Graphical Models 73 (2011) 243–260

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

FABRIK: A fast, iterative solver for the Inverse Kinematics problem ☆

Andreas Aristidou *, Joan Lasenby

Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK

ARTICLE INFO

Article history:
Received 12 February 2010
Received in revised form 24 March 2011
Accepted 9 May 2011
Available online 15 May 2011

Keywords:
Human animation
Inverse Kinematics
Joint configuration
Motion reconstruction

ABSTRACT

Inverse Kinematics is defined as the problem of determining a set of appropriate joint configurations for which the end effectors move to desired positions as smoothly, rapidly, and as accurately as possible. However, many of the currently available methods suffer from high computational cost and production of unrealistic poses. In this paper, a novel heuristic method, called Forward And Backward Reaching Inverse Kinematics (FABRIK), is described and compared with some of the most popular existing methods regarding reliability, computational cost and conversion criteria. FABRIK avoids the use of rotational angles or matrices, and instead finds each joint position via locating a point on a line. Thus, it converges in few iterations, has low computational cost and produces visually realistic poses. Constraints can easily be incorporated within FABRIK and multiple chains with multiple end effectors are also supported.

1. Introduction

This paper addresses the problem of manipulating articulated figures in an interactive and intuitive fashion for the design and control of their posture. This problem finds its application in the areas of robotics, computer animation, ergonomics and gaming. In computer graphics, articulated figures are a convenient model for humans, animals or other virtual creatures from films and video games. Inverse Kinematics (IK) has also been used in rehabilitation medicine in order to observe asymmetries or abnormalities. The most popular method for animating such models is motion-capture; however, despite the availability of highly sophisticated techniques and expensive tools, many problems appear when dealing with complex figures. Most virtual character models are very complex; they are made up of many joints giving a system with a large number of degrees of freedom, thus, it is often difficult to produce a realistic character animation.

Inverse Kinematics is a method for computing the posture via estimating each individual degree of freedom in order to satisfy a given task that meets user constraints; it plays an important role in the computer animation and simulation of articulated figures. This paper presents a new heuristic iterative method, Forward And Backward Reaching Inverse Kinematics (FABRIK), for solving the IK problem in different scenarios. FABRIK uses a forward and backward iterative approach, finding each joint position via locating a point on line. FABRIK has been utilised in highly complex systems with single and multiple targets, with and without joint restrictions. It can easily handle end effector orientations and support, to the best of our knowledge, all chain classes. A reliable method for incorporating constraints is also presented and utilised within FABRIK. The proposed method retains all the advantages of FABRIK, producing visually smooth movements without oscillations and discontinuities. Several experiments have been implemented for comparison purposes between the most popular manipulator solvers, including multiple end effectors with multiple tasks, and highly constrained joints. The proposed algorithm is very efficient both in simple and complex problems resulting in similar or even better poses than highly sophisticated methods, requiring

☆ This paper has been recommended for acceptance by Jarek Rossignac.

* Corresponding author. Fax: +44 1223332662.
E-mail address: aa462@cam.ac.uk (A. Aristidou).

1524-0703/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2011.05.003

less processing time and fewer iterations to reach the target. Another important advantage of the proposed methodology is the simplicity of the algorithm, which enables easy configuration to any IK problem.

2. Background and motivation

The production of realistic and plausible motions remains an open challenge within the robotics and animation communities. Several models have been implemented for solving the IK problem from many different areas of study. Zhao and Badler [1] poses the IK task as a problem of finding a local minimum of a set of non-linear equations, defining Cartesian space constraints. However, the most popular numerical approach is to use the Jacobian matrix to find a linear approximation to the IK problem. The *Jacobian solutions* linearly model the end effectors' movements relative to instantaneous system changes in link translation and joint angle. Several different methodologies have been presented for calculating or approximating the Jacobian inverse, such as the Jacobian Transpose, Damped Least Squares (DLS), Damped Least Squares with Singular Value Decomposition (SVD-DLS), Selectively Damped Least Squares (SDLS) and several extensions [2–7]. Jacobian inverse solutions produce smooth postures; however most of these approaches suffer from high computational cost, complex matrix calculations and singularity problems. An alternative approach is given by Pechev in [8] where the Inverse Kinematics problem is solved from a control prospective. This approach is computationally more efficient than the pseudo-inverse based methods and does not suffer from singularity problems.

The second family of IK solvers is based on Newton methods. These algorithms seek target configurations which are posed as solutions to a minimisation problem, hence they return smooth motion without erratic discontinuities. The most well known methods are Broyden's method, Powell's method and the Broyden, Fletcher, Goldfarb and Shanno (BFGS) method [9]. However, the Newton methods are complex, difficult to implement and have high computational cost per iteration.

A very popular IK method is the Cyclic Coordinate Descent (CCD) algorithm, which was first introduced by Wang and Chen [10] and then biomechanically constrained by Welman [11]. CCD has been extensively used in the computer games industry [12] and has recently been adapted for protein structure prediction [13]. CCD is a heuristic iterative method with low computational cost for each joint per iteration, which can solve the IK problem without matrix manipulations; thus it formulates a solution very quickly. However, CCD has some disadvantages; it can suffer from unrealistic animation, even if manipulator constraints have been added, and often produces motion with erratic discontinuities. It is designed to handle serial chains, thus, it is difficult to extend to problems with multiple end effectors. Unzueta et al. [14] describes a Sequential IK (SIK) solver, and is a direct extension of Boulic et al. [15], in that its inputs are end effector positions, such as wrists, ankles, head and pelvis, which are used to find the human pose. The IK problem is then solved sequentially using simple analytic-iterative IK algorithms (CCD), in different parts of the body, in a specific order. Kulpa and Multon [16] also adopted the CCD kinematic algorithm and solved its crucial problem of resulting unnatural poses. The proposed extension in [16] is able to solve problems with humanoid hierarchy, dividing the whole body into groups of joints near an end effector (typically head, trunk, arms and legs). In order to satisfy the desired centre of mass, the lightest group moves first, adjusting its centre of mass by changing the length of the limb and rotating it (assuming it as a rigid body).

Recently, Courty and co-workers [17,18] proposed a Sequential Monte Carlo Method (SMCM) and particle filtering approach respectively. The proposed particle IK solver treats the character skeleton as a set of 3 DoFs particles having inter-length constraints. An iterative constrainer, with various pre-conditions and parameters, is then applied over the particles, tuning its behaviour both statically and dynamically. The final particle positions and the length constraints are then used to reconstruct the resulting DoFs of the body. Neither method suffers from matrix singularity problems and both perform reasonably well. However, these statistical methods have high computational cost. Grochow et al. [19] presents a style-based IK method which is based on a learned model of human poses. Given a set of constraints, the proposed system can produce, in real-time, the most likely pose satisfying those constraints. The model has been trained on different input data that leads to different styles of IK; it can generate any pose, but poses are highly related to those which are most similar to the space of poses in the training data. Sumner and co-workers [20,21] used mesh-based IK techniques to configure the animated shapes. Mesh-based IK learns a space of natural deformations from example meshes. Using the learned space, they generate new shapes that respect the deformations exhibited by the examples, and satisfy vertex constraints imposed by the user. However, these methods require an off-line training procedure, their results are highly dependent on the training data and limited only to those models and movements on which the system has been trained.

The Triangulation algorithm [22] is an IK solver that uses the cosine rule to calculate each joint angle, starting at the root of the kinematic chain and moving outward towards the end effector. Although it can reach the target in just one iteration, having low computational cost, its results are often visually unnatural. The joints close to the end effector are usually in a straight line, emphasising the rotation on the joints neighbouring the root. The Triangulation IK method can only be applied to problems with a single end effector and does not support imposed joint limits. An improved version is given in [23] where the *n*-link IK problem is reduced to a *two*-link problem in which each link is rotated at most once in an attempt to reach the target position.

Brown et al. [24] presents a real-time method which uses a 'Follow-the-Leader' (FTL) non-iterative technique which is similar to each individual iteration of FABRIK. While FTL was specifically designed for rope simulation, it can be applied to manipulate kinematic chains (ball-and-socket joints connected by rigid links). Since FTL does

not work in an iterative fashion, the authors cope with node constraints (for example the root position is fixed, as in the IK definition) by averaging the results of FTL (intermediate nodes) in each direction. However, the averaging of the results induces a variation of the segment lengths and produces, in some cases, unnatural poses. Although similar to FABRIK in its basic structure, the FTL algorithm has not been extended to support joint constraints and orientations (these are largely superfluous in rope simulation), nor has it been applied to cases where multiple end effectors exist.

3. The articulated body model

A rigid multibody system consists of a set of rigid objects, called *links*, connected together by *joints*. A *joint* is the component concerned with motion; it permits some degree of relative motion between the connected segments. Virtual body modelling is important for human posture control. A well constrained model can restrict postures to a feasible set, therefore allowing a more realistic motion. Most models assume that body parts are rigid, although this is just an assumption approximating reality. The skeletal structure is usually modeled as a hierarchy of rigid segments connected by joints, each defined by their length, shape, volume and mass properties.

A manipulator such as a robot arm or an animated graphics character is modelled as a *chain* composed of rigid *links* connected at their ends by rotating *joints*. Any translation and/or rotation of the i-th joint affects the translation and rotation of any joint placed later in the chain. The chains can be formalised as follows: All joints with no children are marked as *end effectors*; a chain can be built for each end effector by moving back through the skeleton, going from parent to parent, until the root (the start of the chain) is reached. By definition, in the IK problem, the root joint is assumed fixed but methods can generally cope with translation of the root.

Algorithm 1. A full iteration of the FABRIK algorithm

Input: The joint positions $\mathbf{p}_i$ for $i = 1,\ldots,n$, the target position $\mathbf{t}$ and the distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$ for $i = 1,\ldots,n-1$.
Output: The new joint positions $\mathbf{p}_i$ for $i = 1,\ldots,n$.

1.1 *% The distance between root and target*
1.2 $dist = |\mathbf{p}_1 - \mathbf{t}|$
1.3 *% Check whether the target is within reach*
1.4 **if** $dist > d_1 + d_2 + \ldots + d_{n-1}$ **then**
1.5 *% The target is unreachable*
1.6 **for** $i = 1,\ldots,n-1$ **do**
1.7 *% Find the distance r_i between the target $\mathbf{t}$ and the joint position $\mathbf{p}_i$*
1.8 $r_i = |\mathbf{t} - \mathbf{p}_i|$
1.9 $\lambda_i = d_i / r_i$
1.10 *% Find the new joint positions $\mathbf{p}_i$.*
1.11 $\mathbf{p}_{i+1} = (1 - \lambda_i)\,\mathbf{p}_i + \lambda_i \mathbf{t}$
1.12 **end**
1.13 **else**
1.14 *% The target is reachable; thus, set as $\mathbf{b}$ the initial position of the joint $\mathbf{p}_1$*
1.15 $\mathbf{b} = \mathbf{p}_1$
1.16 *% Check whether the distance between the end effector $\mathbf{p}_n$ and the target $\mathbf{t}$ is greater than a tolerance.*
1.17 $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.18 **while** $dif_A > tol$ **do**
1.19 *% STAGE 1: FORWARD REACHING*
1.20 *% Set the end effector $\mathbf{p}_n$ as target $\mathbf{t}$*
1.21 $\mathbf{p}_n = \mathbf{t}$
1.22 **for** $i = n-1,\ldots,1$ **do**
1.23 *% Find the distance r_i between the new joint position $\mathbf{p}_{i+1}$ and the joint $\mathbf{p}_i$*
1.24 $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.25 $\lambda_i = d_i / r_i$
1.26 *% Find the new joint positions $\mathbf{p}_i$.*
1.27 $\mathbf{p}_i = (1 - \lambda_i)\,\mathbf{p}_{i+1} + \lambda_i \mathbf{p}_i$
1.28 **end**
1.29 *% STAGE 2: BACKWARD REACHING*
1.30 *% Set the root $\mathbf{p}_1$ its initial position.*
1.31 $\mathbf{p}_1 = \mathbf{b}$
1.32 **for** $i = 1,\ldots,n-1$ **do**
1.33 *% Find the distance r_i between the new joint position $\mathbf{p}_i$ and the joint $\mathbf{p}_{i+1}$*
1.34 $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.35 $\lambda_i = d_i / r_i$
1.36 *% Find the new joint positions $\mathbf{p}_i$.*
1.37 $\mathbf{p}_{i+1} = (1 - \lambda_i)\mathbf{p}_i + \lambda_i \mathbf{p}_{i+1}$
1.38 **end**
1.39 $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.40 **end**
1.41 **end**

4. FABRIK: a new heuristic IK solution

In this section, a new heuristic method for solving the IK problem, FABRIK, is presented. It uses the previously calculated positions of the joints to find the updates in a forward and backward iterative mode. FABRIK involves minimising the system error by adjusting each joint angle one at a time. The proposed method starts from the last joint of the chain and works forwards, adjusting each joint along the way. Thereafter, it works backward in the same way, in order to complete a full iteration. This method, instead of using angle rotations, treats finding the joint locations as a problem of finding a point on a line; hence, time and computation can be saved.

Assume $\mathbf{p}_1,\ldots,\mathbf{p}_n$ are the joint positions of a manipulator. Also, assume that $\mathbf{p}_1$ is the root joint and $\mathbf{p}_n$ is the end

effector, for the simple case where only a single end effector exists. The target is symbolised as **t** and the initial base position by **b**. FABRIK is illustrated in pseudo-code in Algorithm 1 and a graphical representation of a full iteration with a single target and 4 joints is presented and explained in Fig. 1.

First calculate the distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$, for $i = 1,\ldots,n-1$. Then, check whether the target is reachable or not; find the distance between the root and the target, *dist*, and if this distance is smaller than the total sum of all the inter-joint distances, $dist < \sum_1^{n-1} d_i$, the target is within reach, otherwise, it is unreachable. If the target is within reach, a full iteration is constituted by two stages. In the first stage, the algorithm estimates each joint position starting from the end effector, $\mathbf{p}_n$, moving inwards to the manipulator base, $\mathbf{p}_1$. So, let the new position of the end effector be the target position, $\mathbf{p}'_n = \mathbf{t}$. Find the line, l_{n-1}, which passes through the joint positions $\mathbf{p}_{n-1}$ and $\mathbf{p}'_n$. The new position of the $(n-1)^{th}$ joint, $\mathbf{p}'_{n-1}$, lies on that line with distance d_{n-1} from $\mathbf{p}'_n$. Similarly, the new position of the $(n-2)^{th}$ joint, $\mathbf{p}'_{n-2}$, can be calculated using the line l_{n-2}, which passes through the $\mathbf{p}_{n-2}$ and $\mathbf{p}'_{n-1}$, and has distance d_{n-2} from $\mathbf{p}'_{n-1}$. The algorithm continues until all new joint positions are calculated, including the root, $\mathbf{p}'_1$.

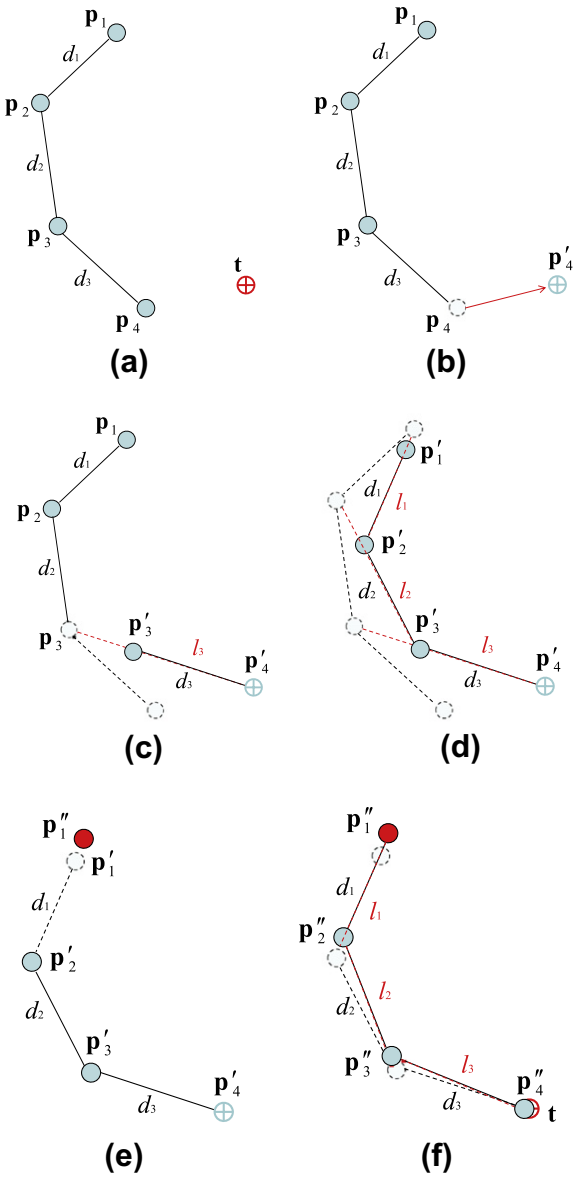

Fig. 1. An example of a full iteration of FABRIK for the case of a single target and 4 manipulator joints. (a) The initial position of the manipulator and the target, (b) move the end effector $\mathbf{p}_4$ to the target, (c) find the joint $\mathbf{p}'_3$ which lies on the line l_3 that passes through the points $\mathbf{p}'_4$ and $\mathbf{p}_3$, and has distance d_3 from the joint $\mathbf{p}'_4$, (d) continue the algorithm for the rest of the joints, (e) the second stage of the algorithm: move the root joint $\mathbf{p}'_1$ to its initial position, (f) repeat the same procedure but this time start from the base and move outwards to the end effector. The algorithm is repeated until the position of the end effector reaches the target or gets sufficiently close.

Having in mind that the new position of the manipulator base, $\mathbf{p}'_1$, should not be different from its initial position, a second stage of the algorithm is needed. A full iteration is completed when the same procedure is repeated but this time starting from the root joint and moving outwards to the end effector. Thus, let the new position for the 1^{st} joint, $\mathbf{p}''_1$, be its initial position **b**. Then, using the line l_1 that passes through the points $\mathbf{p}''_1$ and $\mathbf{p}'_2$, we define the new position of the joint $\mathbf{p}''_2$ as the point on that line with distance d_1 from $\mathbf{p}''_1$. This procedure is repeated for all the remaining joints, including the end effector. In cases where the root joint has to be translated to a desired position, FABRIK works as described with the difference that in the backward phase of the algorithm, the new position of the root joint, $\mathbf{p}''_1$, will be the desired and not the initial position.

After one complete iteration, it is almost always the case (observed empirically) that the end effector is closer to the target. The procedure is then repeated, for as many iterations as needed, until the end effector is identical or close enough (to be defined) to the desired target. The unconstrained version of FABRIK converges to any given chains/goal positions, when the total length of serial links is greater than the distance to the target (the target is reachable). However, if the target is not within the reachable area, there is a termination condition which compares the previous and the current position of the end effector, and if this distance is less than an indicated tolerance, FABRIK terminates its operation. Also, in the extreme case where the number of iterations has exceeded an indicated value and the target has not been reached, the algorithm is terminated (however, we have never encountered such a situation).

Several optimisations can be achieved using Conformal Geometric Algebra (CGA) [25,26] to produce faster results and to converge to the final answer in fewer iterations; CGA has the advantage that basic entities, such as spheres, lines, planes and circles, are simply represented by algebraic objects. Therefore, a direct estimate of a missing joint, when it is between 2 true positions, can be achieved by intersecting 2 spheres with centres the true joint positions and radii the distances between the estimated and the true joints respectively; the new joint position will be taken as the point on the circle (created by the intersection of the 2 spheres) nearest to the previous joint position. Another simple optimisation is the direct construction of a line pointing towards the target, when the latter is unreachable.

The proposed method has all the advantages of existing iterative heuristic algorithms. The computational cost for each joint per iteration is low, meaning the solution is arrived at very quickly. It is also easy to implement, since it is simply a problem involving points, distances and lines and always returns a solution when the target is in range. It does not require complex calculations (e.g Jacobian or Hessian matrix) or matrix manipulations (inversion or singular value decomposition), it does not suffer from singularity problems and returns smooth motion without erratic discontinuities.

4.1. FABRIK with multiple end effectors

In reality, most of the multibody models, such as hands, human or legged bodies etc, are comprised of several kinematic chains, and each chain generally has more than 1 end effector. Therefore, it is essential for an IK solver to be able to solve problems with multiple end effectors and targets. The proposed algorithm can be easily extended to process models with multiple end effectors. However, prior knowledge of the model, such as the sub-base[1] joints, and the number and structure of chains is needed.

The algorithm is divided into two stages, as in the single end effector case. In the first stage, the normal algorithm is applied but this time starting from each end effector and moving inwards to the parent sub-base. This will produce as many different positions of the sub-base as the number of end effectors connected with that specific sub-base. The new position of the sub-base will then be the centroid of all these positions. Thereafter, the normal algorithm should be applied inwards starting from the sub-base to the manipulator root. If there are more intermediate sub-bases, the same technique should be used. In the second stage, the normal algorithm is applied starting now from the root and moving outwards to the sub-base. Then, the algorithm should be applied separately for each chain until the end effector is reached; if more sub-bases exist, the same process is applied. The method is repeated until all end effectors reach the targets or there is no significant change between their previous and their new positions. An example of a model figure having multiple end effectors and sub-bases is presented in Fig. 2.

More sophisticated (and complex) models can be also tackled. Extending the proposed algorithm to take into account the figure's shape, constraints and properties, will reduce the number of iterations needed to reach the target and will return more feasible postures. For example, FABRIK has been successfully applied to real-time hand tracking and reconstruction in motion capture [27,28].

4.2. FABRIK within closed loops

FABRIK can also cope with cases where the "end effector" is not positioned at the end of the chain (i.e. it is a leaf) in the same way as for the sub-bases described in Section 4.1. The whole model could be divided into groups of joints near the end effectors (such as head, trunk, arms and legs) and then the body postures can be sequentially adapted in a specific order, similarly to Unzueta et al. [14] and Kulpa and Multon [16]. Obviously, the adaption hierarchy varies between models. FABRIK has been successfully used within closed loops of a humanoid, achieving real-time centre of rotation correction in motion capture, under marker occlusions [29].

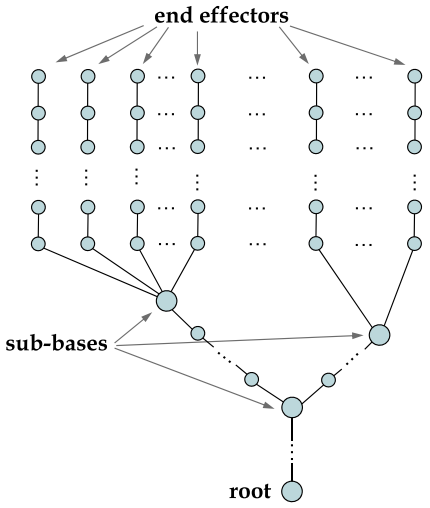

Fig. 2. An example of a model figure with multiple end effectors and multiple sub-bases.

4.3. Applying joint limits to FABRIK

Most legged body models are comprised of joints having biomechanical constraints, which provide natural restrictions on their motion. Such constraints are essential in physical simulations, IK techniques and tracking in motion capture systems to reduce visually unrealistic movements.

Several biomechanically and anatomically correct models have been presented that formalise the range of motion of an articulated figure. These models are mainly characterised by the number of parameters which describe the motion space and are hierarchically structured. Because of their complex nature, most of the proposed models are simplified or approximated by more than one joint. The most well-known models are: *the shoulder model*, a complex model composed of 3 different joints [30–33]; *the spine model*, a complex arrangement of 24 vertebrae (usually, for simplicity, the spine is modelled as a simple chain of joints [34–37]); *the hand model*, this is the most versatile part of the body comprising a large number of joints [38–40]; *the strength model*, which takes account of the forces applied from the skeletal muscles to the bones [34].

A joint is defined by its position and orientation and, in the most general case, has 3 DoFs. A bone rotation can be described by factoring it into two rotations: one "simple rotation", termed here *rotational* (2 DoFs), that moves the bone to its final direction vector, and another which we call

[1] A sub-base joint is a joint which connects 2 or more chains. A pre-analysis of the body can determine exactly where the sub-bases are located.

orientational (1 DoF), which represents the twist around this final vector. Thus, the range of movement of a bone can be controlled by dividing the joint restriction procedure into two interconnected phases, a rotational and an orientational phase, contributing equally to the joint restrictions. The essential feature of a joint is that it permits a relative motion between the two limbs it connects. Most of the existing structure models, such as those described above, use techniques which restrict the bone to lie within the rotational and orientational limits of the joint. Blow [41] proposes a loop hung in space, limiting the range of motion of the bone to "reach windows" described by star polygons. Wilhelms and Van Gelder [42] present a 3D "reach cone" methodology using planes, treating the joint limits in the same way as [41]. Korein and co-workers [36,43] parameterise realistic joint boundaries of the ball-and-socket joint by decomposing the arbitrary orientation into two components and controlling the rotational joint limits so they do not exceed their bounds. Once a proper parametrisation is defined for each joint of the articulated body, an animation engine is utilised.

Tolani et al. [44] presents analytical and numerical constraints suitable for anthropomorphic limbs; they treat the limbs of 3D characters independently in closed forms resulting in fast analytical solutions. However, analytic solutions, in general, lack flexibility for under-constrained instances. A pin-and-drag interface for articulated characters is presented in [45], where multiple-priority-levels architectures for combining end effector and centre of mass position control is illustrated.

In this section, a reliable methodology for incorporating manipulator constraints is described using FABRIK. Since FABRIK is iterative, the joint restrictions can be enforced at each step just by taking the resultant orientation and forcing it to stay within the valid range. FABRIK's ability to converge on an answer, if the target is within reach, is not affected by any imposed joint limits.

The main idea behind this methodology is the re-positioning and re-orientation of the target to be within the allowed range bounds; ensuring that these restrictions are always satisfied means a more feasible posture can be achieved. This can be accomplished by checking if the target is within the valid bounds, at each step of FABRIK, and if it is not, to guarantee that it will be moved accordingly. In contrast to most existing techniques for joint constraints, the proposed methodology simplifies the 3D problem into a 2D problem, meaning that the complexity and the required processing time is reduced. In this section, a joint restriction methodology is presented for the general case of a ball and socket joint; this example should be considered as an illustration of how joint or model constraints can be incorporated within FABRIK. Similar techniques can be easily adopted to limit different joint models.

Assume we have a ball-and-socket joint with orientational limits described by the rotor **R** and rotational limits described by the angles $\theta_1,\ldots,\theta_4$. A graphical representation of a joint limit boundary could be an irregular cone which is defined by these angles. The rotational limits are enforced by re-positioning the target point as the nearest

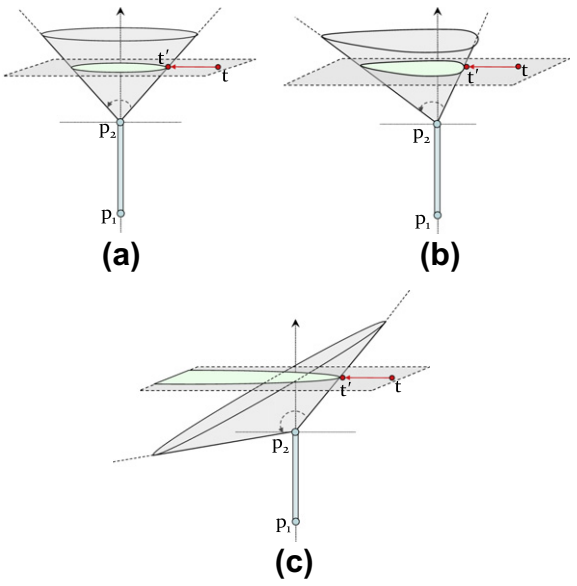

Fig. 3. The target is re-positioned within the allowed range of motion which is defined by the conic section. There are 3 types of joint restriction, as described by the angles $\theta_1,\ldots,\theta_4$: (a) a circle, (b) an ellipsoidal shape and (c) a parabolic shape.

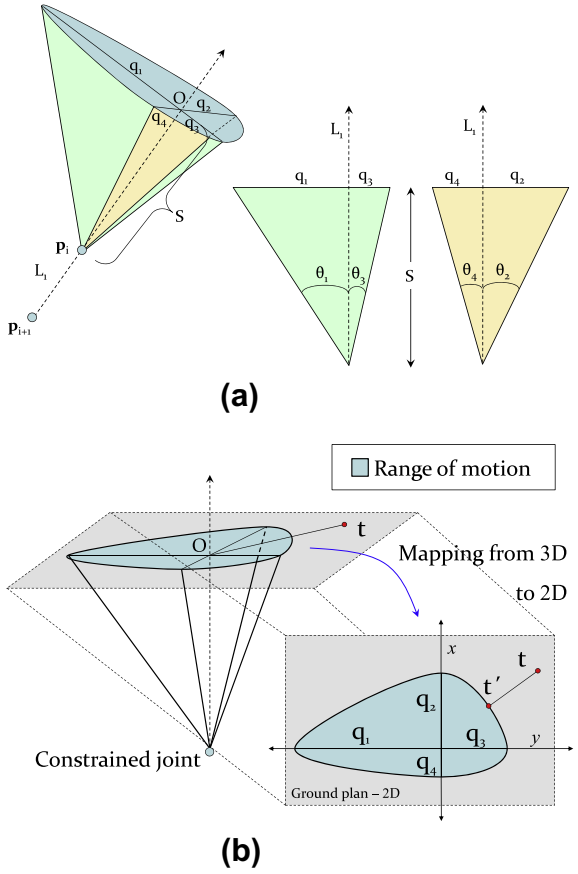

Fig. 4. (a) The ball-and-socket joint, $\mathbf{p}_i$, with its associated irregular cone which defines the allowed range of motion. (b) Shows the composite ellipsoidal shape created by the distances q_j mapped from 3D to 2D.

point on a conic section from the target position; this procedure is described in detail later. There are 3 possible conic sections, according to the angles defining the irregular cone: if all θs are equal, the conic section is a circle; if all θs are greater or smaller than 90° and are not equal, the conic section has an ellipsoidal shape; finally, if there are θs both greater and smaller than 90°, then the joint boundary limits are defined by a parabolic shape, as illustrated in Fig. 3. In subsequent analysis, the joint limits are assumed to be defined by an ellipsoidal shape, since this is the most common case, but similar procedures apply for different conic sections. Fig. 4 gives a graphical representation of the implemented constraints and the irregular cone describing the rotational motion bounds for the case of an ellipsoidal shape.

The orientation of the joint can be assigned as follows: Assume we are in the first stage of the algorithm, i.e. we have just calculated the new position of joint $\mathbf{p}'_i$, and we want to find the new position of the $(i-1)^{th}$ joint, $\mathbf{p}'_{i-1}$. Find the rotor expressing the rotation between the orientation frames at joints $\mathbf{p}'_i$ and $\mathbf{p}_{i-1}$ and if this rotor represents a rotation greater than a limit, reorient the joint $\mathbf{p}_{i-1}$ in such a way that the rotation will be within the limits. Repeat the procedure for all the joints on both stages of the algorithm. The methodology is also described in pseudocode in Algorithm 2.

Algorithm 2. The orientational constraints

Input: The rotor **R** expressing the rotation between the
    orientation frames at joints $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$.
Output: The new re-oriented joint $\mathbf{p}'_{i-1}$.
2.1 Check whether the rotor **R** is within the motion range
bounds
2.2 **if** within the bounds **then**
2.3   do nothing and exit
2.4 **else**
2.5   reorient the joint $\mathbf{p}_{i-1}$ in such a way that the rotor will
    be within the limits
2.6 **end**

Once the joint orientation is established, the rotational (2 DoFs) limits, described by angles $\theta_1,\ldots,\theta_4$, can be applied as follows. Firstly, we find the projection O of the target **t** on line L_1, where L_1 is the line passing through the joint under consideration, $\mathbf{p}_i$, and the previous joint of the chain, $\mathbf{p}_{i+1}$. Then determine the distance S from the point O to the joint position $\mathbf{p}_i$ and calculate the distances $q_j = S\tan(\theta_j)$, for $j = 1,\ldots,4$, as shown in Fig. 4. We then apply a rotation and translation which takes O to the origin and the axes defining the constraints to the x and y axes, as in Fig. 4(b). Working in this 2D plane, we locate the target in a particular quadrant and find the ellipse defined on that quadrant using the associated distances q_j; for example, in Fig. 4(b) we are working with the ellipse which is defined by the angles θ_2 and θ_3 (or the distances q_2 and q_3). Finally, find the nearest point on that ellipse from the target, if the latter

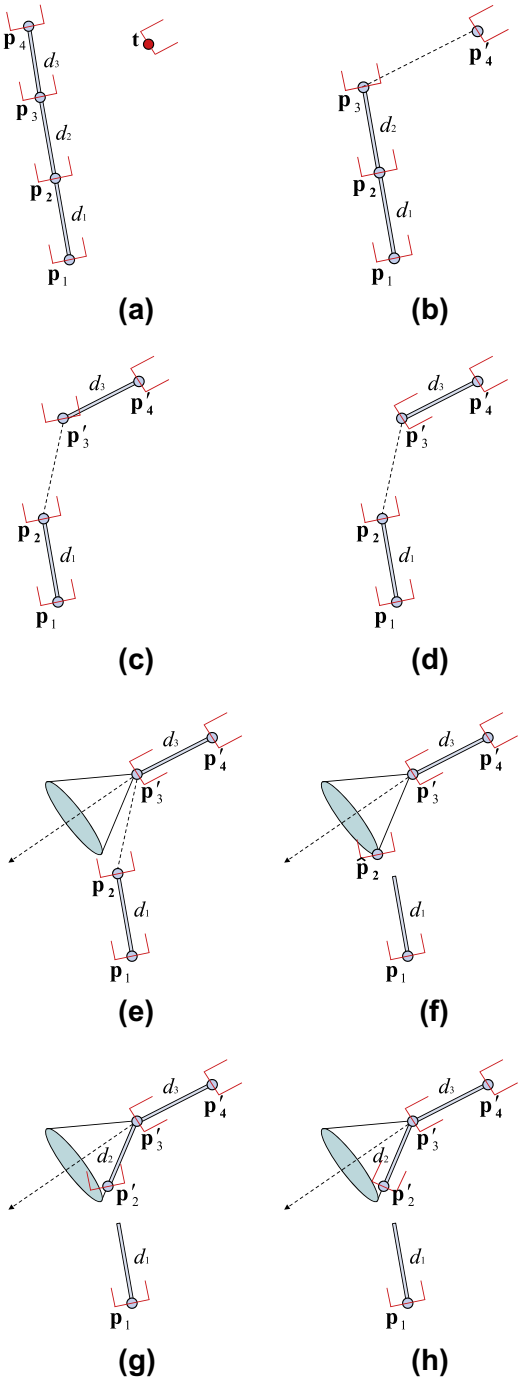

Fig. 5. Incorporating rotational and orientational constraints within FABRIK. (a) The initial configuration of the manipulator and the target, (b) relocate and reorient joint $\mathbf{p}_4$ to target **t**, (c) move joint $\mathbf{p}_3$ to $\mathbf{p}'_3$, which lies on the line that passes through the points $\mathbf{p}'_4$ and $\mathbf{p}_3$ and has distance d_3 from $\mathbf{p}'_4$, (d) reorient joint $\mathbf{p}'_3$ in such a way that the rotor expressing the rotation between the orientation frames at joints $\mathbf{p}'_3$ and $\mathbf{p}'_4$ is within the motion range bounds, (e) the rotational constraints: the allowed regions shown as a shaded composite ellipsoidal shape, (f) the joint position $\mathbf{p}_2$ is relocated to a new position, $\hat{\mathbf{p}}_2$, which is the nearest point on that composite ellipsoidal shape from $\mathbf{p}_2$, ensuring that the new joint position $\mathbf{p}'_2$ will be within the allowed rotational range. (g) move $\hat{\mathbf{p}}_2$ to $\mathbf{p}'_2$, to conserve bone length, (h) reorient the joint $\mathbf{p}'_2$ in order to satisfy the orientation limits. This procedure is repeated for all the remaining joints in a forward and backward fashion.

is not in the allowed motion range. The nearest point on an ellipse from a point can be found by simultaneously solving the ellipse equation and the equation of the tangent line at the orthogonal contacting point on the ellipse using the Newton-Raphson method, as described in [46]. Obviously, it is not necessary to calculate all the ellipses which define the composite ellipsoidal shape of Fig. 4(b), but only the ellipse related to the quadrant in which the target is located. It is important here to recall that, if the constraints which define the allowed range of motion are described by a different conic section (circle or parabola), the target should be re-positioned as the nearest point on that conic section, similarly to the ellipsoidal case. The last step is to undo the initial transformation which mapped O to the origin. This procedure is illustrated in pseudo-code in Algorithm 3 and a demonstration of the process is given in Fig. 5.

Algorithm 3. The rotational constraints

Input The target position **t** and the angles defining the rotation constraints θ_j for $j = 1,\ldots,4$.
Output: The new target position **t**′.
3.1 Find the line equation L_1
3.2 Find the projection O of the target **t** on line L_1
3.3 Find the distance between the point O and the joint position
3.4 Map the target (rotate and translate) in such a way that O is now located at the axis origin and oriented according to the x and y-axis ⇒ Now it is a 2D simplified problem
3.5 Find in which quadrant the target belongs
3.6 Find what conic section describes the allowed range of motion
3.7 Find the conic section which is associated with that quadrant using the distances $q_j = S\tan\theta_j$, where $j = 1,..,4$
3.8 *% Check whether the target is within the conic section or not*
3.9 **if** within the conic section **then**
3.10 use the true target position **t**
3.11 **else**
3.12 Find the nearest point on that conic section from the target
3.13 Map (rotate and translate) that point on the conic section via reverse of **3.4** and use that point as the new target position
3.14 **end**

This is a versatile, and easily visualisable, method of restricting where the bone can go. Incorporating this methodology within an IK solver, such as FABRIK, will give us the opportunity to reconstruct or track animated figures with high accuracy. IK algorithms are generally more effective if the constraints are applied at each step (not at the end of the algorithm), ensuring that the rotational and orientational restrictions are satisfied at each iteration. Thus, the proposed joint constraints can be applied within FABRIK by ensuring that the target, at each step, is moved to be within the allowed orientational and rotational bounds. Hence, assume that we are in the first stage of the algorithm, and have just calculated the new positions of the joints, $\mathbf{p}'_{i+1}$ and $\mathbf{p}'_i$, and we want to find the new position of the $(i-1)^{th}$ joint, $\mathbf{p}'_{i-1}$. Check if the joint $\mathbf{p}_{i-1}$ satisfies the orientational limits and if so, check whether it is within the composite ellipsoidal shape that describes the allowed range bounds, as illustrated above. If it is not, then $\mathbf{p}_{i-1}$ should be re-oriented and/or re-positioned within the allowed bounds ($\hat{\mathbf{p}}_{i-1}$). Thereafter, $\mathbf{p}'_{i-1}$ can be defined as the point on the line l_{i-1}, which passes through the joint positions $\hat{\mathbf{p}}_{i-1}$ and $\mathbf{p}'_i$ and has d_{i-1} distance from $\mathbf{p}'_i$, as is illustrated in Fig. 5.

The same technique for constraining joints is applied in the second stage of the algorithm and for each iteration until the target is reached or there is no significant change in the end effectors' positions. The algorithm copes with joints and limbs having 3 DoFs, and it can handle cases of joint and limb twist. It is important to recall here that the inter-joint lengths do not change over time since these distances are implicity kept constant by FABRIK.

The proposed restriction methodology can be easily extended to manage joint limits greater than 180 degrees. For

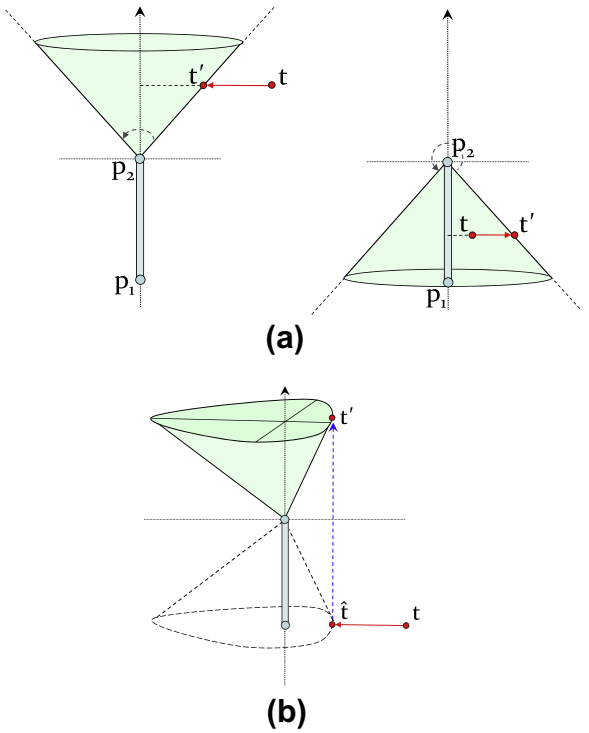

Fig. 6. Solution for special joint restriction cases: (a) the original case and when the allowed range of motion is greater than 180 degrees, (b) when the target is located on a different hemisphere than the irregular cone.

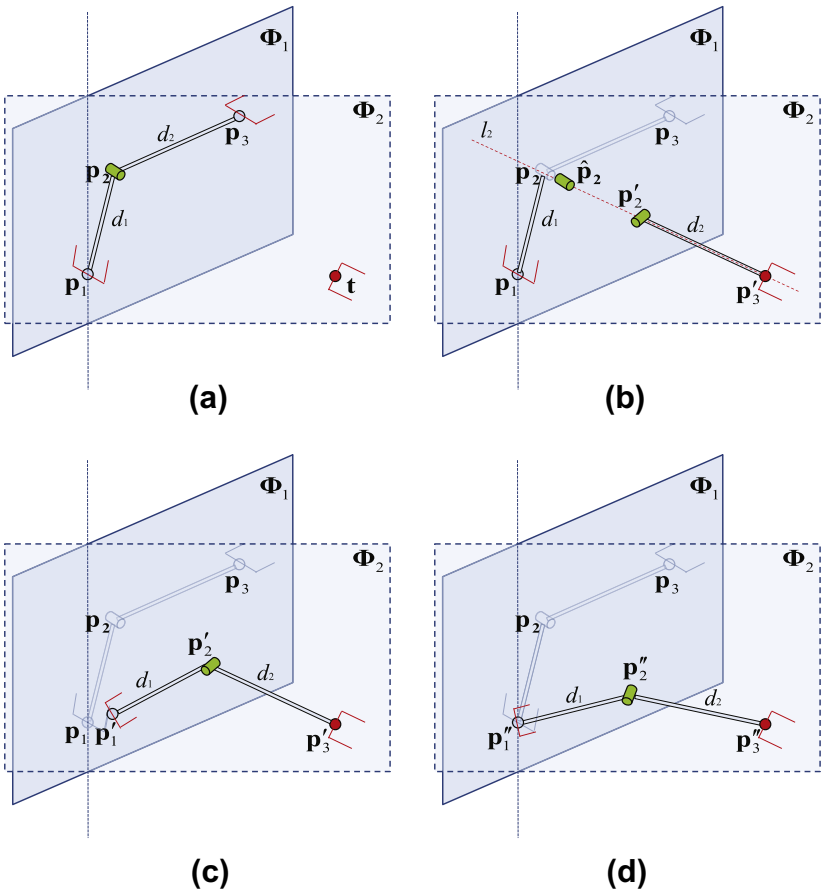

Fig. 7. Incorporating constraints for a hinge joint. (a) The initial configuration of the manipulator and the target. $\mathbf{\Phi}_1$ gives the plane of the allowed motion and it is defined by the hinge joint $\mathbf{p}_2$ (1 DoF). The root and the target, which is oriented, also define the plane $\mathbf{\Phi}_2$. (b) Relocate and reorient joint $\mathbf{p}'_3$ to target $\mathbf{t}$. Then, project $\mathbf{p}_2$ onto the plane $\mathbf{\Phi}_2$ to give a new point $\hat{\mathbf{p}}_2$, and find the point $\mathbf{p}'_2$ on line l_2 that passes from the joint position $\mathbf{p}'_3$ and the projected joint position $\hat{\mathbf{p}}_2$ and has distance d_2 from $\mathbf{p}'_3$. Reorient the new joint using the orientation constraints. (c) move and reorient joint $\mathbf{p}_1$ to $\mathbf{p}'_1$, which lies on the line that passes through the points $\mathbf{p}'_2$ and $\mathbf{p}_1$ and has distance d_1 from $\mathbf{p}'_2$, (d) The problem is now again a 2D problem and all joints lie on plane $\mathbf{\Phi}_2$. Thus, FABRIK can then be applied to of all the remaining joints in a forward and backward fashion.

instance, when the angle which defines the allowed range of motion is greater than 180 degrees, the associated irregular cone will define the area which is outside the limits. In that case, the joint restriction methodology will work in a reverse fashion; if the target is within the irregular cone area, meaning it is outside the limits, it will be projected to the cone surface, as is demonstrated in Fig. 6(a). Another special case of joint restriction occurs when the target is located in such a position that is not in the same hemisphere (in Fig. 6(b), the upper hemisphere) as the irregular cone. The limits of motion are defined as the irregular cone in the upper hemisphere and a reflection of the cone in the lower hemisphere; the target in the lower hemisphere is projected onto the limit boundary by first projecting its position onto the reflected cone and taking the associated point on the regular cone, as shown in Fig. 6(b). Obviously, the algorithm works in a similar way for different conic sections.

One big advantage of the proposed methodology is that no bone requires rotation to lie in any cone or polygon window, such as those described in [41,42]; it is only necessary to check whether the target is within the composite ellipsoidal shape defined by the restrictions on the motion. It loses none of the advantages of the FABRIK algorithm, incorporating joint limits via only points, lines and basic 2D entities; no rotational matrices need to be calculated, resulting in large savings in computational time. If it is desirable to retrieve the joint angles, all necessary information is of course available (position and orientation of each joint).

If more information about the allowed range of motion is available, the proposed methodology can be extended to include increased sophistication, supporting more complex joint types. Thus, instead of having an ellipsoidal entity to describe the sub-area in which the target can be placed, a polygonal area can be implemented. If the target is out of range, we would look for the nearest point on the polygon.

The constraining methodology can also be modified to support other IK solvers. There are, however, some limitations on what joint types this prototype version can support, since it is assumed that the inter-joint distance remains constant over time. Prismatic, slicing or shifting

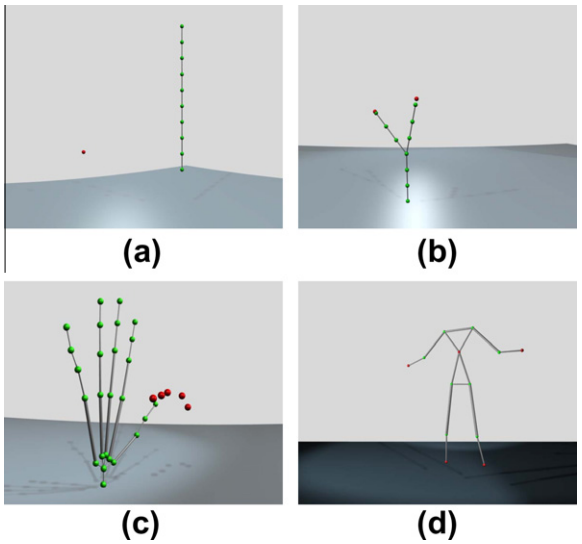

Fig. 8. The structure of the models used in our experimental examples. (a) A kinematic chain consisting of 10 joints and 1 end effector. There are 2 kinematic chain models, an unconstrained and a constrained version, (b) a kinematic model with 10 unconstrained joints and 2 end effectors, (c) a hand model with 26 unconstrained joints and 5 end effectors, (d) a humanoid with 13 unconstrained joints and 5 end effectors. The target positions (end effectors) are shown in red, while the joint positions are shown in green.

joints (joint types more usually discussed in robotics) are not directly supported and more information about the joint is required to attain a solution. Self-collisions can be handled using existing techniques, such as [47]; but more work is needed to ascertain if the FABRIK framework gives any advantages when dealing with self occlusions.

The problem of limiting the joint movements for simple 2D joint models, such as the *hinge* joint, can be simplified using alternative approaches. Since FABRIK operates on the joint coordinates by adjusting the positions in an iterative fashion, the 2D restrictions can be enforced by projecting the joint onto the plane of orientation. That plane is defined by the root and the (oriented) target position. An illustration showing how restrictions can be enforced for a hinge manipulation is given in Fig. 7. Similar techniques can be applied to incorporate constraints for different types of joint, in a variety of motions.

The ability of the constrained version of the algorithm to converge has not been reduced. If the target is within the reachable area and there is a joint configuration which allows the chain to bend enough and reach the target, FABRIK will attain the solution. Obviously, since the chain is not free from joint restrictions, there are instances where the target is not reachable. For those instances, there is a termination condition, similar to the unconstrained version, which compares the previous and the current position of the end effector.

5. Experimental results

A target database has been created for the validation and testing of the IK methods. The database consists of

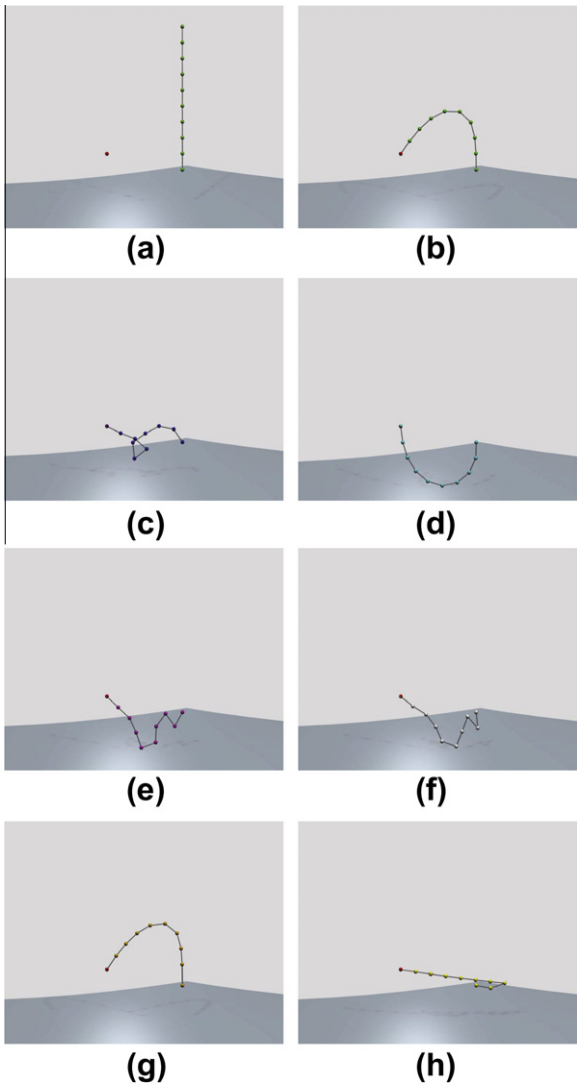

Fig. 9. Experimental solutions using some of the most popular IK methods. The kinematic chains consisted of 10 unconstrained joints, allowing 3 DoFs on each joint. (a) Initial position, (b) FABRIK, (c) CCD, (d) J. Transpose, (e) J. DLS, (f) J. SVD-DLS, (g) FTL, (h) Triangulation. Note that the link lengths of the resulting FTL posture are not equal to their initial size.

reachable and unreachable targets, targets with different distances from the end effectors and targets that move smoothly in space with end effectors tracking their position. The tests also consist of reconstructing sequences with different classes of motion in order to process different swivel angles and axial orientations of the root joint. The examples are demonstrated in 6 different kinematic models; a chain with 10 unconstrained joints allowing 3 DoFs on each joint; a chain with 10 constrained joints allowing limited angle rotations with 3 DoFs; a model containing a 'Y-shape' having 10 unconstrained joints and 2 end effectors; a fully unconstrained and un-modelled hand with 26 joints, 3 DoFs on each joint of which 5 are end effectors and one the root; and a 13 joint humanoid model,

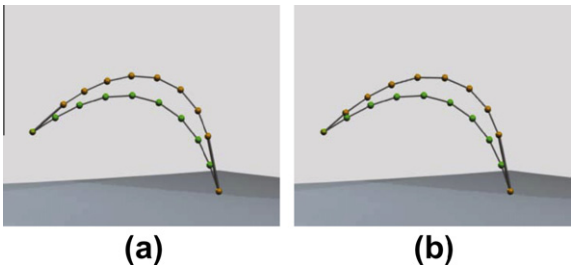

Fig. 10. A comparison between the FTL and the FABRIK approaches; the FABRIK posture is shown in green and the FTL posture in orange. (a) The FTL algorithm used fewer averaging iterations to calculated the final pose, thus the resulting posture has more link length variations, (b) the link length variation of the FTL algorithm was reduced over several iterations. Nevertheless, the changes in link size is still significant.

in a constrained and unconstrained version, with 3 DoFs on each joint and 5 end effectors. The IK problem for the hand and humanoid model is solved sequentially using closed loops in a predefined hierarchical order. Fig. 8 shows the different kinematic models used within this work.

IK techniques will mostly work with specified positions and orientations of specific joints, usually the end effectors, since they are more easily specified by the animator and tracked by the motion capture system; thereby, they automatically configure the remaining joints according to different criteria that depend on the model variant and joint restrictions.

Some of the most popular IK methods have been tested against FABRIK, such as CCD, Jacobian Transpose, Jacobian DLS, Jacobian pseudo-inverse DLS (SVD-DLS), FTL and Triangulation. In some of our experiments, we implemented examples with large distances between target and end effectors; hence, some methods tend to require more iterations to reach the target and thus the convergence differences are more obvious. The Jacobian and DLS parameter values used in our experiments are the parameter values suggested by Buss and Kim [7]; the damping constant was set to $\lambda = 1.1$. Several tests and comparisons have been implemented between the proposed algorithms in respect of their computational cost, processing time, convergence, the number of iterations needed to reach the target and the reconstruction quality.

5.1. A single end effector

In this section, the methods have been tested on problems with a single end effector and fixed target positions. These experiments did not include any joint constraints, but all methods could be enhanced to enforce rotational and orientational limits. An example with the resulting postures for each methodology is presented in Fig. 9.

FABRIK produces results significantly faster than all IK methods tested. It is approximately 10 times faster than the CCD method and a thousand times faster than the Jacobian-based methods, for these examples with large end effector movements. FABRIK has the lowest computational cost and, at the same time, produces visually the smoothest and most natural movements. It needs the fewest iterations to reach the target, it converges faster to the desired position and, when the target is unreachable, it keeps the end effector pointing to the target. On average, FABRIK needs 15.4 iterations and just 13.2ms to attain a reachable target and 67 iterations and 62ms for an unreachable target. When the target is unreachable, for this unconstrained model, FABRIK converges to a final answer in just 1 iteration and only 0.2ms.

CCD can also be applied in real-time. It is much faster than any Jacobian-based method; it needs, on average, 26 iterations and 123ms to reach the target when it is within reach. On the other hand, when the target is not reachable, it needs almost 400 iterations and 4sec to converge to its final solution (using the default algorithm without optimisations). However, CCD can often generate unrealistic postures since it can roll and unroll itself before reaching the target. CCD also tends to overemphasise the movements of the joints closer to the end effector of the kinematic chain.

The Jacobian methods return reasonable results; the reconstructed chain poses are visually more natural than CCD. Nevertheless, the biggest advantage of the Jacobian methods over all other methods is that, by default, they can treat problems with multiple end effectors very easily. Constraints can be applied within the Jacobian algorithms, but the way in which these restrictions are incorporated is not straightforward. Some Jacobian methods also suffer from singularity problems, since matrix inverses need to be calculated. The Transpose and DLS methods do not suffer in this way since they do not use the matrix inverse. The Jacobian methods also incur high computational cost making this family of methods non-ideal for real-time applications. The Jacobian methods generally converge very slowly to their final solutions since they use a linear approximation with a small step. This is more obvious in Fig. 11, where the number of iterations needed to reduce the distance between target and end effector as this changes over time is presented for each methodology. In this example, the original chain is 900mm long, the distance between target and end effector is 600mm, and the termination tolerance is 1×10^{-3}mm.

The FTL algorithm produces poses in real-time and its resulting postures tend to resemble poses in the first step of the first iteration of FABRIK (Fig. 10 and 11 shows a comparison between the FTL and FABRIK methods). However, the averaging step of the algorithm induced a variation of the link lengths compared to their initial size (this could be considered as its major drawback). Even after a large number of iterations, the length variations still exist. In cases where the target is unreachable, FTL stretches the chain length in order to reach the target. In addition, the iterations needed to restore the link lengths to their initial size (in some cases, FTL requires more than double the number of iterations of FABRIK) increase the computational cost and the processing time required to achieve a final pose. Brown et al. [24] does not provide solutions for problems with multiple end effectors and targets and, because of the averaging step, it is difficult to incorporate rotational and orientational constraints.

The Triangulation algorithm incurs a lower computational cost than the CCD algorithm and is substantially faster than the Jacobian methods. However, the resulting

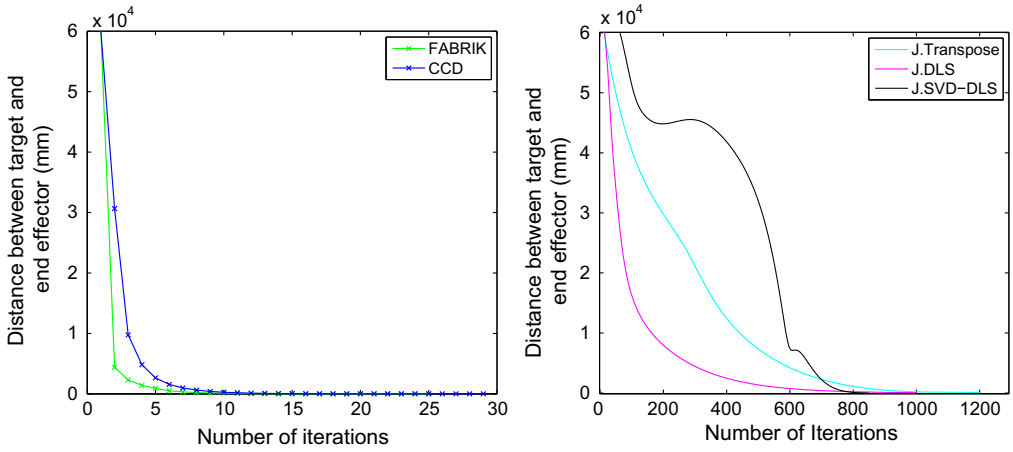

Fig. 11. The number of iterations needed to reach the target against the distance between target and end effector as this changes over time.

kinematic chain has a visually unrealistic shape; the joints close to the end effector are usually in a straight line, with the emphasis on rotation of the joints neighbouring the root. Another important drawback of the Triangulation algorithm is that it cannot be adapted for multiple end effectors, thus it cannot be used for complex character models. It also suffers from an inability to reach a feasible solution when joint constraints are applied; the end effector often cannot reach the target, even if there is a solution, since each joint position is calculated independently without considering the restrictions on the next joint.

Table 1 presents the average runtimes of each of the methods, as well as the number of iterations needed to reach the target, for both cases of a reachable and an unreachable target. Runtimes are in seconds and were measured with custom MATLAB code on a Pentium 2 Duo 2.2 GHz. No optimisations were used for any method reported in the table. It also indicates the time needed per iteration for each method and how many iterations per second each methodology can support. An iteration of FABRIK has the lowest computational cost since, instead of using angle rotations, it treats finding the joint locations as a problem of finding a point on a line.

Fig. 9 compares the performance of each algorithm for solving Inverse Kinematic problems; it shows the initial configuration and the goal solution obtained with each methodology. The manipulator is fully unconstrained and has no limits on the range of allowed movement for each joint. In each case a position goal is specified for the end effector and the Inverse Kinematic problem is solved to varying degrees of accuracy. Fig. 12 plots the convergence of each method, meaning the time taken to achieve the solution with the requested degree of accuracy. It is clearly observed that FABRIK converges to the target faster than any other implemented methodology. Also, Fig. 12 verifies that FABRIK always converges to the target, if the latter is reachable, in those cases tested.

The FABRIK, CCD, DLS and SVD-DLS methods have also been tested when the targets are moving in a sinusoidal trajectory and the end effectors are tracking their positions when they are within reach, and keeping the end effectors pointing at the targets when they are unreachable. The accuracy of the tracking was measured over a period of a thousand simulation steps. FABRIK tracks the target in real-time producing smooth and visually natural motion without erratic discontinuities. CCD produces reasonable results within the real-time constraints; however there are instances where the motion produced is not visually realistic. It is important to mention that CCD's performance improves when the target is within a small distance from the end effector's position or the frame rate is high. This happens because the kinematic chain does not roll and unroll itself. On the other hand, the Jacobian-based methods can produce oscillating motion with discontinuities. Their

Table 1
Average results (over 20 runs) for a single kinematic chain with 10 joints.

	Reachable Target				Unreachable Target	
	Number of Iterations	Matlab exe. time (*sec*)	Time per iteration (in *msec*)	Iterations per second	Number of Iterations	Matlab exe. time (*sec*)
FABRIK	15.461	0.01328	0.86	1164	67.564	0.06207
CCD	26.308	0.12356	4.69	213	390.135	3.92869
Jacobian Transpose	1311.190	12.98947	9.90	101	6549.000	33.90473
Jacobian DLS	998.648	10.48051	10.49	95	2881.667	14.87918
Jacobian SVD-DLS	808.797	9.29652	11.50	87	2808.452	15.97591
FTL	21.125	0.02045	0.97	1033	22.325	0.02526
Triangulation	1.000	0.05747	57.47	21	1.000	0.06993

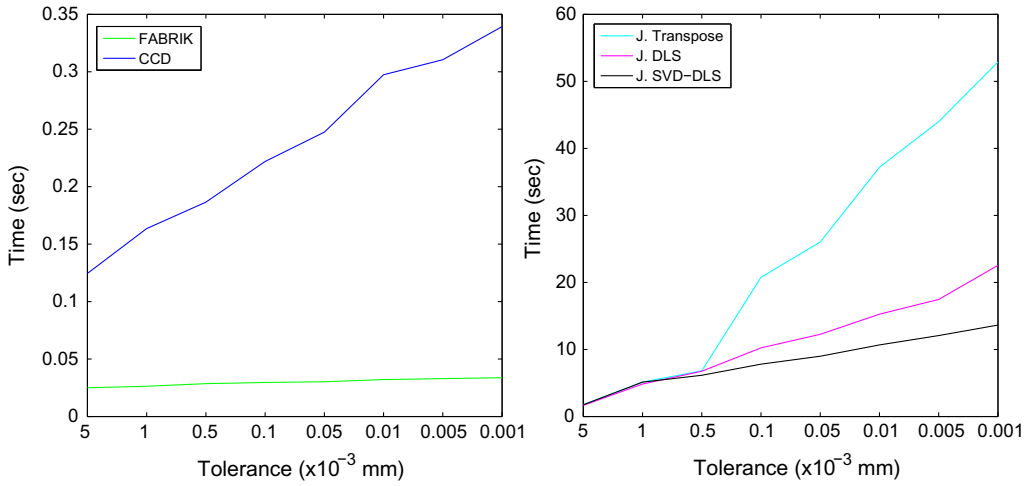

Fig. 12. An example presenting the time needed for each methodology to achieve the solution with the degree of accuracy requested.

biggest drawback however is the time needed to track the target; only under some circumstances, eg using fast C++ matrix libraries, can these kinds of methods reach the goal of real-time application. Fig. 13 presents the performance of each method on selected frames over time.

The FABRIK algorithm has been also implemented and tested within the Kine [12] application; Kine is a 2D real-time gaming application that initially has a kinematic chain with six joints. Kine allows you to interact with the IK solver; you click on the screen and the snake (the kinematic chain is drawn as a snake) moves to solve the IK problem. There is also an option where you click and drag on the screen and the snake tracks the mouse. Fig. 14 shows examples of the FABRIK and CCD methods implemented within the Kine environment when the end effector moves through large distances. It is clearly observed that FABRIK out-performs CCD in producing smoother poses.

5.2. Multiple end effectors

Most real models, such as the hand, legged bodies etc, consist of multiple chains, each chain having at least one end effector. Hence, it is essential to test our methodology in cases where more than 1 end effector exists. To test FABRIK under these conditions, we implemented the 'Y-shaped' multibody pictured in Fig. 15(a), also used in [7], and a hand multibody presented in Fig. 15(b). The 'Y-shape' multibody has 10 joints with 2 end effectors. The target positions (the red balls in the figures) moved in sinusoidally varying curves in and out of reach of the multibody. The target positions moved in small increments and in each time step the joint positions were updated. The simulations were visually inspected for oscillations and tracking quality. The end effectors can successfully track the target positions when they are within reach, and remain pointing at the targets when these are out of reach. Fig. 15(a) presents a simple example of how FABRIK performs with multiple end effectors; although it is hard to show in images, FABRIK can easily track both targets with a smooth motion and without oscillations, shaking or erratic discontinuities.

Fig. 15(b) shows another example of implementing FABRIK into a multiple end effector model. This is a fully unconstrained hand example with 5 end effectors and 26 joints in total, allowing 3 DoFs on each joint. Incorporating a highly constrained model that also considers the anatomically and physiologically properties of the hand, such as [40], the motion of each joint can be restricted to a feasible set and the hand will have even more natural movements.

Fig. 16 shows an example of a tracking animation of a humanoid with 13 joints, 5 of which are treated as end effectors. In this demonstration, the frame rate was low (3 frames per second); the 3Hz frame rate selection increases the distance between target and end effector, thus the performance of each method is more obvious. FABRIK can easily track the animated humanoid in real-time, producing very reasonable results. Fig. 17 shows the reconstruction quality of different methodologies over the same humanoid model. The differences between the implemented methodologies, on these unconstrained humanoid examples, are more obvious on shoulders, elbows and hips. FABRIK produces visually natural postures, having the smaller reconstruction error compared to the original sequences. These animations have been obtained from an optical markered motion capture system and have not been filtered; thus, the algorithm is shown to be robust in noisy environments. Selected internal joints have been artificially deleted in order to examine the reconstruction quality of each methodology. These humanoids do not have a mesh that defines their external shape, so self collisions are not considered within these reconstruction examples.

Table 2 shows the performance (over 20 runs) of each methodology for the case of a dancing humanoid model. The computational cost and the reconstruction quality for tracking the animated model is also presented. FABRIK gives the best results with respect to computational cost and reconstruction quality; it requires the fewest iterations to achieve the desired posture and produces visually the

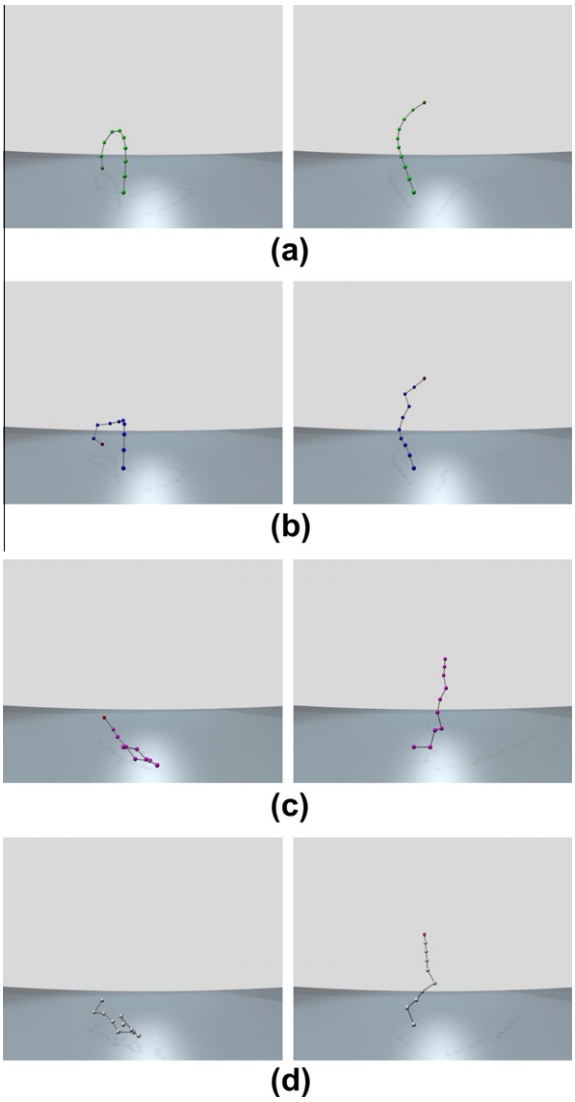

Fig. 13. Selected frames during target tracking at 25Hz using different IK solvers. (a) FABRIK, (b) CCD, (c) J. DLS, (d) J. SVD-DLS

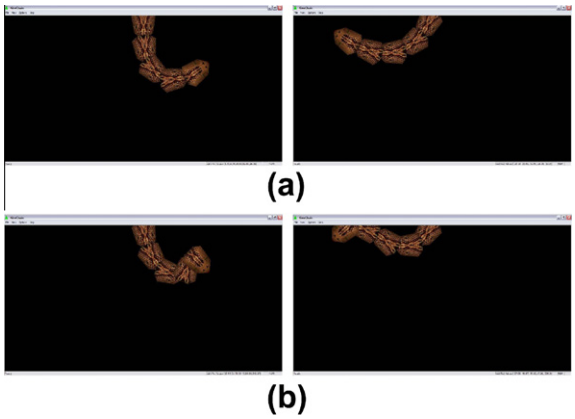

Fig. 14. FABRIK and CCD solution using the Kine application. (a) FABRIK solution, (b) CCD solution.

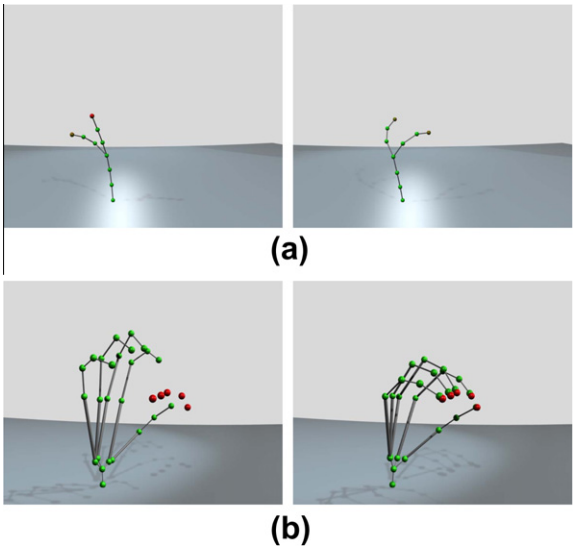

Fig. 15. An example of FABRIK implementation with multiple end effectors over time; (a) a kinematic chain with 10 unconstrained joints, 2 end effectors and 2 targets, (b) A hand motion example using FABRIK; this is a fully unconstrained hand example, allowing 3 DoFs on each joint.

smoothest poses. The median error presented in Table 2 refers to the difference between the estimated joint positions and the true joint positions.

5.3. Applying restrictions

Most IK problems have rotational and orientational restrictions since most real world joints have limitations on their movements. The experimental dataset used to test the reconstruction quality of the constrained FABRIK is made up of 10 joints, each having angle rotational restrictions allowing movements only within a range. The same humanoid model, as described in Section 5.2, is used to examine the reconstruction quality of the proposed methodology with and without constraints.

FABRIK can be easily constrained producing visually realistic postures without oscillations and discontinuities. The constrained version is slightly slower than its unconstrained counterpart, requiring now almost 3.0ms to reach the target. Nevertheless, it is still much faster than other IK methods and approximately 10 times faster than constrained CCD. The reconstruction quality is high, producing postures with an average error of just over 30mm, almost half the average error of the unconstrained version. On the other hand, while it is not difficult to apply manipulator constraints to CCD, the resulting animation often still has unnatural movements, especially when the target is at a significant distance from the end effector. The unconstrained version of CCD produces different joint poses compared to its constrained version, even if the latter is not violating the angle restrictions. It is interesting to note that there are instances where the constrained version of CCD needs fewer iterations and therefore performs slightly

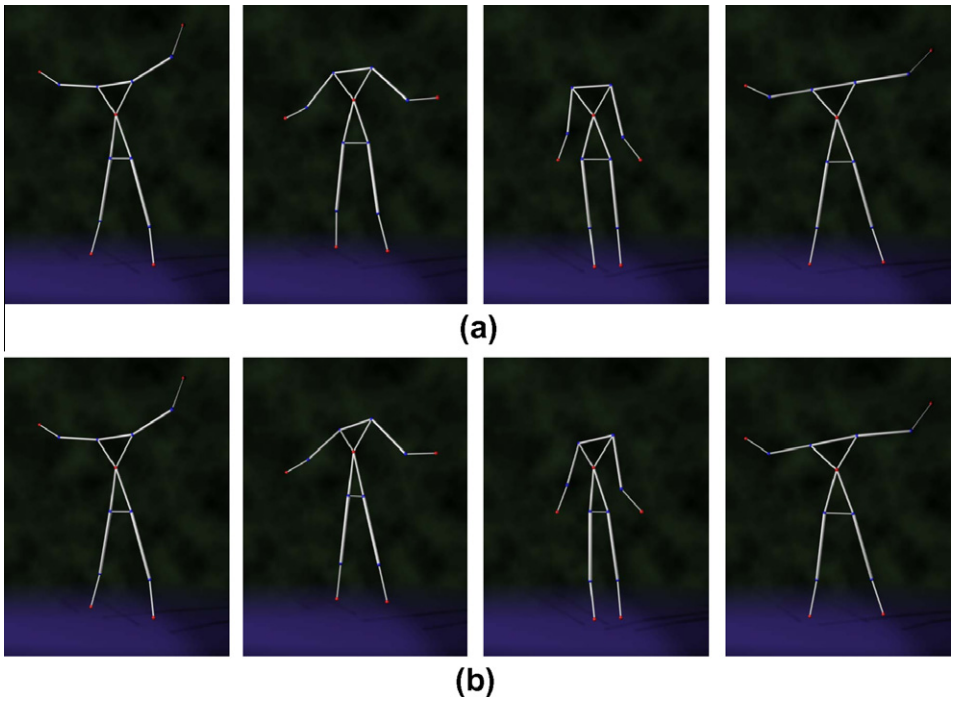

Fig. 16. A low rate body tracking example. The joints in red are the known positions of the end effectors and those in blue are the estimated joint positions. (a) shows the true body poses and (b) the estimated poses using FABRIK.

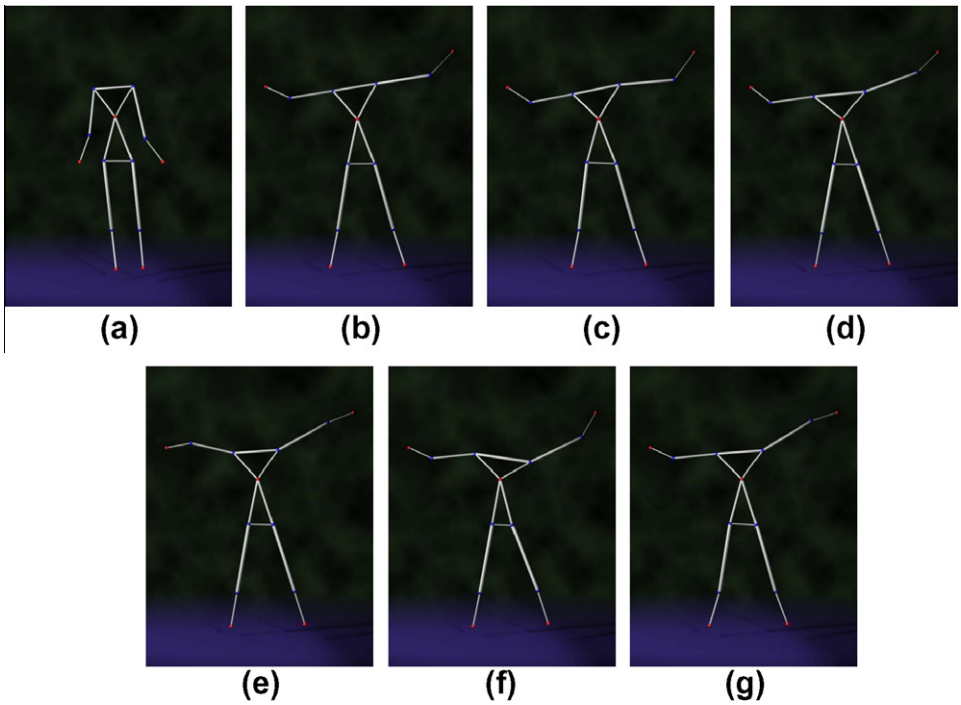

Fig. 17. Body reconstruction using different IK methodologies. The joints in red are the known positions of the end effectors and those in blue are the estimated joint positions. (a) shows the initial position and (b) the true final position. (c) shows the FABRIK solution, (d) the CCD solution, (e) the J. Transpose solution, (f) the J. DLS solution, (g) the J. SVD-DLS solution

faster than its unconstrained version. This happens because the constraints prevent the chain from rolling and unrolling itself before reaching the target. Fig. 18, shows examples of FABRIK and CCD implementations with and

Table 2
Reconstruction comparison. Average results (over 20 runs).

	Number of Iterations	Median time †	Time per iteration †	Median Error (*mm*)
FABRIK	65	1.6	0.0246	58.68
CCD	67	20.5	0.3060	69.99
J.Transpose	1352	1928.0	1.4334	137.42
J.DLS	804	1533.0	1.9067	84.84
J.SVD-DLS	723	1494.0	2.0664	83.73

† This is a MATLAB executable time in *msec*

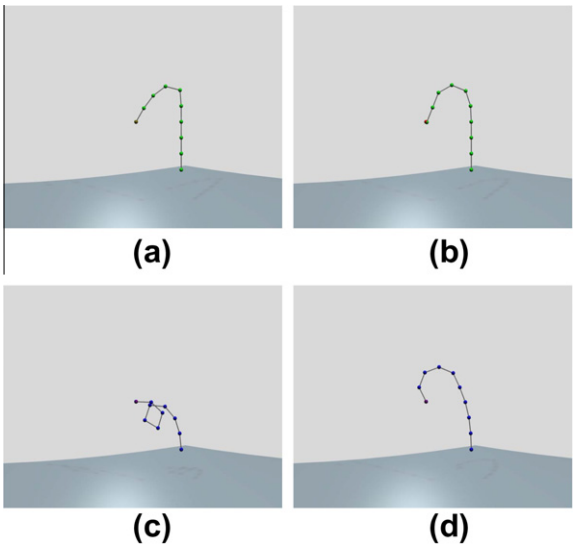

Fig. 18. An example of FABRIK and CCD implementations with and without incorporating constraints. (a) presents the FABRIK unconstrained solution and (b) the FABRIK constrained solution. (c) presents the CCD unconstrained solution, and (d) is the CCD constrained solution.

without joint restrictions. On that example, rotational limits have been applied restricting the allowed bending of each joint angle to a maximum value (treated as ball and socket joints). Fig. 19 shows the reconstruction improvement between an unconstrained and a constrained version of FABRIK applied to the humanoid model; rotational and orientational constraints have been employed on each joint (including the hinge joint technique for elbows and knees) limiting the angle and the twist between limbs to a feasible set.

5.4. FABRIK limitations

During the implementations and testing of the algorithm, no significant limitations have been encountered. Some minor limitations on what joint types the algorithm supports are mentioned in Section 4.3; however, these limitations can be coped with if more information is provided about the joint. For instance, in the case of a prismatic joint, FABRIK can obtain a solution if the final inter-joint distance or its minimum and maximum allowed length are known a priori. A breakdown of the algorithm (a singularity problem) has been also observed when the kinematic chain

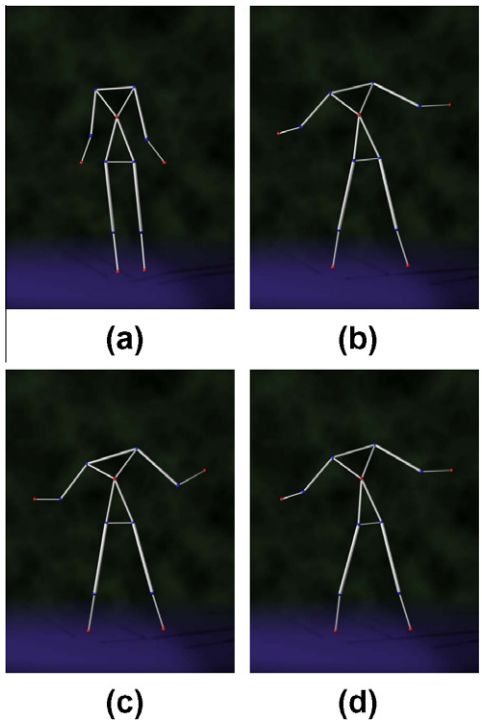

Fig. 19. An example of implementation. (a) The initial position, (b) the real posture, (c) the solution using unconstrained FABRIK, (d) the solution after incorporating joint restrictions.

was completely straight and the target was located exactly on that alignment but between the two joints (on the line which connects two joints). In that case, the chain would be displaced towards the root but would still be straight after the forward step of the process. And after the second step (backward), it would be back at the original place, thus entering to an infinite loop (a similar problem is encountered in the CCD algorithm). However, this rare singularity problem can be easily handled by allowing the chain to bend by a very small angle within the user constraints (by repositioning the target during the first backward stage and then returning it to its original location).

6. Conclusions and future work

IK methods are used to control the postures of articulated bodies in frame animation production. However, most of the currently available methods suffer from high computational cost and/or production of unrealistic poses. In this paper, FABRIK, a simple, fast and reliable IK solver is presented. This is the first algorithm to use an iterative method with points and lines to solve the IK problem. It divides the problem into 2 phases, a forward and backward reaching approach, and it supports (to the best of our knowledge) all the rotational joint limits and joint orientations by repositioning and re-orienting the target at each step. It does not suffer from singularity problems and it is fast and computationally efficient. No pre-recorded motion database is necessary, thereby avoiding the need for extra memory. Also, a reliable methodology for applying joint restrictions,

which supports and utilises all the advantages of FABRIK, is presented. Our experiments show that FABRIK requires on average fewer iterations to reach the target than any other IK method tested, both with constrained and unconstrained kinematic chains. At the same time, it produces visually smooth postures, with and without constraints, reaching the desired position with very low computational cost. FABRIK can be also extended to a multiple end effector version supporting multiple kinematic chains. Future work will see the introduction of the proposed algorithm within analytically and anatomically correct models. A further study of the collision problem, with simultaneous study of more sophisticated joint types, is also essential for the production of more natural movements.

Acknowledgments

The kine environment presented in Section 5.1 has been adapted from the work of Jeff Lander [12]; we would like to express our enormous thanks to Jeff for giving us permission to use his code and application. We would also like to thank Richard Wareham, Jonathan Cameron and Charles Lee for their invaluable discussions and help with capturing data and producing video examples.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.gmod.2011.05.003.

References

[1] Jianmin Zhao, Norman I. Badler, Inverse kinematics positioning using nonlinear programming for highly articulated figures, ACM Transactions on Graphics (TOG) 13 (4) (1994) 313–336.
[2] A. Balestrino, G. De Maria, L. Sciavicco, Robust control of robotic manipulators, in: Proc. of the 9th IFAC World Congress, vol. 5, 1984, pp. 2435–2440.
[3] W.A. Wolovich, H. Elliott, A computational technique for inverse kinematics, in: The 23rd IEEE Conf. on Decision and Control, vol. 23, 1984, pp. 1359–1363.
[4] J. Baillieul, Kinematic programming alternatives for redundant manipulators, in: Proc. of the IEEE International Conf. on Robotics and Automation, vol. 2, March 1985, pp. 722–728.
[5] C.W. Wampler, Manipulator inverse kinematics solutions based on vector formulations and damped least-squares methods, IEEE Transactions on Systems, Man and Cybernetics 16 (1) (1986) 93–101.
[6] Y. Nakamura, H. Hanafusa, Inverse kinematic solutions with singularity robustness for robot manipulator control, Transactions ASME, Journal of Dynamic Systems, Measurement, and Control 108 (3) (1986) 163–171.
[7] Samuel R. Buss, Jin-Su Kim, Selectively damped least squares for inverse kinematics, Journal of Graphics Tools 10 (3) (2005) 37–49.
[8] Alexandre N. Pechev, Inverse kinematics without matrix invertion, in: Proc. of the 2008 IEEE International Conf. on Robotics and Automation, Pasadena, CA, USA, May 19–23 2008, pp. 2005–2012.
[9] Roger Fletcher, Practical Methods of Optimization, second ed., Wiley Interscience, New York, NY, USA, 1987.
[10] Li-Chun Tommy Wang, Chih Cheng Chen, A combined optimization method for solving the inverse kinematics problems of mechanical manipulators, IEEE Transactions on Robotics and Automation 7 (4) (1991) 489–499.
[11] Chris Welman, Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation, Master Dissertation, Simon Fraser University, Department of Computer Science, 1993.
[12] Jeff Lander, Making kine more flexible, Game Developer 5 (3) (1998) 15–22.
[13] Adrian A. Canutescu, Roland L. Dunbrack, Cyclic coordinate descent: a robotics algorithm for protein loop closure, Protein Science 12 (5) (2003) 963–972.
[14] Luis Unzueta, Manuel Peinado, Ronan Boulic, Ángel Suescun, Full-body performance animation with sequential inverse kinematics, Graphical Models 70 (5) (2008) 87–104.
[15] Ronan Boulic, Javier Varona, Luis Unzueta, Manuel Peinado, Angel Suescun, Francisco Perales, Evaluation of on-line analytic and numeric inverse kinematics approaches driven by partial vision input, Virtual Reality 10 (1) (2006) 48–61.
[16] Richard Kulpa, Franck Multon, Fast inverse kinematics and kinetics solver for human-like figures, in: International Conference on Humanoid Robots, IEEE-RAS, December 2005, pp. 38–43.
[17] Nicolas Courty, Elise Arnaud, Inverse kinematics using sequential monte carlo methods, in: Proc. of the V Conf. on Articulated Motion and Deformable Objects, AMDO'08, LNCS, vol. 5098, Mallorca, Spain, 2008, pp. 1–10.
[18] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John Deweese, Jordan Maynard, Kees van Prooijen, Real-time motion retargeting to highly varied user-created morphologies, ACM Transactions on Graphics (TOG) 27 (3) (2008) 1–11.
[19] Keith Grochow, Steven L. Martin, Aaron Hertzmann, Zoran Popović, Style-based inverse kinematics, in: SIGGRAPH '04: ACM Trans. on Graphics, ACM, New York, NY, USA, August 2004, pp. 522–531.
[20] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, Jovan Popović, Mesh-based inverse kinematics, ACM Transactions of Graphics 24 (3) (2005) 488–495.
[21] Kevin G. Der, Robert W. Sumner, Jovan Popović, Inverse kinematics for reduced deformable models, in: ACM SIGGRAPH, ACM, New York, NY, USA, 2006, pp. 1174–1179.
[22] R. Mü ller-Cajar, R. Mukundan, Triangulation: a new algorithm for inverse kinematics, in: Proc. of the Image and Vision Computing New Zealand 2007, New Zealand, December 2007, pp. 181–186.
[23] R. Mukundan, A robust inverse kinematics algorithm for animating a joint chain, International Journal of Computer Applications in Technology 34 (4) (2009) 303–308.
[24] Joel Brown, Jean-Claude Latombe, Kevin Montgomery, Real-time knot-tying simulation, The Visual Computer: International Journal of Computer Graphics 20 (2) (2004) 165–179.
[25] David Hestenes, Garret Sobczyk, Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics, D. Reidel, 1984.
[26] Chris Doran, Anthony Lasenby, Geometric Algebra for Physicists, Cambridge University Press, Cambridge, UK, 2003.
[27] Andreas Aristidou, Joan Lasenby, Motion capture with constrained inverse kinematics for real-time hand tracking, in: Proc. of the International Symposium on Communications, Control and Signal Processing, Limassol, Cyprus, March 3–5 2010.
[28] Andreas Aristidou, Joan Lasenby, Inverse kinematics solutions using conformal geometric algebra, in: L. Dorst, J. Lasenby (Eds.), Guide to Geometric Algebra in Practice, Springer Verlag, 2011.
[29] Andreas Aristidou, Tracking and Modelling Motion for Biomechanical Analysis, PhD Thesis, University of Cambridge, Cambridge, UK, October 2010.
[30] Walter Maurel, Daniel Thalmann, Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones, Computers & Graphics 24 (2) (2000) 203–218.
[31] Xuguang Wang, Jean Pierre Verriest, A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation, Journal of Visualization and Computer Animation 9 (1) (1998) 33–47.
[32] N. Klopcar, M. Tomšič, J. Lenarčič, A kinematic model of the shoulder complex to evaluate the arm-reachable workspace, Journal of Biomechanics 40 (1) (2007) 86–91.
[33] Lorna Herda, Raquel Urtasun, Andrew Hanson, Pascal Fua, Automatic determination of shoulder joint limits using experimentally determined quaternion field boundaries, International Journal of Robotics Research 22 (6) (2003).
[34] Norman I. Badler, Cary B. Phillips, Bonnie Lynn Webber, Simulating Humans: Computer Graphics Animation and Control, Oxford University Press, New York, Oxford, 1993.
[35] Augustus A. White III, Manohar M. Panjabi, Clinical Biomechanics of the Spine, second ed., J.B. Lippincott Company, 1990.
[36] James Urey Korein, A Geometric Investigation of Reach, MIT Press, Cambridge, MA, USA, 1985.
[37] Gary Monheit, Norman I. Badler, A kinematic model of the human

spine and torso, IEEE Computer Graphics and Applications 11 (2) (1991) 29–38.
[38] Hans Rijpkema, Michael Girard, Computer animation of knowledge-based human grasping, in; SIGGRAPH '91: Proc. of the 18th Annual Conf. on Computer Graphics and Interactive Techniques, ACM, New York, NY, USA, 1991, pp. 339–348.
[39] Richard M. Murray, S. Shankar Sastry, Li Zexiang, A Mathematical Introduction to Robotic Manipulation, CRC Press, Inc., Boca Raton, FL, USA, 1994.
[40] Paris Kaimakis, Joan Lasenby, Gradient-based hand tracking using silhouette data, in: Proc.of the 3rd International Symposium on Visual Computing (ISVC), vol. 1, Lake Tahoe, NV/CA, USA, November 26–28 2007, pp. 24–35.
[41] Jonathan Blow, Inverse Kinematics with Quaternion Joint Limits, Game Developer, April 2002.
[42] Jane Wilhelms, Allen Van Gelder, Fast and easy reach-cone joint limits, Journal of Graphic Tools 6 (2) (2001) 27–41.
[43] Paolo Baerlocher, Ronan Boulic, Parametrization and range of motion of the ball-and-socket joint, in: Deformable Avatars, Kluwer Academic Publishers, 2001. pp. 180–190.
[44] Deepak Tolani, Ambarish Goswami, Norman I. Badler, Real-time inverse kinematics techniques for anthropomorphic limbs, Graphical Models 62 (5) (2000) 353–388.
[45] Kiyomi Yamane, Yoshihiko Nakamura, Natural motion animation through constraining and deconstraining at will, IEEE Transactions on Visualization and Computer Graphics 9 (3) (2003) 352–360.
[46] Sung Joon Ahn, Wolfgang Rauh, Hans-Jnrgen Warnecke, Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola, Pattern Recognition 34 (12) (2001) 2283–2303.
[47] Ming C. Lin, Stefan Gottschalk, Collision detection between geometric models: a survey, in: Proc. of IMA Conf. on Mathematics of Surfaces, 1998, pp. 37–56.