F. O'Brien, "**Animating Human Athletics**", *Siggraph '95*, pp. 71 - 78, 1995.

[HUA 97] L-G. Huang, L. Yaotong, "**Kinematic Control of Redundant Manipulators : Implementation Issues**", *ICAR '97*, pp. 147 - 152, 1997.

[HUA 97b] Z. Huang, "**Motion control for human animation**", Ph. D. thesis #1601, LIG - EPFL, 1997.

[HUS 90] R.L. Huston, "**Multibody dynamics**", Butterworth-Heinemann, Stoneham, 1990.

[JUN 95] E.S. Jung, D. Kee, M.K. Chung, "**Upper body reach posture prediction for ergonomic evaluation models**", *International Journal of Industrial Ergonomics*, Vol. 16, pp. 95 - 107, 1995.

[KHA 85] O. Khatib, "**Real-Time Obstacle Avoidance for Manipulators and Mobile Robots**", *IEEE International Conference on Robotics and Automation*, St. Louis, 1985.

[KHW 98] A.A. Khwaja, M.O. Rahman, M.G. Wagner, "**Inverse Kinematics of Arbitrary Robotic Manipulators Using Genetic Algorithms**", in: "Advances in Robot Kinematics: Analysis and Control", Lenarcic and Husty (eds.), Kluwer Academic Publishers, pp. 375 - 382, 1998.

[KIN 81] E.C. Kingsley, N.A. Schofield, K. Case, "**SAMMIE - A computer aid for man-machine modeling**", *Computer Graphics*, Vol. 15, No. 3, pp. 163 - 169, 1981.

[KLE 83] C.A. Klein, C.H. Huang, "**Review of Pseudoinverse control for Use with Kinematically Redundant Manipulators**", *IEEE Trans. Systems, Man, Cybernetics*, Vol. SMC-13, pp. 245 - 250, 1983.

[KLE 87] C.A. Klein, B.E. Blaho, "**Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators**", *The International Journal of Robotics Research*, Vol. 6, No. 2, pp. 72 - 83, 1987.

[KOG 94] Y. Koga, K. Kondo, J. Kuffner, J-C. Latombe, "**Planning Motions with Intentions**", *Computer Graphics Proceedings*, Vol. 28, pp. 395 - 408, 1994.

[KOH 89] T. Kohonen, "**Self-Organization and Associative Memory**", Springer Series in Information Sciences, Third Edition, 1989.

[KOM 99] T. Komura, Y. Shinagawa, T.L. Kunii, "**Calculation and Visualization of the Dynamic Ability of the Human Body**", *The Journal of Visualization and Computer Animation*, Vol. 10, No. 2, pp. 57 - 78, 1999.

[KOR 82] J.U. Korein, N. Badler, "**Techniques for Generating the Goal-Directed Motion of Articulated Structures**", *IEEE Computer Graphics and Applications*, pp. 71-81, Nov. 1982.

[KOR 85] J.U. Korein, "**A Geometric Investigation of Reach**", The MIT Press, Cambridge, 1985.

[KRA 92] G. Kramer, "**Solving geometric constraint systems : a case study in kinematics**", MIT Press, 1992.

[KRE 90] E. Kreighbaum, K.M. Barthels, "**Biomechanics**", Third edition, Macmillan Publishing Company, New York, 1990.

[KRO 74] K.H. Kroemer, "**Horizontal push and pull forces**", *Applied Ergonomics*, Vol. 5, No. 2, pp. 94 - 102, 1974.

[KRO 90] K.H. Kroemer, H.J. Kroemer, K.E. Kroemer - Elbert, "**Engineering Physiology**", Second Edition, Van Nostrand Reinhold,1990.

[LAT 91] J.C. Latombe, "**Robot Motion Planning**", Kluwer Academic Publishers, 1991.

[LEB 85] M. Le Borgne, "**Modélisation des robots manipulateurs rigides**", IRISA, Publication No 248, Fév. 1985.

[LEE 00] J. Lee, "**A Hierarchical Approach to Motion Analysis and Synthesis for Articulated Figures**", Ph.D. thesis, Dept. of Computer Science, KAIST, Feb. 2000.

[LEE 90] P. Lee, S. Wei, J. Zhao, N. Badler, "**Strength Guided Motion**", *Computer Graphics*, Vol. 24, No. 4, pp. 253 - 262, Aug. 1990.

[LEP 93] F.X. Lepoutre, "**Human Posture Modelization as a Problem of Inverse Kinematics of Redundant Robots**", *Robotica*, Vol. 11, pp. 339 - 343, 1993.

[LIE 77] A. Liégeois, "**Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms**", *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. SMC-7, No. 12, pp. 868 - 871, 1977.

[LUE 96] H. Luetkepohl, "**Handbook of Matrices**", John Wiley & Sons, 1996.

[MAC 85] A.A. Maciejewski, C.A. Klein, "**Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments**", *The International Journal of Robotics Research*, Vol. 4, No. 3, pp. 109 - 117, 1985.

[MAC 85b] A.A. Maciejewski, C.A. Klein, "**Sam-animation software for simulating articulated motion**", *Computer and Graphics*, Vol. 9, No. 4, pp. 383 - 391, 1985.

[MAC 88] A.A. Maciejewski, C.A. Klein, "**Numerical Filtering for the Operation of Robotic Manipulators through Kinematically Singular Configurations**", *Journal of Robotic Systems*, Vol. 5, No. 6, pp. 527 - 552, 1988.

[MAC 89] A.A. Maciejewski, C.A. Klein, "**The SVD : Computation and Applications to Robotics**", *Int. Journal of Robotic Research*, Vol. 8, No. 6, pp. 63 - 79, 1989.

[MAC 89b] A.A. Maciejewski, "**Kinetic Limitations on the Use of Redundancy in Robotic Manipulators**", *IEEE Conference on Robotics and Automation*, pp. 113 - 118, 1989.

[MAC 90] A.A. Maciejewski, "**Dealing with the Ill-Conditioned Equations of Motion for Articulated Figures**", *IEEE Computer Graphics and Applications*, Vol. 10, No. 3, pp. 63 - 71, May 1990.

[MAD 98] N. Madhavapeddy, S. Ferguson, "**Specialised Constraints for an Inverse Kinematics Animation System Applied to Articulated Figures**", *Proceedings of Eurographics '98*, pp. 215 - 223, 1998.

[MAS 96] R. Mas, "**Balance control of multiple supported articulated systems for computer animation**", PhD Thesis, Univ. de les Illes Balears, Palma de Mallorca, 1996.

[MAU 00] W. Maurel, D. Thalmann, "**Human Upper Limb Modeling including Scapulo-Thoracic Constraint and Joint Sinus Cones**", *Computers & Graphics*, Pergamon Press, Vol. 24, No. 2, pp. 203 - 218, 2000.

[MCG 94] S. McGhee, T.F. Chan, R.V. Dubey, "**Probability-Based Weighting of Performance Criteria for a Redundant Manipulator**", *IEEE Conference*, pp. 1887 - 1894, 1994.

[MCK 96] M. McKenna, D. Zeltzer, "**Dynamic Simulation of a Complex Human Figure Model with Low Level Behavior Control**", *Presence*, Vol. 5, No. 4, pp. 431 - 456, 1996.

[MEN 00] A. Menache, "**Understanding Motion Capture for Computer Animation and Video Games**", Morgan Kaufmann, 2000.

[MOL 99] T. Molet, R. Boulic, D. Thalmann, "**Human Motion Capture Driven by Orientation Measurements**", *Presence*, MIT, Vol. 8, No. 2, pp.187 - 203, 1999.

[MON 00] J.-S. Monzani, P. Baerlocher, R. Boulic, D. Thalmann, "**Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting**", *Eurographics 2000*, Vol. 19, No. 3, pp. 11-19, Interlaken, 2000.

[MON 91] G. Monheit, N. Badler, "**A Kinematic Model of the Human Spine and Torso**", *IEEE Computer Graphics & Applications*, Vol. 11, No. 2, pp. 29 - 38, March 1991.

[MUR 94] R.M. Murray, Z. Li, S.S. Sastry, "**A mathematical introduction to robotic manipulation**", CRC Press, 1994.

[MOO 88] M. Moore, J. Wilhelms, "**Collision detection and Response for Computer Animation**", *Computer Graphics*, Vol. 22, No. 4, Aug. 1988.

[NAK 86] Y. Nakamura, H. Hanafusa, "**Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control**", *Journal of Dynamic Systems, Meas., and Control*, Vol. 108, pp. 163 - 171, Sept. 1986.

[NAK 87] Y. Nakamura, H. Hanafusa, T. Yoshikawa, "**Task-Priority Based Control of Robot Manipulators**", *International Journal of Robotics Research*, Vol. 6, No. 2, pp. 3 - 15, 1987.

[NAS 76] Z. Nashed (ed.), "**Generalized Inverses and Applications**", Academic Press, 1976.

[NEB 99] J.-C. Nebel, "**Keyframe interpolation with self-collision avoidance**", *Computer Animation and Simulation '99*, Springer, pp. 77 - 86, 1999.

[NEL 99] D.D. Nelson, E. Cohen, "**Interactive Mechanical Design Variation for Haptics and CAD**", *Proceedings of Eurographics '99*, Vol. 18, No. 3, pp. 287 - 296, 1999.

[NEU 98] A. Neumaier, "**Solving ill-conditioned and singular linear systems: A tutorial on regularization**", *SIAM Review*, Vol. 40, pp. 636 - 666, 1998.

[OMI 97] O. Omidvar, P. Van der Smagt (eds.), "**Neural systems for robotics**", Academic Press, 1997.

[ORT 70] J.M. Ortega, W.C. Rheinboldt, "**Iterative solution of nonlinear equations in several variables**", New York, Academic Press, 1970.

[OSY 84] A. Osyczka, "**Multicriterion Optimization in Engineering**", Ellis Horwood Series in Engineering Science, 1984.

[PAR 87] P.M. Pardalos, J. Ben Rosen, "**Constrained global optimization: algorithms and applications**", Springer-Verlag, 1987.

[PAU 81] R. Paul, "**Robot Manipulators: Mathematics, Programming and Control**", MIT Press, 1981.

[PHI 90] C.B. Phillips, J. Zhao, N. Badler, "**Interactive Real-Time Articulated Figure Manipulation using Multiple Kinematic Constraints**", *SIGGRAPH 90 Course notes (Human figure animation : approaches and applications)*, 1990.

[PHI 91] C.B. Phillips, N. Badler, "**Interactive Behaviors for Bipedal Articulated Figures**", *Computer Graphics*, Vol. 25, No. 4, pp. 359 - 362, July 1991.

[PRE 92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "**Numerical Recipes in C**", Second edition, Cambridge University Press, 1992.

[RIJ 91] H. Rijpkema, M. Girard, "**Computer Animation of Knowledge-Based Human Grasping**", *SIGGRAPH 91 Proceedings*, pp. 339 - 348, 1991.

[ROB 95] T. Roberts, "**Understanding Balance - The Mechanics of Posture and Locomotion**", Chapman & Hall, 1995.

[ROS 60] J.B. Rosen, "**The Gradient Projection Method for Nonlinear Programming. Part I: Linear Constraints**", *Journal of Society of Industrial Applied Mathematics (SIAM)*, Vol. 8, pp. 181 - 217, Dec. 1960.

[ROS 61] J.B. Rosen, "**The Gradient Projection Method for Nonlinear Programming. Part II: Nonlinear Constraints**", *Journal of Society of Industrial Applied Mathematics (SIAM)*, Vol. 9, No. 4, Dec. 1961.

[SCH 72] F.T. Schanne, "**Three Dimensional Hand Force Capability Model for a Seated Person**", PhD thesis, Univ. of Michigan (Ann Arbor, MI), 1972.

[SCH 97] F. Scheeppers, R.E. Parent, W.E. Carlson, S.F. May, "**Anatomy-Based Modeling of the Human Musculature**", *Computer Graphics Proceedings*, pp. 163 - 172, 1997.

[SCH 85] K. Schoemake, "**Animating Rotation with Quaternion Curves**", *Computer Graphics*, Vol. 19, No. 3, pp. 245 - 254, 1985.

[SCI 88] L. Sciavicco, B. Siciliano, "**A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulators**", *IEEE Journal of Robotics and Automation*, Vol. 4, pp. 403 - 410, 1988.

[SIC 91] B. Siciliano, J-J. Slotine, "**A General Framework for Managing Multiple Tasks in Highly Redundant Robotic Systems**", *ICAR '91*, pp. 1211 - 1216, 1991.

[SIM 88] K. Sims, D. Zeltzer, "**A Figure Editor and Gait Controller for Task Level Animation**", *SIGGRAPH Course notes (Synthetic Actors: the Impact of Artificial Intelligence and Robotics on Animation)*, 1988.

[SIM 94] K. Sims, "**Evolving Virtual Creatures**", *Computer Graphics Proceedings*, SIGGRAPH, 1994.

[STR 72] Strauss W.L., "**The human figure by Albrecht Dürer**", Dover Publications, New York, 1972.

[SUT 63] L. Sutherland, "**Sketchpad - a man-machine graphical communication system**", Ph.D. Thesis, MIT, 1963.

[TIK 63] A.N. Tikhonov, "**Solution of incorrectly formulated problems and the regularization method**", *Soviet Mah. Dokl.*, Vol. 4, pp. 1035 - 1038, 1963.

[TOL 96] D. Tolani, N. Badler, "**Real-Time Inverse Kinematics of the Human Arm**", *Presence*, Vol. 5, No. 4, pp. 393 - 401, 1996.

[TOL 00] D. Tolani, A. Goswami, N. Badler, "**Real-time inverse kinematics techniques for anthropomorphic arms**", *Graphical Models*, Vol. 62, pp. 353 - 388, 2000.

[VAN 92] C.W.A.M. Van Overveld, E. Van Loon, "**Hanging Cloths and Dangling Rods: a Unified Approach to Constraints in Computer Animation**", *The Journal of Visualization and Computer Animation*, Vol. 3, No. 1, pp. 45 - 59, 1992.

[WAL 75] G.R. Walsh, "**Methods of Optimization**", John Wiley and Sons, 1975.

[WAL 88] I.D. Walker, S.I. Marcus, "**Subtask Performance by Redundancy Resolution for Redundant Robot Manipulators**", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 3, June 1988.

[WAM 86] C.W. Wampler, "**Manipulator inverse kinematic solutions based on vector formulations and damped least-squares method**", *IEEE Trans. Syst., Man, Cybernetics*, Vol. 14, pp. 93 - 101, 1986.

[WAN 98] X. Wang, J.-P. Verriest, "**A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation**", *The Journal of Visualization and Computer Animation*, Vol. 9, pp. 33 - 47, 1998.

[WAN 98b] X.G. Wang, F. Mazet, N. Maia, K. Voinot, J.P. Verriest, M. Fayet, "**Three-dimensional modelling of the motion range of axial rotation of the upper arm**", *Journal of biomechanics*, Vol. 31, No. 10, pp. 899 - 908, 1998.

[WAT 92] A. Watt, M. Watt, "**Advanced Animation and Rendering Techniques**", Addison-Wesley, ACM Press, 1992.

[WEL 93] C. Welman, "**Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation**", Master Thesis, Simon Fraser University, 1993.

[WHI 69] D.E. Whitney, "**Resolved Motion Rate Control of Manipulators and Human Prostheses**", *IEEE Trans. on Man-Machine Systems*, Vol. MMS-10, No. 2, pp. 47 - 53, 1969.

[WHI 90] A.A. White, M.M. Panjabi, "**Clinical Biomechanics of the Spine**", J.B. Lippincott Company, Second Edition, 1990.

[WIL 97] D.J. Wiley, J.K. Hahn, "**Interpolation Synthesis of Articulated Figure Motion**", *IEEE Computer Graphics & Applications*, Vol. 17, No. 6, Nov./Dec. 1997.

[WIT 77] J. Wittenburg, "**Dynamics of systems of rigid bodies**", Teubner, Stuttgart, 1977.

[WIT 88] A. Witkin, M. Kass, "**Spacetime Constraints**", *SIGGRAPH '88*, pp. 159 - 168, 1988.

[WOL 84] W.A. Wolovich, H. Elliot, "**A computational technique for inverse kinematics**", *Proc. of 32nd Conference on Decision and Control*, pp. 1359 - 1362, Dec. 1984.

[ZHA 89] J. Zhao, N. Badler, "**Real Time Inverse Kinematics with Joint Limits and Spatial Constraints**", Technical Report, University of Pennsylvania, 1989.

[ZHA 94] J. Zhao, N. Badler, "**Inverse Kinematics Positioning using Nonlinear Programming for Highly Articulated Figures**", *ACM Transactions on Graphics*, Vol. 13, No. 4, pp. 313 - 336, Oct. 1994.

[ZHA 94b] X. Zhao, N. Badler, "**Interactive Body Awareness**", *Computer Aided Design*, Vol. 26, No. 12, pp. 861 - 868, Dec. 1994.

Curriculum Vitae

General informations

Name	Paolo Baerlocher
Date of birth	2 october 1971 in Locarno (TI), Switzerland
Nationality	Swiss and French
Mother tongues	French and Italian
Other languages	Good knowledge of English and German

Education

1986 - 1990

Maturity type C (scientific), Liceo of Locarno

1990 - 1995

Engineering diploma in Computer Science (EPFL, Lausanne)

Diploma project: a physically-based simulator of 3D articulated figures

Professional activities

1990 - 1991

Development of two commercial video games for the Archimedes microcomputer.

1995 - 1996

Participation to the development of a Formula One video game for PC, at Visiware (Paris).

1996 - 2001

Research assistant at the Computer Graphics Lab (EPFL, Lausanne)

Publications

- P. Baerlocher, R. Boulic, “**Task-Priority Formulations for the Kinematic Control of Highly Redundant Articulated Structures**”, Proceedings of IEEE Conference “Intelligent Robots and Systems” IROS 98, Victoria (Canada), pp. 323 - 329, Oct. 1998.
- P. Baerlocher, R. Boulic, “**Kinematic Control of the Mass Properties of Redundant Articulated Bodies**”, Proceedings of IEEE Conference on Robotics and Automation, ICRA 2000, San Francisco (CA), pp. 2557 - 2562, Apr. 2000.
- J.-S. Monzani, P. Baerlocher, R. Boulic, D. Thalmann, “**Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting**”, Eurographics 2000, Vol. 19, No. 3, pp. 11-19, Interlaken, 2000.
- R. Boulic, P. Baerlocher, “**Enforcing Multiple Levels of Kinematic Control for Human Posture Optimization**“, CD ROM Proceedings, Journée spécialisée “Les modèles numériques de l’homme pour la conception de produits”, INRETS, Lyon, March 2000.
- P. Baerlocher, R. Boulic, “**Parametrization and range of motion of the ball-and-socket joint**”, AVATARS 2000 Conference, Lausanne, Nov. 2000.
- P. Baerlocher, R. Boulic, “**Combining Multiple Tasks with Different Priorities for Interactive Human Posture Control**”, GTEC 2001 Conference on Game Technology, Hong Kong, Jan. 2001.
- R. Boulic, P. Baerlocher, “**Cinématique inverse pour personnages 3D: solutions analytiques et variationnelles**”, Revue de CFAO, Numéro spécial AFIG, 2001.