COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2016; **27**:35–57
Published online 2 February 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1630

RESEARCH ARTICLE

Extending FABRIK with model constraints

Andreas Aristidou[1]*, Yiorgos Chrysanthou[1] and Joan Lasenby[2]

[1] Department of Computer Science, University of Cyprus, Nicosia, 1678, Cyprus
[2] Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ, UK

ABSTRACT

Forward and Backward Reaching Inverse Kinematics (FABRIK) is a recent iterative inverse kinematics solver that became very popular because of its simplicity, convergence speed and control performance, especially in models with multiple end effectors. In this paper, we extend and/or adjust FABRIK to be used in problems with leaf joints and closed-loop chains and to control a fixed inter-joint distance in a kinetic chain with unsteady data. In addition, we provide optimisation solutions when the target is unreachable and a proof of convergence when a solution is available. We also present various techniques for constraining anthropometric and robotic joint models using FABRIK and provide clarifications and solutions to many questions raised since the first publication of FABRIK. Finally, a human-like model that has been structured hierarchically and sequentially using FABRIK is presented, utilising most of the suggested joint models; it can efficiently trace targets in real time, without oscillations or discontinuities, verifying the effectiveness of FABRIK. Copyright © 2015 John Wiley & Sons, Ltd.

KEYWORDS

animation; FABRIK; human modelling; inverse kinematics; joint configuration

Supporting information may be found in the online version of this article.

***Correspondence**

Andreas Aristidou, Department of Computer Science, University of Cyprus, Nicosia, 1678, Cyprus.
E-mail: a.aristidou@ieee.org

1. INTRODUCTION

The production of realistic and plausible motions has been a long-standing problem for scholars in many fields, including robotics technology and computer graphics. During recent decades, several approaches have been implemented for solving the inverse kinematics (IK) problem; IK is a method for computing the skeletal configuration of a figure via estimating each individual degree of freedom (DoF) in order to satisfy a given task. IK finds applications in many areas where the animation and/or control of different virtual creatures is necessary. IK methods are also frequently used in the video games industry and in the field of computer-aided ergonomics, especially in human model development and for simulation purposes. However, most of the currently available IK solvers seem to have drawbacks, such as erratic discontinuities and singularities. They also suffer from unnatural poses, have difficulties in dealing with complex figures and are computationally expensive.

Forward and Backward Reaching IK (FABRIK) [1] is a recent, iterative algorithm that uses points and lines to solve the IK problem. It divides the problem into two relevant phases, a forward step and a backward step, and supports all the rotational joint limits and joint orientations by re-positioning and re-orienting the target at each step. It does not suffer from singularity problems and produces smooth motion without discontinuities. The main advantages of the FABRIK approach are its simplicity, the ease with which it can be fit into various models, the support of direct optimisations and its ability to control multiple end effectors, making it ideal for applications in systems that require real-time computation.

In this paper, we present various extensions of the FABRIK algorithm, thus demonstrating its effective usage in real-world scenarios. An assortment of different anthropometric and robotic joint models has been constructed, incorporating manipulator constraints. Moreover, a variety of different solutions using adjustments of FABRIK are presented, solving the IK problem in cases with multiple end effectors, in cases where the 'end effector' is not positioned at the end of the chain (leaf joints) and in cases where the chain is in a closed-loop form. The algorithm has also been modified to ensure fixed distances under noisy or unsteady data. In addition, some minor problems that have been raised since the first publication

of the algorithm are clarified, and solutions are clearly described. We also indicate when the targets are reachable or not for cases of single or multiple end effector, providing optimisation solutions that can save an important amount of computational time. We also provide a proof for the convergence of the unconstrained version of FABRIK, if there is a solution available. Finally, the proposed joint and model constraints have been incorporated in a human-like model for evaluation purposes; the human-like model has been structured hierarchically, while each individual kinematic chain is solved sequentially using the constrained solutions proposed in this paper, so as to enable tracking of multiple targets.

2. LITERATURE REVIEW

2.1. Inverse Kinematics

The solutions to the IK problem can be classified as belonging to one of six main categories, the *analytical*, *Jacobian*, *Newton*, *statistical*, *data-driven* and *heuristic* methods. *Analytical* methods, such as those in [2–5], find all possible solutions as a function of the lengths of the mechanism, its starting posture and the rotation constraints. The analytical methods are mainly used in robotics in order to solve the IK problem of an anthropometric arm and leg. The analytical IK solutions have no singularity problems, they offer a global solution and they are fast, simple and reliable; however, the non-linear nature of the kinematic equations makes them not suitable for redundant systems with more than 7 DoFs.

The *Jacobian solutions* are linear approximations of the IK problem; they 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, singular value decomposition, damped least squares, selectively damped least squares and several extensions [6–11]. Jacobian inverse solutions produce smooth postures; however, most of these approaches suffer from high computational cost, complex matrix calculations and singularity problems. Recently, Kenwright proposed an approach that solves the IK problem using the Gauss–Seidel iterative approximation method [12] that does not suffer from singularity problems.

Another family of IK solvers is based on *Newton* methods. These algorithms seek target configurations that 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 method [13]. However, the Newton methods are complex, are difficult to implement and have a high computational cost per iteration.

Recently, some papers have been introduced that solve the IK problem from a statistical point of view. Courty and Arnaud [14] proposed a sequential Monte Carlo method to incorporate kinematic constraints. Hecker *et al*. [15] utilised an iterative IK solver (particle IK) with various parameters for tuning the character skeleton behaviour both statically and dynamically. Neither method suffers from matrix singularity problems, and both perform reasonably well. An alternative approach is given by Pechev in [16] where the problem is solved from a control perspective; this approach is computationally efficient and does not suffer from singularity problems.

Data-driven methods use pre-learned postures to solve the IK problem, via methods that are based on neural nets and artificial intelligence. For instance, Grochow *et al*. [17] presented a style-based IK method that is based on a learned model of human poses. Given a set of constraints, the proposed system was able to produce, in real time, the most likely pose satisfying those constraints. Another alternative approach to the style-based IK was proposed by Wu *et al*. [18], named NAT-IK; instead of using continuous poses, they used discrete poses in order to add robustness to the IK solver. Wei *et al*. [19] have presented a data-driven algorithm for interactive posing of 3D human characters for large training motion database, while Ho *et al*. [20] proposed a data-driven framework that conserves the topology of the synthesising postures; using a *Gauss linking integral*, they create realistic human control, while avoiding body part penetration by distinguishing topologically different postures. Sumner *et al*. [21], and later Der *et al*. [22], proposed mesh-based IK solvers that learn the space of shapes from example meshes. Nevertheless, this family of methods produces poses that require a pre-learning phase and are highly dependent on the training data.

Finally, the *heuristic* methods are the simplest and fastest IK solvers. A very popular solution is the cyclic coordinate descent (CCD) method, first introduced by Wang and Chen [23]. CCD has a low computational cost for each joint per iteration and can solve the IK problem without matrix manipulations. However, CCD can suffer from unrealistic animation, even if manipulator constraints have been added. It is designed to handle serial chains; thus, it is difficult to extend to problems with multiple end effectors or target positions for internal joints. Nevertheless, there are several extensions of the CCD algorithm that deal with the production of unrealistic postures, such as the one proposed by Kulpa and Multon [24]. The triangulation algorithm [25] is another heuristic 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; it can only be applied to problems with a single end effector and does not support imposed joint limits. An improved version is given in [26] 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. A more detailed overview of IK techniques is given in [27].

DOI: 10.1002/cav

2.2. Forward and Backward Reaching Inverse Kinematics

In this paper, we discuss and extend the FABRIK algorithm; FABRIK is a recent, real-time IK solver that returns smooth postures in an iterative fashion. Instead of using angle rotations, FABRIK treats finding the joint locations as a problem of finding a point on a line; the algorithm traces back step by step to different positions of the joint of a chain, crossing the chain and back in a finite number of iterations. Although FABRIK is a recent algorithm, it has become a popular IK solver; many researchers and game developers have implemented or extended FABRIK because of its efficiency and simplicity. The latter is why it is most suitable for being applied on top of animation, when the pose becomes rewritten in each frame. For instance, FABRIK has been used for hand skeleton reconstruction [28,29] and for skeletal control under marker occlusion in motion capture (mocap) technology [30]. Poddighe and Roos [31] used FABRIK to enable a NAO humanoid robot to play tic-tac-toe, while they show that FABRIK outperforms methods that use the Jacobian inverse on all aspects; they also conclude that it is the only method from the tested algorithms that always yields results with the error tolerance set to zero and is well balanced near singularities. Moreover, Liu implemented FABRIK for robot manipulation [32] and Munshi for robot simulation [33], while Lo and Xie [34] used FABRIK in a redundant four-revolute (4R) spherical wrist mechanism for an active shoulder exoskeleton. Recently, Hwang and Choi [35] exploited the advantages of FABRIK to manipulate multiple chains and used it to estimate the root joint of small-articulated animals. In addition, different variations of FABRIK are currently available; Ramachandran and John [36] solve the IK problem using an alternative version of FABRIK with an intersection of circles, while Naour *et al*. [37] uses FABRIK within a global iterative optimisation process. Furthermore, Huang and Pelachaud [38] use a variation of FABRIK in order to solve the IK problem from an energy transfer perspective. They used a mass–spring model to adjust the joint positions by minimising the force energy that is conserved in springs. Recently, Moya and Colloud, in [39], proved that FABRIK can cope with target priorities, adjusting the initial algorithm to deal with joints that have more than two segments. Bentrah *et al*. [40] propose an extension of FABRIK to handle environmental obstacles and conflicts between tasks. The flexibility of the algorithm to be easily adapted into different problems, its easy configuration, its low computational cost and the its performance in closed loops or problems with multiple end effectors make FABRIK a popular and efficient IK solver.

2.3. Joint and Model Constraints

Several biomechanically and anatomically correct models that formalise the range of motion of an articulated figure have been presented. Joint and model constraints are mainly characterised by the number of parameters that describe the motion space and are hierarchically structured. For instance, 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] presented a 3D 'reach cone' methodology using planes, treating the joint limits in the same way as in [41]. Korein in [2] and Baerlocher and Boulic in [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 that 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*. [4] presented analytical and numerical constraints suitable for anthropomorphic limbs; they treat the limbs of 3D characters independently in closed form, 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 by Yamane and Nakamura [44], where multiple-priority-level architectures for combining end effector and centre of mass position control are illustrated.

Model restrictions, because of their complex nature, are simplified or approximated by more than one joint. The most well-known models are the following: *the shoulder model*, a complex model composed of three different joints [45–48]; *the spine model*, a complex arrangement of 24 vertebrae (usually, for simplicity, the spine is modelled as a simple chain of joints [2,49–51]); *the hand model*, which is the most versatile part of the body comprising a large number of joints [52–54]; and *the strength model*, which takes account of the forces applied from the skeletal muscles to the bones [49].

3. CONVERGENCE PROOF

In this section, we study the articulated system and indicate whether a target is reachable or not; if the target is unreachable, we provide optimisation solutions that return the final pose in just one iteration. In addition, we present a proof of convergence when there is a solution available.

The definitions of the terms *articulated body*, *links*, *joints*, *kinematic chain* and *end effector*, as well as their inter-connection, are as described in the original FABRIK paper [1].

3.1. Reachable and Non-reachable Targets

Even in the simple IK problem, where no movement constraint exists, the target is not always reachable because of the chain configuration and the target location. Thus, it is very important to check whether the target is within reach or not; the step to identify the conditions that cause a target position to be unreachable is easy to implement, especially for cases where no rotational or orientation restrictions exist, and can importantly lead to a large saving in processing time.

Assume $\mathbf{p}_i$ is the position of the i-th joint of the kinematic chain, where $i = 1, \ldots, n$. The reachability check proceeds as follows: find the distance between the root and the target, d, and if this distance is smaller than the total sum of all the inter-joint distances, $d < \sum_{i=1}^{n-1} d_i$, where $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$, for $i = 1, \ldots, n-1$, and $\mathbf{p}_i$ is the position of the i-th joint of the kinematic chain, the target is within reach; otherwise, it is unreachable. In the case where the target is within the feasible bounds, the FABRIK algorithm is applied normally; otherwise, the solution will be the direct construction of a line pointing towards the target, while keeping the inter-joint distances constant. This can be performed by applying only the backward step of the algorithm, starting from the root joint, and instead of using the joint positions as the intermediate targets at every step, using the target position. The final pose will be structured in just one iteration. In addition, there are cases where the kinematic chain cannot bend enough to reach the target, even if the latter is within the reachable bounds. Such a case occurs when the kinematic chain consists of a link with size $d_{\max}$ that is larger than the sum of all the remaining links $d_{\max} > \sum_{i=1}^{n-1} d_i - d_{\max}$ and the target is located in a distance $d < 2d_{\max} - \sum_{i=1}^{n-1} d_i$; this is more obvious in Figure 1(a), where a solution cannot be formed if the target is outside the circle with centre $\mathbf{p}_i$ radius *dist*.

Thus, in order to avoid cases where the iterative process enters an endless loop, even though we may never encounter experimentally such a situation, it is advisable to add termination conditions: the first termination condition would be to compare the position of the end effector at the previous and current iterations, and if this distance is less than an indicated tolerance, FABRIK should terminate its operation. Furthermore, in the extreme case where the number of iterations has exceeded an indicated value and the target has not been reached, the algorithm should also be terminated.

3.2. Targets Located on the Kinematic Chain

In the rare case where the kinematic chain is straight and the target is located on the line segment between the points $\mathbf{p}_i$ and $\mathbf{p}'_i$ ($\mathbf{p}'_i$ is the reflection of $\mathbf{p}_i$ in the root joint $\mathbf{p}_1$), as illustrated in Figure 1(b), the algorithm is not able to find a solution; the kinematic chain remains straight, not allowing the end effector to reach the target. Thus, the algorithm should be modified to allow a small degree of sideways bending, say 2° (the well-known random perturbation case) during the backward step of the first iteration, that will change the straight format of the kinematic chain. Thereafter, the algorithm can be applied in its standard form and will return the solution as usual.

3.3. Proof of Convergence

The unconstrained version of FABRIK converges to any given chains/goal positions, when the target is within the reachable bounds and there is a solution available; the algorithm is able to find a solution for any family of inputs with a single chain of at least three joints.

The convergence proof can be divided into two cases. In Case A, the kinematic chain is formed in a straight line, and

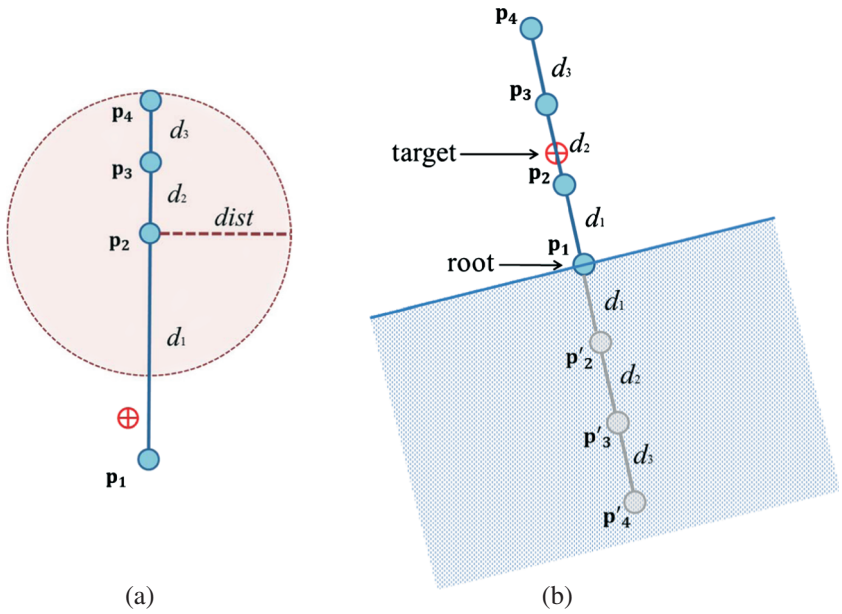


Figure 1. (a) The target is unreachable even if the distance between the root and the target is less than the length of the kinematic chain. (b) The target is located on the kinematic chain: if the kinematic chain is straight and the target lies on the line segment between points $\mathbf{p}_4$ and $\mathbf{p}'_4$, then the algorithm should allow bending during the backward step of the first iteration.

DOI: 10.1002/cav

the target is located on that line segment. This is a common problem of all the IK solvers; the algorithm is able to detect the case and recovers via a random perturbation of a joint to a different position, as described in Section 3.2.

In Case B, the kinematic chain is not in a straight line or the target is not located on the straight line, either because of correction or because it was not initialised in that manner. It is important to note that, if the kinematic chain is not straight and the target is within the reaching bounds, then the kinematic chain will never be in a straight line after a full set of iterations of the algorithm.

An iteration of FABRIK ends when both the forward and backward steps have been completed; by construction of the algorithm, the forward and backward steps are identical in process. Thus, we look at the iteration as two repetitions of one forward step and one backward step; on the forward step, the end effector moves to the desired position (target) and the algorithm is applied till the root, while on the backward step, the root joint moves back to its initial position and the algorithm is applied forward to the target. Even though they are alike, each step of the algorithm has a different target; let the target of the forward step be called the *forward target*, F_t, and the target of the backward step called the *backward target*, B_t. After the forward step, we get closer to F_t, while after the backward step, we get closer to B_t; recall that the IK's target coincides with B_t. Thus, when both steps are completed, the end effector has moved closer to the target, and after a number of iterations, the end effector reaches the target.

Each step is comprised of $n-1$ identical propagations, where n is the number of joints of the kinematic chain. At each propagation step, there are three positions involved, the target $\mathbf{t}$, the 'acting' end effector $\mathbf{p}_i$ and the joint next to the end effector $\mathbf{p}_{i-1}$, where $i = 1, \ldots, n$, as shown in Figure 2(a). The distance between the target and the end effector, called the residual distance, is determined as d_t, while the distance between joints $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$ is defined as d_{i-1}. Each propagation proceeds as follows: the end effector moves to the target position $\mathbf{p}'_i$, the new target position $\mathbf{t}'$ is assigned as the point on the line l_{i-1} that passes through $\mathbf{p}'_i$ and $\mathbf{p}_{i-1}$ and has distance d_{i-1} from $\mathbf{p}'_i$, while $\mathbf{p}_{i-1}$ is allocated as the new end effector position, as shown in Figure 2(b). The new residual distance d'_t is the distance between the new target $\mathbf{t}'$ and the new end effector $\mathbf{p}_{i-1}$. The same procedure is repeated for the rest of the kinematic chain, till the root joint $\mathbf{p}_1$.

Demonstrating that the residual distance between the end effector and the target, at each propagation, is always decreasing means that, at each step, the distances between the end effector and F_t and B_t is becoming smaller, respectively. By generalising this for each iteration, we can show that the end effector moves closer to the target, and we have a converging solution. Thus, we want to prove that

$$d'_t < d_t \tag{1}$$

at each propagation, where d_t is the distance between the end effector and the target in the beginning of the propagation and d'_t is the same distance after the propagation step.

Observing Figure 2(b), we see that a triangle is formed, which is highlighted by the red shape, which is defined by the points $\mathbf{t}$, $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$. Because the length of one side of a triangle is always smaller than the sum of the other two, we can conclude that the distance d_{i-1} is less than the sum of the distances d_t and *dist*, where *dist* is the distance between the joint $\mathbf{p}_{i-1}$ and the target $\mathbf{t}$. Thus,

$$d_{i-1} < d_t + dist \tag{2}$$

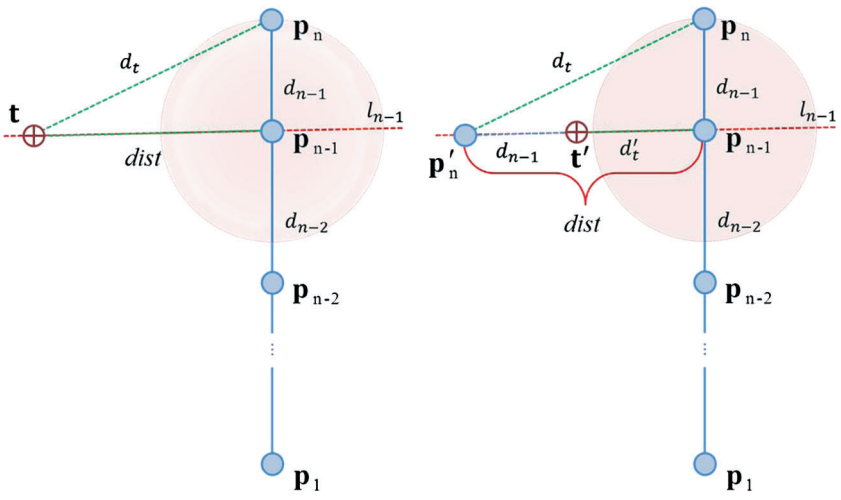


Figure 2. Indicating the joints and the triangle used in the proof of convergence. (a) The joint and target configuration within a propagation step, where three joints are involved, the target $\mathbf{t}$, the end effector $\mathbf{p}_i$ and the joint next to the end effector $\mathbf{p}_{i-1}$, where $i = 1, \ldots, n$. (b) The triangle formed by the points $\mathbf{t}$, $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$ and the residual distance after a propagation step.

and

$$dist < d_t + d_{i-1} \tag{3}$$

After the propagation step ends, the three joints $\mathbf{p}_{i-1}$, $\mathbf{p}'_i$ and $\mathbf{t}'$ lie on the line l_{i-1}, creating three different distances, the distances $dist$, d_{i-1} and d'_t. Therefore, we have the following equalities:

$$d'_t = d_{i-1} - dist \quad \text{if} \quad d_{i-1} > dist \tag{4}$$

and

$$d'_t = dist - d_{i-1} \quad \text{if} \quad dist > d_{i-1} \tag{5}$$

Substituting Equation (2) in Equation (4), and Equation (3) in Equation (5), we have

$$d'_t < d_t + dist - dist = d_t \quad \text{if} \quad d_{i-1} > dist \tag{6}$$

and

$$d'_t < d_t + d_{i-1} - d_{i-1} = d_t \quad \text{if} \quad dist > d_{i-1} \tag{7}$$

respectively.

Thus, we have proved that d'_t is always smaller than d_t, meaning that the distance between the end effector and the target decreases at each propagation step of the algorithm. Because all propagations within a step are alike, we have proved that both steps progress to their own target; the forward step converges towards F_t, while the backward step converges towards B_t. Given that both steps within an iteration are identical, we can safely conclude that the distance between the end effector and the target always decreases and the algorithm converges to a solution. That is always true except in cases where all joints lie on a line, in which case the inequality changes to equality, $d_{n-1} = d_t + dist$, and the algorithm does not progress. However, if there is at least one propagation step of the iterative process where joints are not aligned, the algorithm will converge, and the end effector will move closer to the target. Recall that, if the kinematic chain forms a straight line, it will be handled with a random perturbation of a joint to a different position, as described in Section 3.2.

4. FORWARD AND BACKWARD REACHING INVERSE KINEMATICS IMPLEMENTATION ON SPECIAL CASES

This section aims to show the flexibility of the FABRIK algorithm and how easily it can be adapted to a variety of different problems; we extend the multiple-end-effector version of FABRIK to identify whether a target is reachable or not, offering optimisation solutions that return the final posture in just one iteration. We also adjust FABRIK to cope with closed-loop problems, with chains in which the end effector is not positioned at the end of the chain, and for inter-joint distance control. The solutions presented can be extended or modified to solve different models in a similar manner. Note that the cases described in this section are constraint free, but they can easily be adjusted to consider joint limitations, as described in Section 5.

4.1. Multiple End Effectors

One of the main advantages of FABRIK is its ability to effectively treat chains with multiple end effectors. Most of the available models, especially the human-like models, are comprised of several kinematic chains, and each chain generally has more than one end effector.

As described in [1], the algorithm is divided into two stages: 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; a sub-base joint is a joint that connects two or more chains and is assigned as the 'half-long target'. This will produce as many different positions of the sub-base as the number of end effectors connected with that particular 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 starting from the new sub-base position 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 to the sub-base. Then, the algorithm is applied separately for each chain until the end effector is reached. The method is repeated until all end effectors reach the targets or there is no significant change between their previous and their new positions.

Huang and Pelachaud [38] recently presented an alternative of the FABRIK multiple-end-effector solution using an energy transfer approach; the sub-base position has the same use and properties; however, it is directly obtained from the mass–spring model. The flexibility of FABRIK allows a variety of different solutions based on the problem specifications and model requirements.

4.1.1. Reachable or Non-reachable Targets.

As in the single-end-effector case, it is highly advisable to check whether targets are reachable or not. In the multiple-end-effector models, there are more instances to consider because of the higher-complexity problem that occurs from the multiple chains. In this section, for simplicity reasons, we describe the two-end-effector case; the proposed checking procedure can be extended for cases with more end effectors in a similar manner.

The first step is to check whether targets are within the reaching area; hence, the distance between each target and the root should not exceed the sum of the inter-joint lengths

DOI: 10.1002/cav

between the root and the corresponding end effector. There are three different instances, the case where both targets are outside the reaching area, the case where one target is outside and the other is within the reaching area and the case where both targets are within the reaching area.

In the first case, the FABRIK algorithm can be applied in its usual format; after a few iterations, the termination procedure will be activated, ending in a straight line starting from the root joint, passing through the sub-base and terminating in straight lines towards the targets. However, an optimised version can be applied; during the forward step of the algorithm, move the end effector to the target position, as in the usual form of FABRIK. Thereafter, the new intermediate joints are re-positioned to lie on the line that passes through the end effector and the sub-base (or the root joint, depending on the problem configuration), keeping the appropriate inter-joint distance fixed. This process will return two straight kinematic chains and two new possible positions for the sub-base; the new sub-base position is assigned to be their centroid position. Then, the normal algorithm is applied till the root joint. On the backward step, move the root joint to its initial position and apply the FABRIK algorithm till the new sub-base. Thereafter, the remaining joints are positioned on the line that passes through the new sub-base position and their corresponding target. Both optimised and standard procedures will return the same solution; nevertheless, the optimised version requires only one iteration, saving an important amount of processing time. The optimised procedure is illustrated in Figure 3.

In the second case, where one target is outside and the other target is within the reaching area, there are two possible solutions. The first solution would be to attain one target, leaving the other end effector away from its corresponding target. The second solution is to keep both end effectors away from their corresponding targets with equal distances. The system specifications usually determine what is the right approach to follow; nonetheless, the user can choose to apply a process similar to the aforementioned optimisation procedure, or to let the algorithm enter the the termination procedure (the solution will coincide with one of the aforesaid cases, depending on the shape of the kinematic chain and the target location).

In the last case, both targets are within reach; nevertheless, there are two possible cases: the case where both targets are reachable (the normal multiple-end-effector version of FABRIK can be applied) and the case where it is only possible to reach one target. Even if both targets are within the reach area, nothing assures us that they are reachable; for instance, if the distance between the two targets, as shown in Figure 4, is greater than the maximum possible distance between the two end effectors, then only one target can be reached. Once more, if no system specifications are defined, the user can select whether one of the end effectors will reach the target or not. If no optimisation procedure is applied, neither end effector will reach the target, but both will be the same distance from their corresponding target.

4.1.2. Targets Located on the Kinematic Chain.

As in the single-end-effector case, there is a rare instance where even if a possible solution exists, FABRIK fails to bend the kinematic chain and reach the target; this happens when both targets have equal distance from the root and the kinematic chain between the sub-base and the root is straight. The solution is simple. First, identify the unreachable conditions, as for the configuration in Figure 1. Then, during the forward step of the first iteration of the

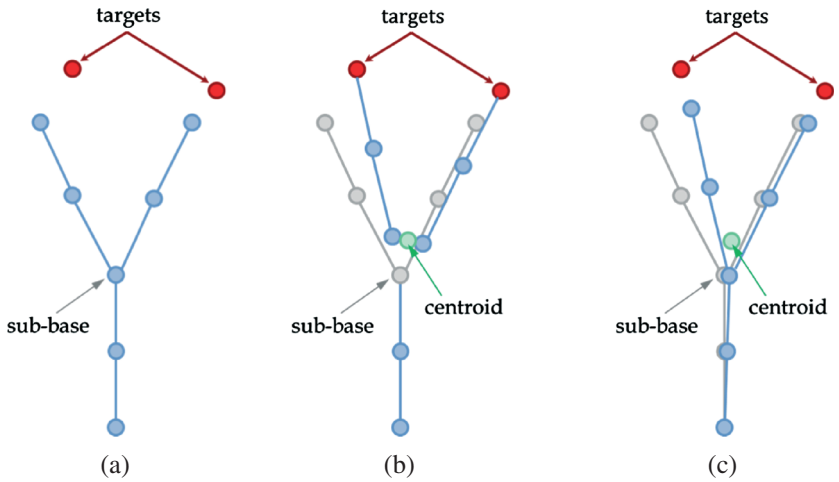


Figure 3. The optimised solution when both targets are not within the reaching bounds. (a) The initial configuration. (b) The forward step of the algorithm: starting from the end effector, the intermediate joints are re-positioned on the line that passes through the end effector and the sub-base. (c) The backward step of the algorithm: starting from the root joint, the joints are positioned on the line that passes through the root and the new sub-base position (centroid). Then, the remaining joints are positioned on the line that passes through the new sub-base and the end effectors, respectively. The final posture is given in just one iteration.

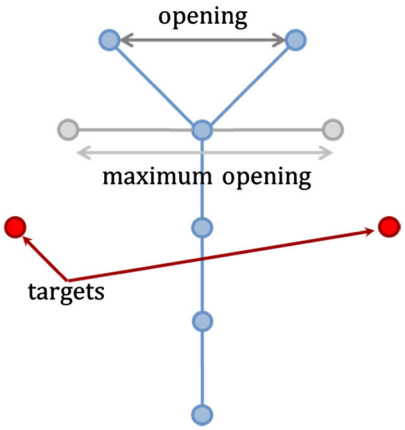


Figure 4. The targets are unreachable, even if they are located within the reaching bounds, as the target opening is larger than the maximum possible opening of the end effectors.

algorithm, let the new sub-base position be given by only one of the kinematic chains (not the centroid). In the extreme case where this position remains on the line segment, then apply a similar procedure to the single-end-effector case and allow a small bend of the chain, taking into consideration the system limitations.

4.2. Closed Loops

A human-like model, as well as many other models, does not only consist of single or multiple kinematic chains; there are kinematic chain structures in the form of a closed loop. The FABRIK algorithm is capable of returning solutions in closed-loop problems, keeping the primary IK assumption that the inter-joint distance should remain unchanged.

Figure 5 demonstrates a simple example of a closed-loop implementation with three joints, where joint $\mathbf{p}_1$ is assumed to be the end effector and it is necessary to move $\mathbf{p}_1$ to the target position. The procedure remains similar to the initial FABRIK algorithm; during the forward step of the algorithm, the end effector moves to the target position (named here as $\mathbf{p}'_1$). Then, $\mathbf{p}_3$ is re-positioned at the line that passes through the joint positions $\mathbf{p}'_1$ and $\mathbf{p}_3$ and has distance d_3 from $\mathbf{p}'_1$. Similarly, the new position $\mathbf{p}'_2$ can be calculated using the line that passes through $\mathbf{p}'_3$ and $\mathbf{p}_2$ and has distance d_2 from $\mathbf{p}'_3$. Obviously, if the kinematic chain consists of more joints, the algorithm will continue till all joints are re-positioned. The backward step performs exactly the same procedure, but this time starting from the other side of the chain, meaning $\mathbf{p}'_2$. The process is then repeated, for as many iterations as needed, until joint positions ($\mathbf{p}_2$ and $\mathbf{p}_3$) at the current iteration do not differ (or differ less than an indicated tolerance) from the previous iteration. Note that the algorithm performs differently if it starts on the opposite side of the loop (e.g. $\mathbf{p}_2$ instead of $\mathbf{p}_3$), and the final solution is therefore not unique.

4.3. Leaf Joints

Forward and Backward Reaching IK can also cope with cases where the 'end effector' is not positioned at the end of the chain (i.e. it is a leaf). The kinematic chains in this case can be divided into two parts. Thus, move the end effector at the target position and then apply FABRIK in both parts of the kinematic chain, simultaneously. Obviously, if it is desirable to keep the root joint at its initial position, FABRIK will iterate to a solution; otherwise, it will return the solution in just one iteration. This procedure is demonstrated in Figure 6.

4.4. Controlling the Inter-joint Length in Kinematic Chains

Another major application of FABRIK is the control of the kinematic chain in models with incomplete, flipped or noisy data. In this section, we present another variant of FABRIK, by adjusting the algorithm to control the inter-joint lengths in kinematic chains. We consider the three most representative and common cases of inter-joint distance control; obviously, different kinematic chains can be handled in a similar way. Note that the true inter-joint distances are known *a priori* for all the cases described in the following.

4.4.1. Joint Control in Serial Chains.

A serial kinematic chain has one root joint, which should not be moved, and a number of noisy joints that need to be re-positioned to meet the fixed inter-joint distance assumption. In this case, FABRIK does not work in an iterative fashion but uses only the backward step. It starts from the root joint and works backward, adjusting each of the internal joints along the way until the last joint. Thus, assume $\mathbf{p}_i$ is the root joint position. The joint update $\mathbf{p}'_{i+1}$ is set as the point on line l_i that passes through $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and has d_i distance from $\mathbf{p}_i$. This procedure is repeated for all the remaining joints until the end of the chain. A graphical representation of an implemented example is given in Figure 7.

4.4.2. Joint Control between Two True Joint Positions.

This case has one root joint and an end joint, where both of them should remain at the same position; in addition, it consists of a number of noisy internal joints that need to be re-positioned to meet the fixed inter-joint distance assumption. The solution can be achieved using forward and backward iterative modes. Before applying this iterative procedure, it is advisable to check whether the target is reachable or not; thus, find the distance between the end joint and the root joint, *dist*, and if this distance is smaller than the total sum of all the inter-joint distances, the target is within reach; otherwise, it is unreachable. If the target is unreachable, but both the end and root joints should remain at their current positions, the algorithm should

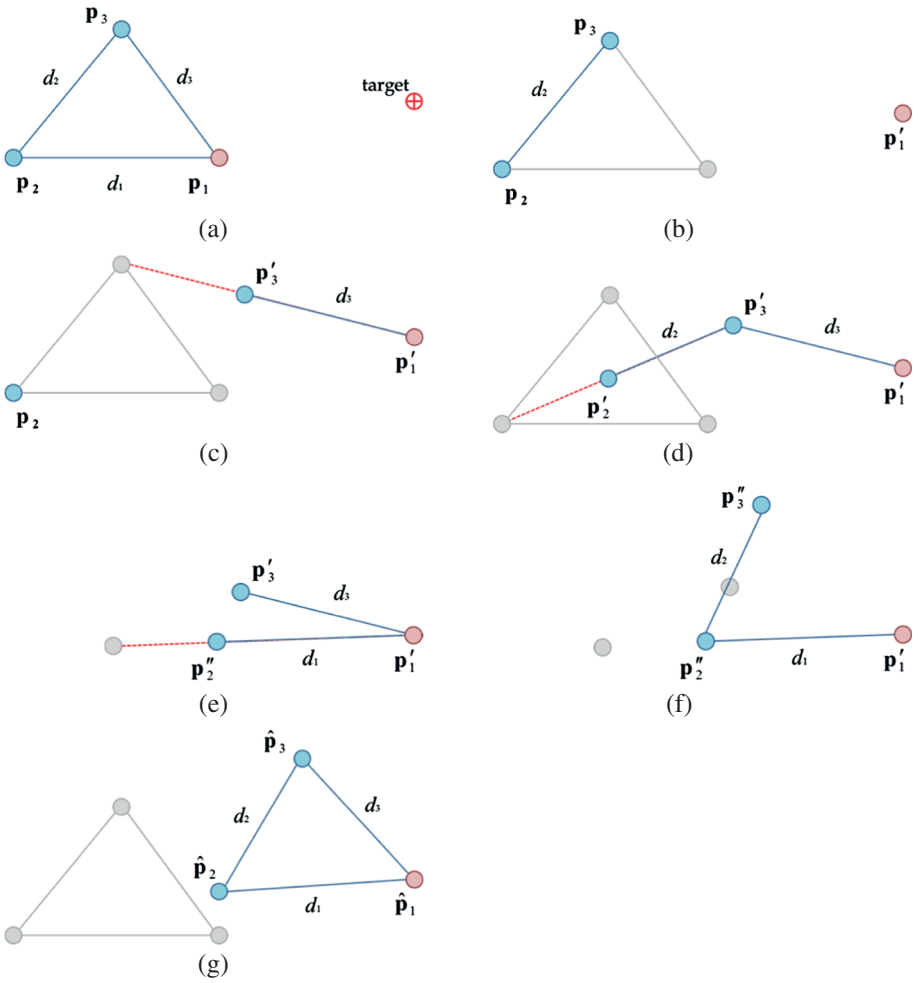


Figure 5. The closed-loop solution: (a) the initial configuration showing the end effector, p_1, in red; (b) the beginning of the forward step; the end effector moves to the target position; (c) p'_3 is re-positioned at the line that passes through the joint positions p'_1 and p_3; (d) the new position of p'_2 lies on the line that passes through p'_3 and p_2 and has distance d_2 from p'_3; (e and f) the backward step of the algorithm, which performs similarly to the forward but this time starting from the other side of the chain, meaning p'_2; (g) the final posture configuration after a number of iterations.

construct a straight line, adjusting the inter-joint distances in such a way that each distance has changed uniformly (extend the length of the chain). In cases where the target is reachable, the normal iterative FABRIK solution is applied starting from the end joint position and moving first forward and later backward, adjusting each of the intermediate joints along the way until the root joint. This procedure is illustrated graphically in Figure 8.

4.4.3. Joint Control in Closed-loop Chains.

The third case is an example of inter-joint distance correction in a simple closed-loop problem; the algorithm is applied consecutively, as shown by arrows (steps) in Figure 9. The solution is divided into five phases; in the first phase, the FABRIK algorithm is applied in a circular form, attempting to correct the noisy joints. Thus, starting from joint $\mathbf{p}_1$, the algorithm re-positions the noisy joints $\hat{\mathbf{p}}_3$ and $\hat{\mathbf{p}}_4$ (Steps 1 and 2), in order to meet the inter-joint distance assumption. Subsequently, the algorithm gives a new temporal value for joint $\mathbf{p}_1$ (in Step 3). The first phase is completed in Step 4, as shown in Figure 9(b). In the second phase of the algorithm, $\mathbf{p}_1$ returns to its initial position, and the algorithm is applied from the other direction, as shown in Steps 5 and 6. The third phase takes into consideration the positions of joints $\mathbf{p}_2$ and $\mathbf{p}_5$; this phase of the algorithm ensures that the distance between these joints and the noisy joints remains constant. Steps 7–10 are cases of the simplest version of a serial chain, where only two joints exist. Lastly, the fourth and fifth phases are similar to the first and second. This procedure is repeated (Steps 7–16) until the positions of the noisy joints between two iterations are identical or their difference is smaller than an acceptable error.

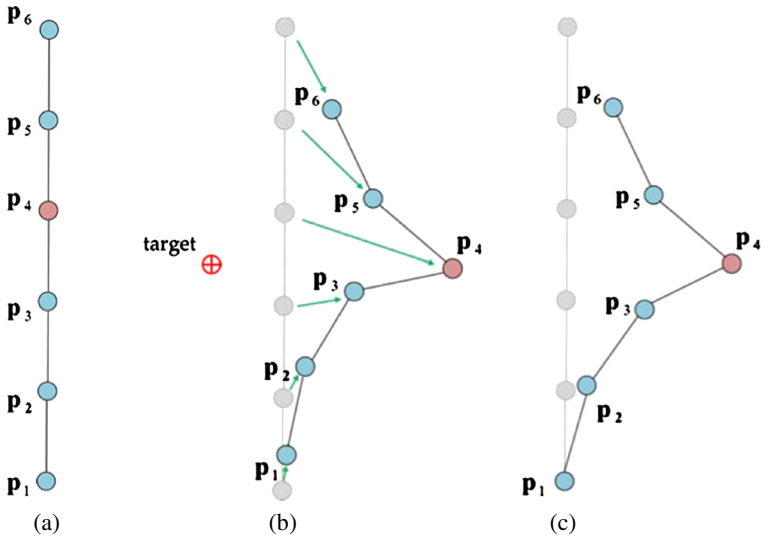


Figure 6. The leaf joint solution: the end effector moves at the target position, dividing the kinematic chain into two parts. Then Forward and Backward Reaching Inverse Kinematics is applied simultaneously to both parts of the kinematic chain. (a) The initial configuration, where $\mathbf{p}_4$ is the end effector. (b) The solution after the first iteration. (c) The solution after a number of iterations, in the case where the root joint must return to its initial position.

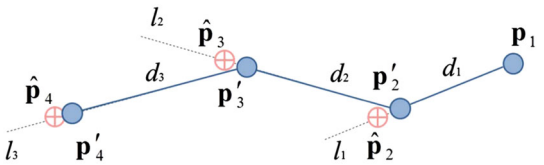


Figure 7. A simple case where the noisy joints are located at the end of the chain. In this example, the serial chain has four joints where $\mathbf{p}_1$ is the root and $\hat{\mathbf{p}}_2$, $\hat{\mathbf{p}}_3$ and $\hat{\mathbf{p}}_4$ positions need to be corrected. Thus, set $\mathbf{p}'_2$ to be the point on line l_1 that has distance d_1 from $\mathbf{p}_1$, $\mathbf{p}'_3$ the point on line l_2 that has a distance of d_3 from joint $\mathbf{p}'_2$ and $\mathbf{p}'_4$ the point on line l_3 that has d_3 distance from joint $\mathbf{p}'_3$.

Different configurations can be treated using the aforementioned approach; for instance, if joint $\mathbf{p}_2$ is not available, Steps 7 and 8 will not be performed. Moreover, if both $\mathbf{p}_2$ and $\mathbf{p}_5$ are not available, then only the first six steps will be applied. By taking advantage of FABRIK's easy adaptation to different models, its flexibility and its low computational cost, a variety of different inter-joint controls can be incorporated for skeletal modelling.

4.5. Self-collision Determination

Collision detection is a significant problem in computer animation, physically based modelling, geometric modelling, robotics and rope simulation. For instance, in human-like models, body segments often collide with others or the main body. Self-collisions in FABRIK can be detected and handled using the technique introduced by Brown *et al*. [55]; all self-collisions must be detected at every iteration, as each one of them will affect the motion of the kinematic chain at the next iteration. More work is needed to ascertain if the FABRIK framework gives any advantages when dealing with self-collision.

5. INCORPORATING JOINT AND MODEL RESTRICTIONS

Human-like models, as well as most legged body models, are comprised of joints having motion restrictions. Thus, in order to keep movements within a feasible range and to reduce any visually unrealistic movements, it is a necessity that IK solvers support model constraints; joint and model limitations are essential in physical simulations and motion tracking. In [1], it is shown that FABRIK supports rotational and orientational constraints. In this section, we extend the rotational and orientational usage of FABRIK, presenting solutions to many different anthropometric and robotic models; we aim to highlight the versatility of FABRIK in various joint models and demonstrate that the algorithm can be integrated and used in complex real-world humanoid models.

A joint is defined by its position and orientation and, in most general cases, has 3 DoFs. The essential feature of a joint is that it permits a relative motion between the two limbs it connects. As explained in [1], a bone rotation can be factored into two components: one 'simple rotation', named *rotational* (2 DoFs), that moves the bone to its final direction vector and another called *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 inter-connected phases, a rotational phase and an

DOI: 10.1002/cav

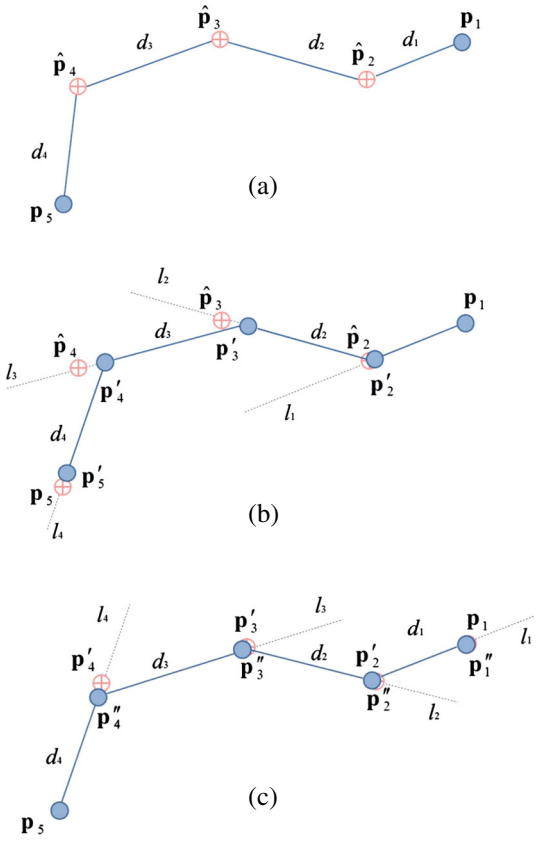


Figure 8. A simple example of Forward and Backward Reaching Inverse Kinematics for the case where noisy data are positioned between the root and end joint positions. (a) The initial configuration of the chain, where $\hat{\mathbf{p}}_i$'s are noisy joint positions. (b) The first phase of the algorithm; the joint $\mathbf{p}'_i$ has been adjusted as the point on line l_{i-1} with distance d_{i-1} from $\mathbf{p}_{i-1}$. (c) The second phase of the algorithm: let the new position of $\mathbf{p}'_5$ be its initial position $\mathbf{p}_5$; repeat the procedure starting this time from the other side of the chain. The algorithm is then repeated for as many iterations as needed until the differences between the initial and current positions of $\mathbf{p}_1$ and $\mathbf{p}_5$ are less than a given tolerance.

orientational phase, contributing equally to the joint restrictions. The operating mode of the algorithm for incorporating constraints is the re-position and re-orientation of the target to be within the allowable bounds; because FABRIK is iterative, the joint restrictions are enforced at each step (propagation) of the algorithm to ensure that the target is within the limits and, if it is not, to guarantee that it will be moved accordingly.

Ramachandran and John [36] developed an alternative of FABRIK that is based on the intersection of circles. They used a geometric approach to control the rotation and orientation of a robot manipulator, while the rotational and orientational limits are controlled independently. Huang and Pelachaud [38] describe how to incorporate joint restrictions in an energy transfer variation of the FABRIK algorithm; they checked all the joints' rotation values after the mass-spring process, and if the rotation in local space is out of bounds, they modify the model to exclude these values.

The joint restriction models presented in this paper 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.

5.1. Anthropometric Joints

In this section, we present the six most common anthropometric joints and describe how to incorporate joint restrictions using FABRIK. Figure 10 shows the six joints discussed in this paper: the *ball-and-socket*, *hinge*, *pivot*, *condyloid*, *saddle* and *gliding* joints, indicating where they can be found on the human body. Note that reference [1] presented the main concept of how to apply joint restrictions using FABRIK, giving as an example the ball-and-socket and hinge joints; in this paper, we demonstrate how joint restriction can be incorporated into the most common anthropometric and robotic joint models. Some more sophisticated anthropometric models (such as the shoulder model) can be formed by a combination of these techniques.

The *ball-and-socket* joint (or spheroidal joint) is a joint in which a ball moves within a socket so as to allow rotary motion in every direction within certain limits. This is the most mobile type of joint in the human body, allowing the greatest range of movements; it allows flexion, extension, rotation, abduction, adduction and circumduction. On the other hand, the *hinge* joint is the simplest type of joint; it can be found in the elbows, knees and the joints of the fingers and toes. It is a bone joint in which the articular surfaces are moulded to each other in such a manner as to permit motion only in one plane/direction about a single axis. It allows flexion and extension movements. The procedure for orientational and rotational control for both the ball-and-socket and hinge joints is described in detail in [1].

A *pivot* joint is one in which a bone rotates around another, permitting only rotating movement; the axis of a convex articular surface is parallel with the longitudinal axis of the bone. It can be found in the neck, allowing a side-to-side turn of the head. Given that the pivot joint allows only rotational movements, it is a requirement that the target should be moved to lie on the same line with the joint, as illustrated in Figure 11. Thus, the target $\mathbf{t}$ is projected on line l_1 that passes through joints $\mathbf{p}_1$ and $\mathbf{p}_2$; the new position of joint $\mathbf{p}_3$ is positioned on line l_1 and has distance d_2 from $\mathbf{p}_2$, while its final orientation is given by the target and should be kept within the allowed limits. The orientation can be adjusted by applying the orientational procedure of the FABRIK algorithm, as described in [1].

A *condyloid* joint (also called ellipsoidal joint) is an ovoid articular surface that is received into an elliptical cavity. This permits biaxial movements, that is, forward–backward and side to side, but not rotation. The condyloid

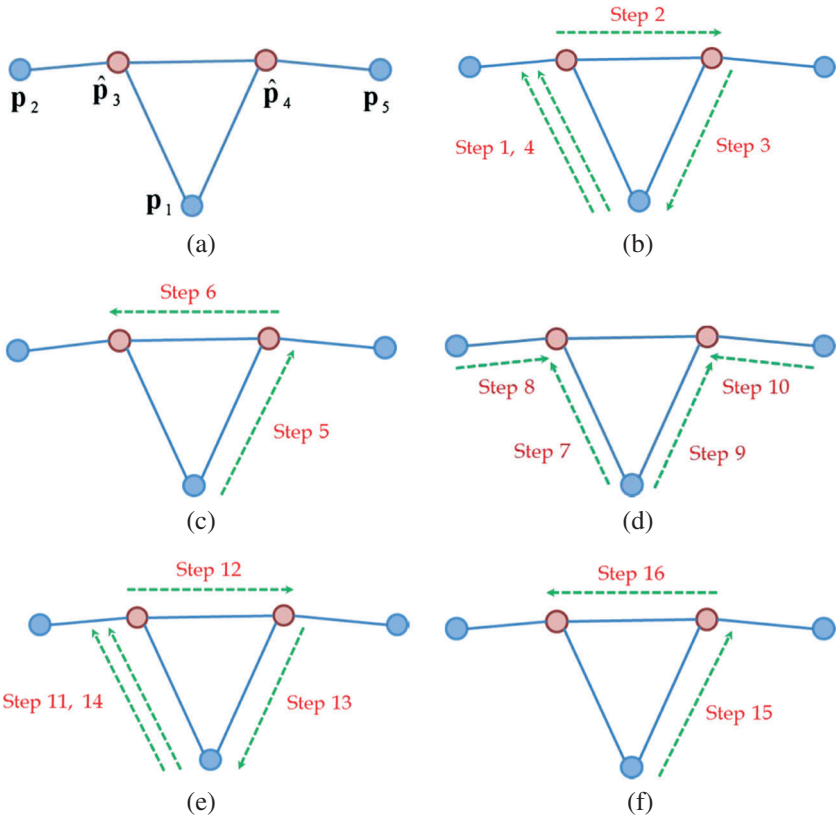


Figure 9. The Forward and Backward Reaching Inverse Kinematics solution for inter-joint control in closed-loop problems. (a) The initial configuration of the problem. (b) The first phase of the algorithm, which is in a circular form, re-positions the joints of the closed loop. (c) The second phase of the algorithm. (d) The third phase, where the other true joint positions contribute to correction of inter-joint distances. (e and f) The last two phases that are identical to Phases 1 and 2, respectively.

joint allows flexion, extension, adduction, abduction and circumduction. Examples of condyloid joints are the wrists and the radius carpal. Condyloid joints are dealt with in a manner similar to the ball-and-socket joint, without applying the orientational procedure because no rotational movement is allowed.

In a *saddle* joint (also known as a sellar joint), the opposing surfaces are reciprocally concave–convex. It can be treated exactly in the same way as the condyloid joint, but different angle limits describe the allowable bounds. A saddle joint supports flexion, extension, adduction, abduction and circumduction; similarly, no axial rotation is permitted. The thumb is an example of a saddle joint.

A *gliding* joint (also known as a plane joint) is a synovial joint that, under physiological conditions, allows only gliding or sliding (sideways) movements. The solution for the gliding joint requires a relaxation of some of the conditions of the IK problem. The solution is demonstrated in Figure 12. The target is first projected onto the joint plane; if the projected position is within the joint limits, then re-position $\mathbf{p}_3$ on the line that passes through the target and its projection with distance d_2 from $\mathbf{p}_2$. In the case that the projected position is out of bounds, move the projected target position accordingly to be within the allowable limits and follow the same procedure. It is important to note that no axis rotation is permitted.

5.2. Robotic Joints

A prismatic joint is a 1-DoF kinematic pair mainly used in robotics and mechanisms. Prismatic joints provide a single-axis sliding function with the axis of the joint coincident with the centre line of the sliding link. Because any prismatic form can be used for the elements of a sliding pair, it does not have a specific axis (as a turning pair does), but merely an axial direction.

This family of joints changes the size of the links connecting the joints; thus, it is generally not supported by most IK solvers. However, FABRIK can be adjusted to deal with prismatic joints, if details about the permissible variations of link sizes are available or the maximum and minimum sizes for each link are *a priori* known.

The solution described in this paper has minimum and maximum values for d_i; this value can be adjusted on the first step or after the algorithm identifies that the target is unreachable, according to the user preferences or the model specifications. For instance, the user may choose to have the mean of the length as the initialisation size and adjust

DOI: 10.1002/cav

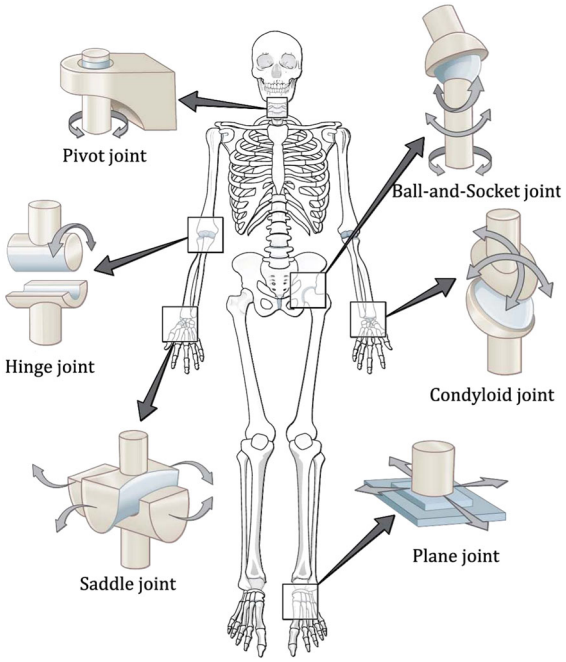


Figure 10. The six most common anthropometric joints. The picture is taken from http://anatomisty.com/

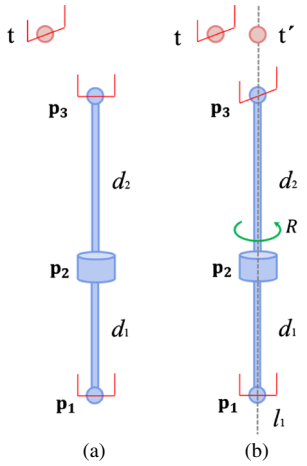


Figure 11. The pivot joint. (a) The initial configuration and (b) the Forward and Backward Reaching Inverse Kinematics solution. The target is projected onto line l_1 that passes through joints $\mathbf{p}_1$ and $\mathbf{p}_2$, and joint $\mathbf{p}_3$ is re-oriented to satisfy the rotation of the target. This picture is taken from [1].

the length according to whether the target is reachable or not (very close and far away). Figure 13 shows a demonstration of the algorithm in cases where the size changed to the maximum allowed in the beginning of the algorithm (Figure 13(b–d)) and the case where the length changed after the algorithm identifies that the target is unreachable (Figure 13(e and f)). Obviously, in the second case, the size of the chain will not be the maximum possible, and the final pose will be in a straight line.

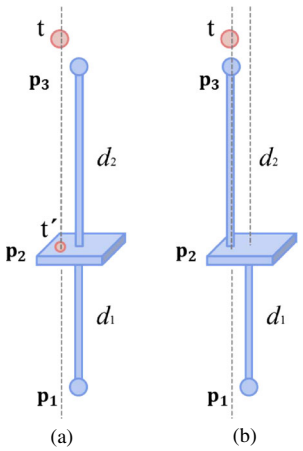


Figure 12. An example of the gliding joint solution. (a) The initial configuration and (b) the solution. The target is projected onto the joint plane; if it is within limits, then locate $\mathbf{p}_3$ on the line that passes through the target and its projection with distance d_2 from $\mathbf{p}_2$, with no axis rotation; otherwise, move the projected target position accordingly to be within limits and follow the same procedure.

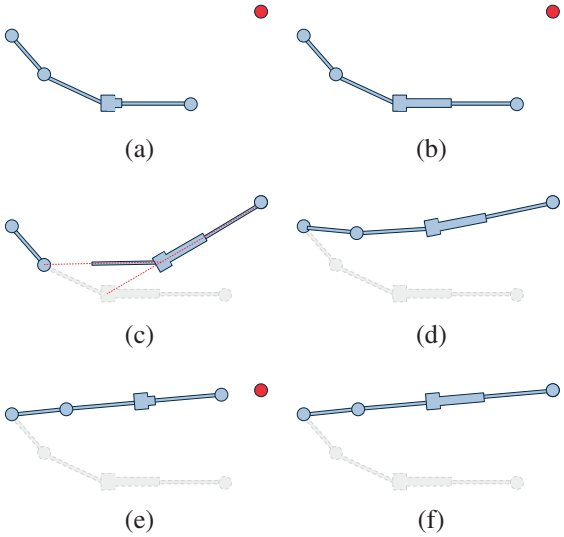


Figure 13. An illustration of Forward and Backward Reaching Inverse Kinematics (FABRIK) for a prismatic joint. (a) The initial configuration. (b) Because the target is unreachable, the length of the link takes its maximum value. (c) The FABRIK algorithm is now applied in the normal way. (d) The final posture when the link takes its maximum value. An alternative approach of the prismatic joint: (e) first apply FABRIK, and because the target is unreachable, the final pose will be a straight line; (f) and then, adjust the length of the link to reach the target (the final posture is a straight line).

5.3. Self-body Dynamics

The main assumption of the IK problem is that the inter-joint distances remain constant over time. However,

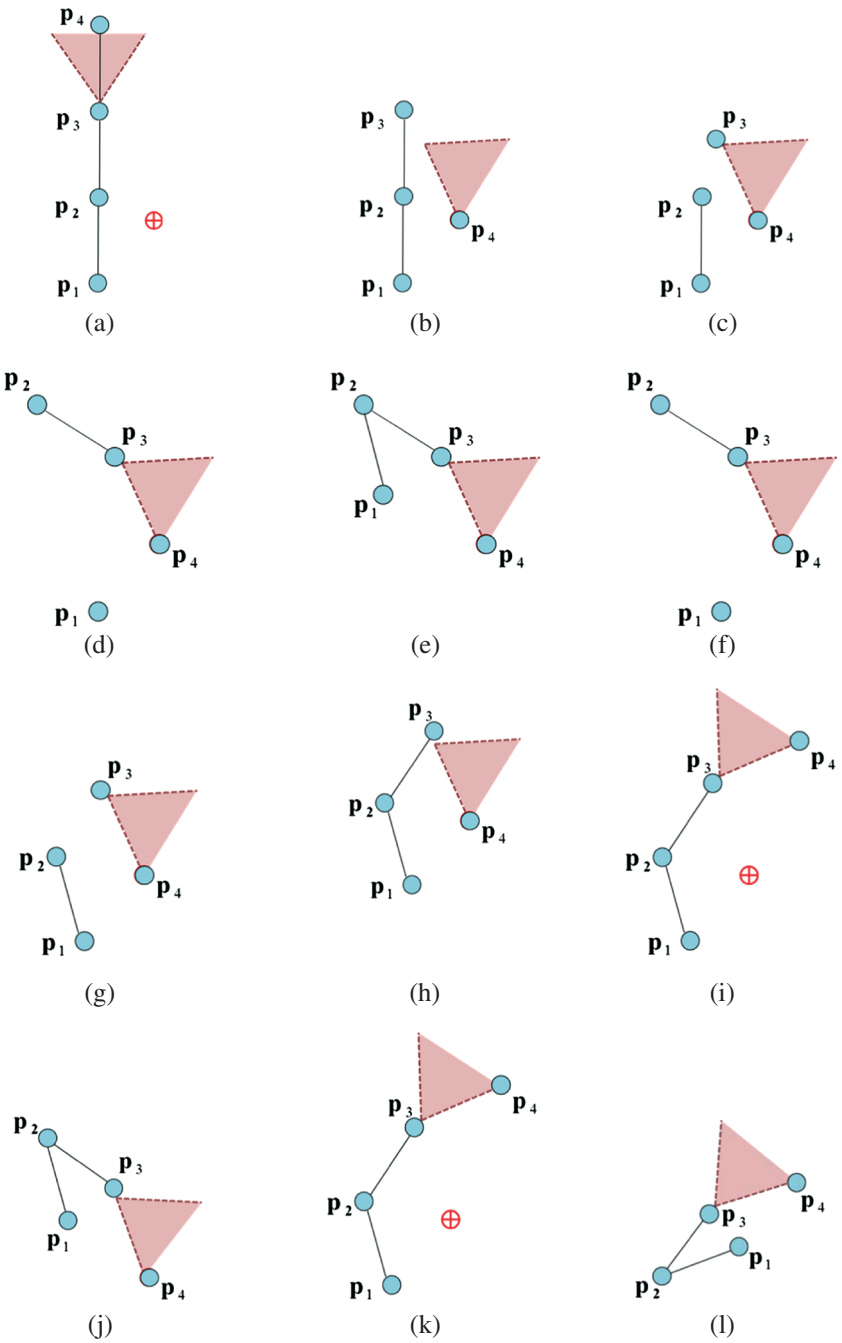


Figure 14. The deadlock case. (a) The initial configuration, where the bounds of joint p_3 are indicated by red shading. (b) The forward step of the algorithm starts; thus, p_4 is moved to the target position; (c) p_3 is positioned on the line that passes through p_4 and its previous position, (d) p_2 is positioned by taking into consideration p_3's constraints and (e) p_1 is placed on the line that passes through p_2 and its previous position. Then, in (f–i) is shown the backward step of the algorithm, where the root joint p_1 returns to its initial position and the rest of the joints are placed based on their joint constraints. If this procedure is repeated, the algorithm enters a deadlock situation, as shown in (j) during the forward and in (k) during the backward step. The algorithm will never find the exotic solution, as shown in (l).

especially in human models, bones allow a small fluctuation in links' length size that most researchers in the literature use a mass–spring model to describe. Therefore, there is a need to allow the IK solvers to increase and decrease the inter-joint distance according to the user or model requirements. Huang and Pelachaud [38] proposed

DOI: 10.1002/cav

a mass–spring-based variation of FABRIK to allow small alterations on the link lengths; however, their energy transfer IK solver suffers from oscillations that derive from using the mass–spring system. In this section, we outline a self-body dynamic approach that uses FABRIK to replace the mass–spring model, in cases where the bone length is allowed to change (such as extreme extensions of the arm to reach a target).

The solution is similar to the prismatic joint case: the link length is slightly adjusted, based on the model specifications, in order to allow the end effector to reach the target. The approach utilised keeps the initial length of the links, and only when their target is unreachable does it change the size; in that manner, the final posture is a straight line. Nevertheless, further investigation is needed to find out when the chain should be expanded or shrunk, according to the human anatomy and physiology, and how much it can be expanded without the user perceiving it [56].

5.4. Dealing with the Deadlock Situation

The constrained version of the FABRIK algorithm encounters a deadlock situation when the kinematic chain is small in size and the joints close to the end effector have strict constraints. The algorithm is then incapable of finding the exotic solution because the kinematic chain was unable to bend enough and reach the target. This is due to the structure of the algorithm in which each joint is treated independently; the algorithm does not take into consideration the restrictions in the previous (parent) or next (child) joints in order to push, if possible, the kinematic chain to bend further in the current joint.

Figure 14(a–k) illustrates the deadlock situation, and Figure 14(l) shows the desirable solution. Essentially, because of rotational constraints, the chain goes into a deadlock mode, when each iteration simply repeats the previous position until the maximum number of tries is reached.

The solution to the problem is simple. First, check if the target is within the reachable area. It is important to note that no optimisation is recommended when the target is out of reach and joint limitations exist, as there are joint constraints that may be violated. If the target is not within the reaching area, the algorithm enters the termination process, and the iterative procedure is terminated. If the target is within reach but the kinematic chain cannot bend enough to reach the target, check whether the distance between the end effector and the target becomes smaller after each iteration. If not, there is a chance that the algorithm has entered a deadlock situation, or the constraints do not allow the chain to bend more. Thus, during the backward step of the first iteration of the algorithm, bend the chain by 2° (or 15°, 30°, etc., based on the user requirements or the system specifications) in the opposite direction of the target, in order to allow other joints to bend more, and and then continue using the FABRIK algorithm in its normal state. If the end effector still does not reach the target, allow more bending till the target is reached or a full turn of 360° is completed. In that way, the algorithm pushes the parent joints to their limits, allowing more flexion for the child joints. Having a small rotation, for instance a rotation of 2–5°, is more effective compared with cases with larger degrees of rotation, ensuring a smooth motion without discontinuities and/or oscillations. This procedure does not affect the efficiency of the FABRIK algorithm, especially in cases where the end effector is simply tracking the target in time with high frequency rate.

5.5. Human-like Model

In this section, we describe an example of how the constraints mentioned in Sections 4 and 5 can be applied to a human-like model. The purpose of setting up the model is to show evidence of the flexibility of the algorithm and illustrate how it can easily be adapted to sequentially and hierarchically structured humanoid models. Obviously, the human-like model can be constrained further by taking into consideration physiological and anatomical constraints; for a well-designed human model, it is essential to study all joint types and their constraint parameters.

5.5.1. The Human Geometry.

We have implemented a human model based on the geometry described in Figure 15. In this example, the human model consists of nine ball-and-socket joints, four hinge joints (knees and elbows), one pivot joint (neck) and five oriented end effectors that are defined by three markers. The implemented model comprises two oriented targets to be tracked by the two hands, a serial chain with an oriented end effector that is tracked by the head and two more oriented targets that are tracked by the feet.

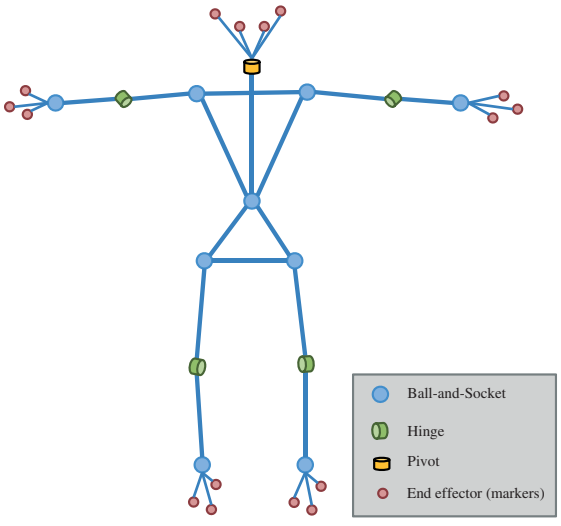


Figure 15. Human configuration and the joint types used in our experimental model.

5.5.2. Hierarchical Structure.

The flexibility of the FABRIK algorithm enables implementation of many different approaches to solve this problem. In this work, we show a simple solution, where the structure of the human skeleton is divided into smaller kinetic chains, which are then sequentially adapted into the body posture in a hierarchical order. The adaption hierarchy may vary between different models. Let us assume that, in this example, the two hands track the moving targets, while the feet should remain constant at their initial position. Figure 16 shows the hierarchical structure of the humanoid model used in our experimental results. First in the hierarchy are the hand chains, which have the end effectors that track the targets. Figure 16(a) shows the two hand chains, while Figure 16(b) the upper triangle of the main body. As the end effectors start to move around, FABRIK re-locates the hand chains and the upper triangle, to meet the inter-joint distance assumption. This process moves the sub-base (as presented in Figure 16(c)) of the model to its new position. The algorithm is iterative, and it will be repeated till the distance between the targets and the two end effectors is smaller than an indicated tolerance.

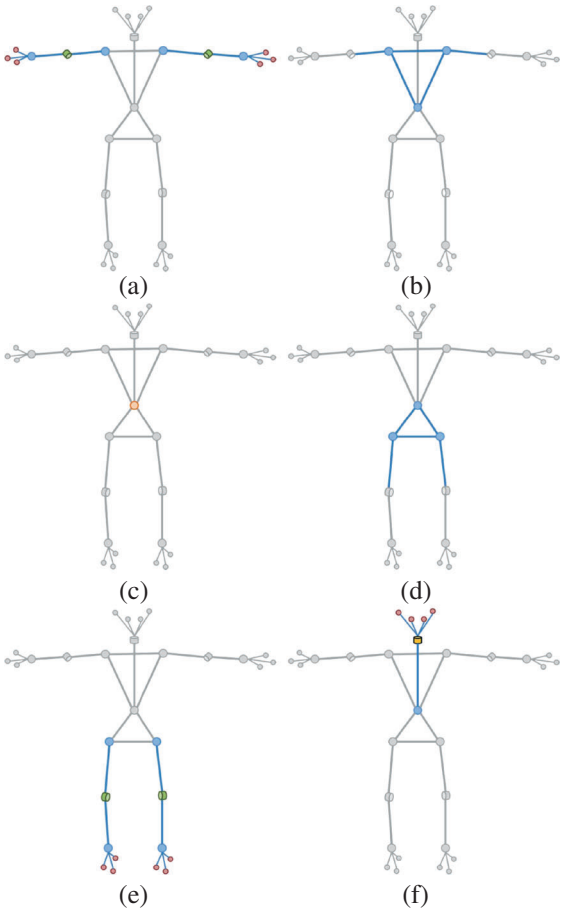


Figure 16. Human structure: the different chains of the human model in a hierarchical order; because hands are the end effectors controlling the movement of the character, they are first in the hierarchy. (a) The upper kinematic chains (hands). (b) The upper triangle of the main body. Panels (a) and (b) are inter-connected; thus, the first iteration will sequentially move all the joints. The left and right hands work simultaneously. (c) The sub-base, (d) the lower triangle of the main body, (e) the lower kinematic chains (feet) and (f) the chain of the head.

In the second stage of the algorithm, FABRIK is applied to re-position the three remaining chains. First, it is applied to control the lower triangle of the main body, as shown in Figure 16(d), and then the leg chains, as shown in Figure 16(e). This is also an iterative process because, in this model, it is assumed that the feet positions should remain constant. Finally, FABRIK is incorporated to re-position the head chain; this is not an iterative process because it is treated as a serial chain (Section 4.4.1).

Obviously, if the humanoid model is different, for instance, there are four end effectors (the two hands and two feet), the upper and lower body will return two different positions for the sub-base joint; the mid-point of these two positions will be chosen, as for the sub-base in the multiple-end-effector case, and the algorithm will continue iteratively till the distance between the targets and end effectors is less than an acceptable threshold.

The proposed human-like model has been adapted to the Aristidou and Lasenby framework [30], which has been integrated in the PhaseSpace Inc. mocap software. A similar implementation for hand modelling and reconstruction is presented in [28], where joint constraints have been incorporated using FABRIK on a hand. Similarly, different models can be designed for various applications, ensuring that the targets will always remain within the allowable space.

6. EXPERIMENTAL RESULTS

In this section, we present the results of various implementations and tests of the aforementioned optimisations and joint and model structures for validation purposes. The experiments are demonstrated using a humanoid model containing 14 partially constrained joints with five end effectors. The IK problem for the humanoid model is solved sequentially using closed loops in a predefined hierarchical order, as described in Section 5.5.

Forward and Backward Reaching IK, as shown in [1], performs faster than other IK solvers that has been tested; in this paper, we ensure that, when the target is unreachable and the optimisation step has been applied, the requisite processing time has been reduced by a factor of 50–60 times, depending on the termination condition and the permitted error tolerance, while we add only 2% to the processing time in cases where the target is within reaching bounds. Furthermore, it is guaranteed that the kinematic chain will be straight, compared with when no optimisation has been applied, where the straightness of the chain is based on the selected error tolerance.

DOI: 10.1002/cav

The human-like model described in Section 5.5 is implemented for testing purposes; even if only limited information about the tracking pose is available, the proposed model returns visually natural solutions that satisfy the user or character constraints. Having available only the position and orientation of the end effector of the model and the initial skeletal configuration, we were able to successfully track the targets and reconstruct and animate the avatar's motion, while the model remains within a feasible set of postures. Figure 17 shows three different examples of target tracking where the end effectors (points of impact) move manually and the human model follows smoothly, retaining the user-pre-defined constraints. In the first case, as shown in Figure 17(a), the hands (the only two end effectors of the model) were able to follow the targets without affecting the posture configuration of the character. This happened because both targets are within the reaching limits and have similar distance from their respective end effectors. In contrast, Figure 17(b and c) shows the cases where the model posture changed because targets were not within the reachable bounds. The end effector pulled the entire body in the direction of the targets, so that it can eventually reach the targets. Nevertheless, in all cases, the algorithm returned a smooth

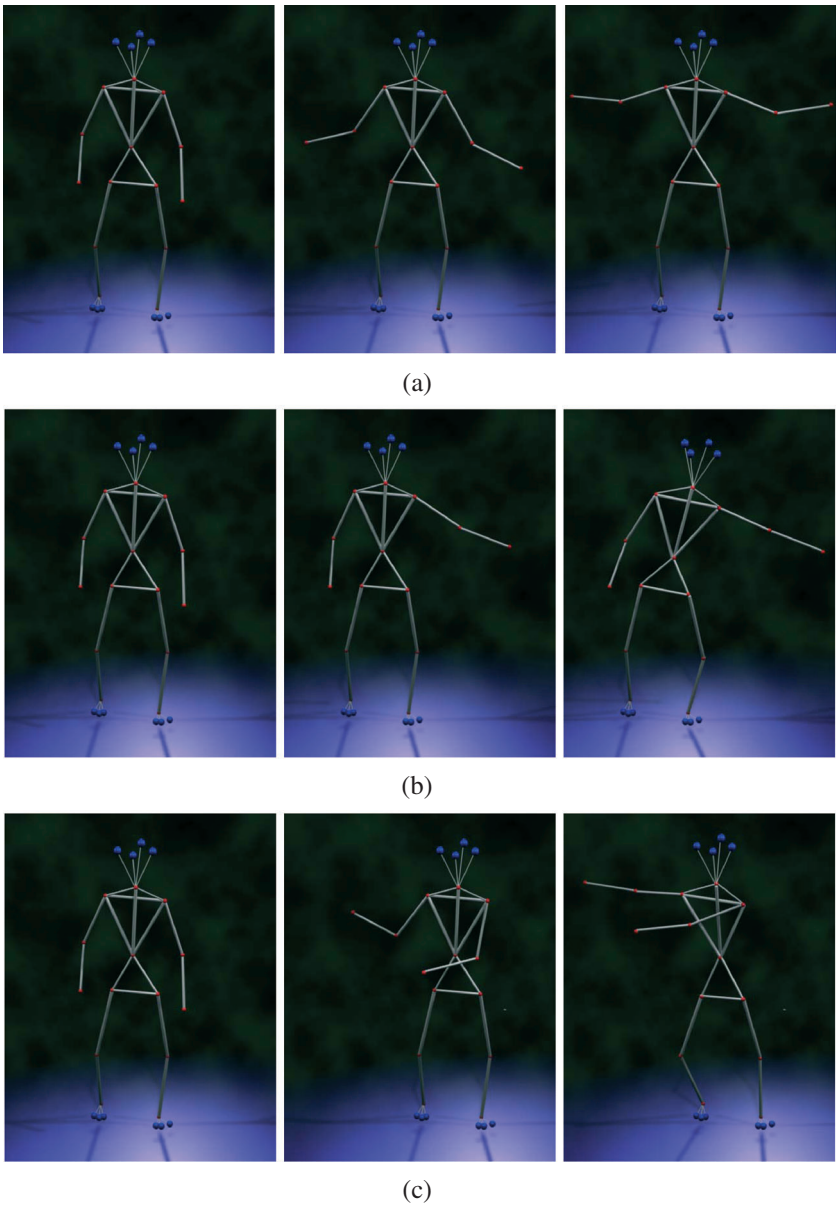


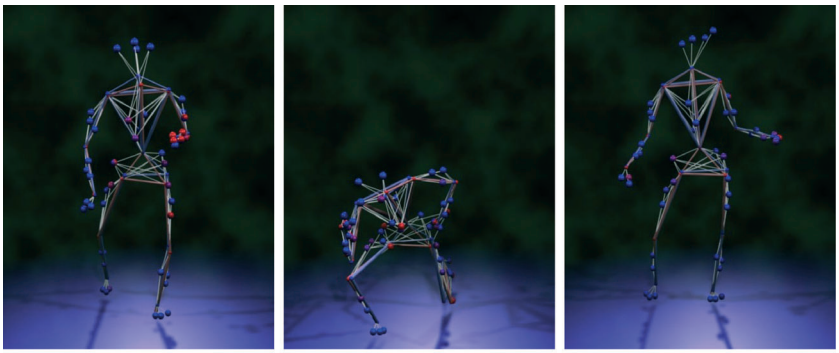
Figure 17. An example of Forward and Backward Reaching Inverse Kinematics implementation; the hands of the avatar smoothly track the moving targets, while the character posture remains within a feasible set defined by the joint and model constraints.

solution, ensuring that the joint and model limitations are satisfied over time. The algorithm performs equally effectively in cases of four, five or even more targets and/or end effectors.

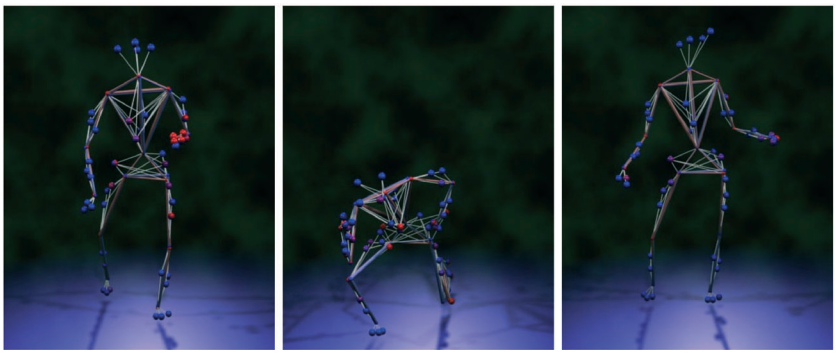
Figure 18 shows an example where the proposed human-like model was incorporated within a mocap framework. The human skeleton was reconstructed using the Cameron and Lasenby method [58], which requires three markers available at each limb segment. In order to evaluate our method, we have artificially deleted a number of markers for an extended period so as to add noise in the joint estimation. The large tracks of deleted marker data were estimated, and the joint positions were calculated using [57], as shown in Figure 18(a). The joint positions were further corrected using the proposed human model, as shown in Figure 18(b). It can be observed that the error between the true and reconstructed (after deletions) skeletons has been reduced when FABRIK was applied, as FABRIK ensures that the inter-joint distance remains constant over time; this is more obvious when looking at the hip and shoulder joints of the skeletons in Figure 18. The reconstructed postures, as also seen in the supplementary video, are animated smoothly without discontinuities, abnormalities or oscillations, resulting in an average joint error of 1.02 cm, compared with 3.37 cm in cases where FABRIK was not applied.

In order to push the FABRIK reconstruction ability to its limits, we implemented FABRIK so as to animate a constrained human model by tracking five end effector positions (also named control points), the two hands, the two feet and the head. The mocap system returns the positions of the control points and the root joint (see sub-base in Figure 16(c)), while the proposed humanoid model tracks the movement of the character; note that the human skeleton, meaning the inter-joint distances, is known *a priori*. This is very useful to efficiently animate interactive characters using low-cost mocap systems, especially in the computer game industry. Figure 19(a–c) shows different examples where the tracking character kneels, kicks and punches, respectively. FABRIK ensures that the end effectors track the targets and the rest of the body satisfies its constraints, resulting in visually natural postures.

Figure 20 shows a snapshot of the FABRIK implementation compared with the true joint positions, as returned by the mocap system. It can be observed that the use



(a)



(b)

Figure 18. Snapshots of Forward and Backward Reaching Inverse Kinematics (FABRIK) implementation within a motion capture framework in cases where a number of marker were artificially deleted. (a) The joint positions were estimated using **[57]**. (b) The joint positions were estimated using **[57]** and corrected by utilising FABRIK, which ensures that the inter-joint distance remains constant over time. The true (before the deletions) skeleton in both cases is given in blue, while the reconstructed in red.

DOI: 10.1002/cav

of such a small number of control points may lead to wrong joint estimates, as well as the production of artefacts, especially in the shoulders and knees; however, the reconstructed skeleton respects the human model constraints, even if some joints are not identical to the true joint positions. The implemented model results in an average positioning error of 1.66 cm; the joint with the higher average error was the left shoulder with 7.6 cm, while the knees and the elbows have an average error of 2.1 cm. The errors occurring in joint estimation, as well as the artefacts and oscillations, can be eliminated by adding more control points, such as the hips and shoulders. Using more control points will allow the kinematic chain to get into smaller closed loops, removing the noise presented in the intermediate joints. In such a case, the artefacts are significantly reduced, making the reconstructed joint average error comparable with the case where the human skeleton is animated using the mocap system. The proposed method could be implemented in applications where a fast interactive control is needed (e.g. in Microsoft Kinect™) or in cases where it is desirable to animate a human character using only a limited number of control points.

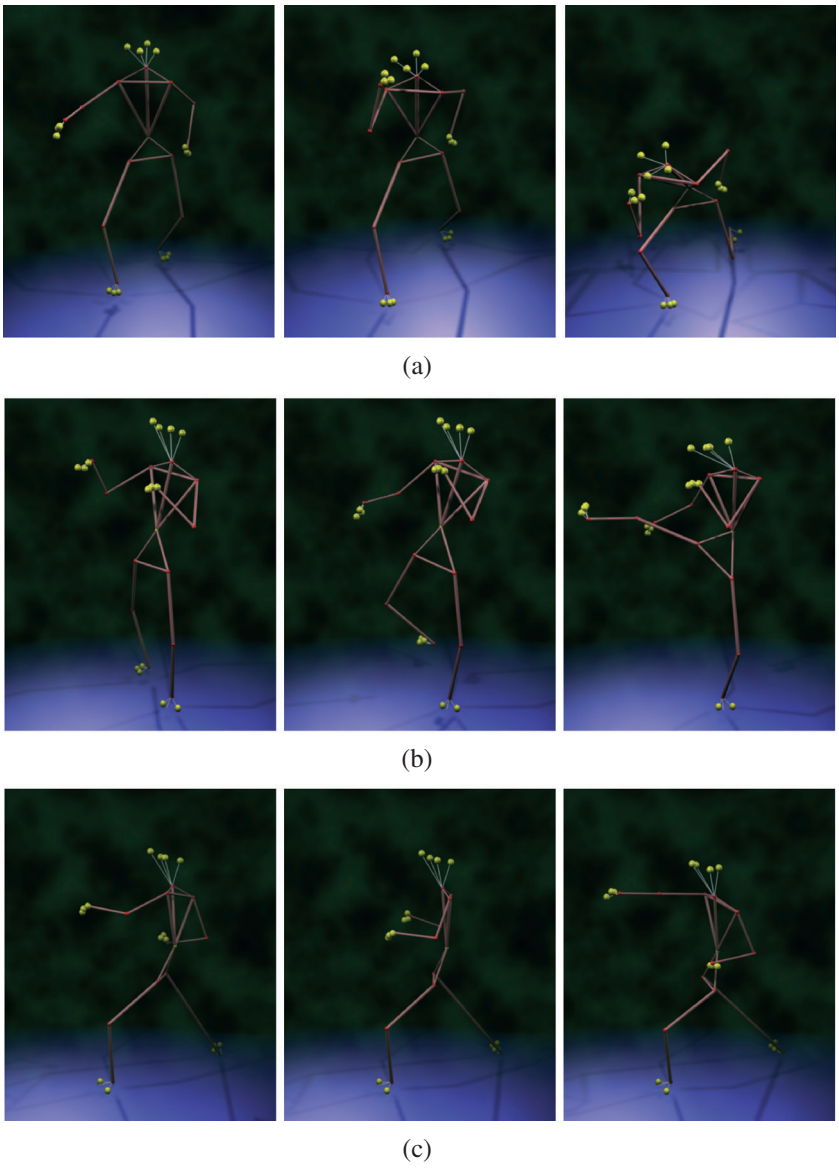


Figure 19. Snapshots of Forward and Backward Reaching Inverse Kinematics implementation when the human skeleton is controlled by five end effectors. The end effectors, which are animated using motion capture data, are in yellow, while the reconstructed joints are in red.

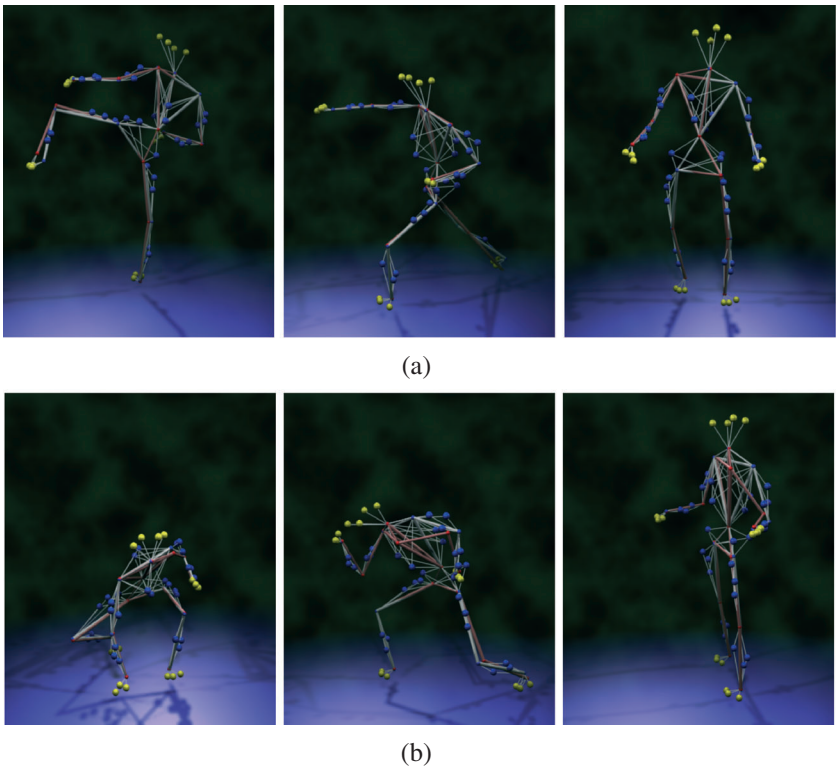


Figure 20. Snapshots of the Forward and Backward Reaching Inverse Kinematics skeleton reconstruction compared with the true joint positions, as returned by the motion capture system. The end effectors are in yellow, the true joint positions and the markers are in blue, while the reconstructed joints are in red. It can be seen that in (a) the skeleton has been successfully tracked and reconstructed, while in (b) there are wrong estimates, especially on the shoulders. Nevertheless, the reconstructed skeleton respects the user and character constraints, even if some joints differ from their true value.

7. CONCLUSIONS

Forward and Backward Reaching IK is a simple, fast and reliable solver that uses an iterative method with points and lines to solve the IK problem. It divides the problem into two phases, a forward and backward reaching approach, and it is able to support a variety of different anthropometric and robotic joint types. Even though it is a recent algorithm, it has become a popular IK solver because of its simplicity and flexibility, enabling easy adaptation into different problems and models. In this paper, we present a framework that detects the cases where the target is unreachable or not; in addition, we prove that FABRIK always converges to an answer if there is a solution available. Furthermore, we introduce optimisation solutions when the targets are unreachable, in kinematic chains with both single or multiple end effectors, converging to a solution in just a single iteration. FABRIK has been also adjusted to control a fixed inter-joint distance under noisy or flipped data, in serial or closed-loop chains. To the best of our knowledge, FABRIK supports all the rotational and orientational joint limits by re-positioning and re-orienting the target at each step. Consequently, we have incorporated joint constraints into different anthropometric and robotic joints, showing that FABRIK is easily adaptable to different models and problems. A humanoid model is implemented, where FABRIK has been applied hierarchically and sequentially to track multiple targets and ensure fixed inter-joint distance. FABRIK remained executable in real time, achieving good results in complex structured humanoid models with joint limitations and closed-loop forms. The efficiency of the proposed methodology was further tested by tracking and animating a human skeleton that uses only five control points, demonstrating its use in applications where fast interactive control is needed.

Future work will see the introduction of the aforementioned approaches in more complex models that take into consideration physiological and anatomical constraints. Further investigation is also needed to consider small changes in limb size, so as to add extra realism in human animation.

ACKNOWLEDGEMENTS

This project is co-financed by the Office of Naval Research Global (N62909-13-1-V090), the European Regional Development Fund and the Republic of Cyprus through the Research Promotion Foundation (DIDAKTOR/0311/73).

The authors would also like to thank Dr Chrysis Georgiou (University of Cyprus), Mr Tracy McSheery and Dr Kan Anant (PhaseSpace Inc.) for their valuable help.

REFERENCES

1. Aristidou A, Lasenby J. FABRIK: a fast, iterative solver for the inverse kinematics problem. *Graphical Models* 2011; **73**(5): 243–260.
2. Korein JU. *A Geometric Investigation of Reach*. MIT Press: Cambridge, MA, USA, 1985.
3. Paul RP, Shimano B, Mayer GE. Kinematic control equations for simple manipulators. *IEEE Transactions on System, Man and Cybernetics* 1988; **11**(6): 1398–1406.
4. Tolani D, Goswami A, Badler NI. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models* 2000; **62**(5): 353–388.
5. Fu Z, Yang W, Yang Z. Computer 'experiments' on classical fluids. I. Thermodynamic properties of Lennard–Jones molecules. *Journal of Mechanisms and Robotics* 2013; **5**: 1–7.
6. Balestrino A, De Maria G, Sciavicco L. Robust control of robotic manipulators. In *Proceedings of the 9th IFAC World Congress,* volume 5, 1984; 2435–440.
7. Wolovich WA, Elliott H. A computational technique for inverse kinematics. *The 23rd IEEE Conference on Decision and Control* 1984; **23**: 1359–1363.
8. Baillieul J. Kinematic programming alternatives for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation,* volume 2, March 1985; 722–728.
9. Wampler CW. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *Proceeding of the IEEE Transactions on Systems, Man and Cybernetics* 1986; **16**(1): 93–101.
10. Nakamura Y, Hanafusa H. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Trans. ASME, Journal of Dynamic Systems, Measurement, and Control* 1986; **108**(3): 163–171.
11. Buss SR, Kim J-S. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools* 2005; **10**(3): 37–49.
12. Kenwright B. Inverse kinematics with dual-quaternions, exponential-maps, and joint limits. *International Journal on Advances in Intelligent Systems* 2013; **6**(1 and 2): 53–65.
13. Fletcher R. *Practical Methods of Optimization,* 2nd edn. Wiley-Interscience: New York, NY, USA, 1987.
14. Courty N, Arnaud E. Inverse kinematics using sequential monte carlo methods. In *Proceedings of the V Conference on Articulated Motion and Deformable Objects,* volume 5098. LNCS, Mallorca, Spain, July 9-11; 1–10.
15. Hecker C, Raabe B, Enslow RW, Deweese J, Maynard J, van Prooijen K. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics (TOG)* 2008; **27**(3): 1–11.
16. Pechev AN. Inverse kinematics without matrix invertion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '08)*, Pasadena, CA, USA, May 19-23 2008; 2005–2012.
17. Grochow K, Martin SL, Hertzmann A, Popović Z. Style-based inverse kinematics. In *ACM Transactions on Graphics (TOG)*. ACM, New York, NY, USA, August 2004; 522–531.
18. Wu X, Tournier M, Reveret L. Natural character posing from a large motion database. *IEEE Computer Graphics and Applications* 2011; **31**(3): 69–77.
19. Wei X, Chai J. Intuitive interactive human-character posing with millions of example poses. *IEEE Computer Graphics and Applications* July 2011; **31**(4): 78–88.
20. Ho ESL, Shum HPH, Cheung YM, Yuen PC. Topology aware data-driven inverse kinematics. In *Computer Graphics Forum*, October 2013.
21. Sumner RW, Zwicker M, Gotsman C, Popović J. Mesh-based inverse kinematics. *ACM Transactions on Graphics (TOG)* 2005; **24**(3): 488–495.
22. Der KG, Sumner RW, Popović J. Inverse kinematics for reduced deformable models. In *ACM SIGGRAPH Papers*. ACM, New York, NY, USA, 2006; 1174–1179.
23. Wang L-CT, Chen CC. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation* 1991; **7**(4): 489–499.
24. Kulpa R, Multon F. Fast inverse kinematics and kinetics solver for human-like figures. In *International Conference on Humanoid Robots, IEEE-RAS*, Tsukuba, Japan, December 2005; 38–43.
25. Müller-Cajar R, Mukundan R. Triangulation: a new algorithm for inverse kinematics. In *Proceedings of the Image and Vision Computing New Zealand 2007*, New Zealand, December 2007; 181–186.
26. Mukundan R. A robust inverse kinematics algorithm for animating a joint chain. *International Journal of Computer Applications in Technology* 2009; **34**(4): 303–308.
27. Aristidou A, Lasenby J. Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. *Technical Reports F-INFENG/TR. 632*, CUED, 2009.
28. Aristidou A, Lasenby J. Inverse kinematics solutions using conformal geometric algebra. In *Guide to*

Geometric Algebra in Practice, Dorst L, Lasenby J (eds). Springer: London, 2011; 47–62.
29. Alver J. A virtual hand for prosthetic training. *Master's Thesis*, Chalmers University of Technology, Gothenburg, Sweden, 2011.
30. Aristidou A, Lasenby J. Real-time marker prediction and CoR estimation in optical motion capture. *The Visual Computer* 2013; **29**(1): 7–26.
31. Poddighe R, Roos N. A NAO robot paying tic-tac-toe: comparing alternative methods for inverse kinematics. In *Proceedings of the 25th Belgium–Netherlands Artificial Intelligence Conference,* BNAIC'13, November 2013.
32. Liu Y-C. Gesture and end-effector trajectory planning of robot manipulator using human motion imitation. *Master's Thesis*, National Taipei University of Technology, China, 2012.
33. Munshi SM. Analysis, investigation and design of flexible universal pneumatic industrial manipulators involving cartesian and joint control in the basis of economic feasibility and appropriate technology. *PhD thesis*, University of New South Wales, Australia, 2012.
34. Lo HS, Xie SSQ. Optimization of a redundant 4R robot for a shoulder exoskeleton. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Wollongong, Australia, July 2013; 798–803.
35. Hwang S, Choi Y. Pose estimation of small-articulated animals using multiple view images. In *Proceedings of the International Conference on Computer Graphics, Visualization and Computer Vision,* WSCG '15, June 2014.
36. Ramachandran S, John NW. A fast inverse kinematics solver using intersection of circles. In *Proceedings of the Eurographics UK Theory and Practice of Computer Graphics Conference*. Eurographics, Bath, UK, September 2013.
37. Le Naour T, Courty N, Gibet S. Cinmatique guide par les distances. *Revue lectronique Francophone dInformatique Graphique* 2012; **6**(1): 15–25.
38. Huang J, Pelachaud C. An efficient energy transfer inverse kinematics solution. In *Motion in Games (MIG),* volume 7660 of *Lecture Notes in Computer Science*, Kallmann M, Bekris KE (eds). Springer, 2012; 278–289.
39. Moya S, Colloud F. A fast geometrically-driven prioritized inverse kinematics solver. In *Proceedings of the XXIV Congress of the International Society of Biomechanics (ISB 2013)*, August 2013; 1–3.
40. Bentrah A, Djeffal A, Babahenini M, Gillet C, Pudlo P, Taleb-Ahmed A. Full body adjustment using iterative inverse kinematic and body parts correlation. In *Computational Science and Its Applications—ICCSA 2014,* volume 8584 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014; 681–694.
41. Blow J. Inverse kinematics with quaternion joint limits. *Game Developer Magazine* 2012: 16–18.
42. Wilhelms J, Gelder AV. Fast and easy reach-cone joint limits. *Journal of Graphic Tools* 2001; **6**(2): 27–41.
43. Baerlocher P, Boulic B. Parametrization and range of motion of the ball-and-socket joint. In *Deformable Avatars*. Kluwer Academic Publishers, 2001; 180–190.
44. Yamane K, Nakamura Y. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics* 2003; **9**(3): 352–360.
45. Maurel W, Thalmann D. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics* 2000; **24**(2): 203–18.
46. Wang X, Verriest JP. A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation. *Journal of Visualization and Computer Animation* 1998; **9**(1): 33–47.
47. Kloǒcar N, Tomšič M, Lenarčič J. A kinematic model of the shoulder complex to evaluate the arm-reachable workspace. *Journal of Biomechanics* 2007; **40**(1): 86–91.
48. Herda L, Urtasun R, Hanson A, Fua P. Automatic determination of shoulder joint limits using experimentally determined quaternion field boundaries. *International Journal of Robotics Research* 2003; **22**(6).
49. Badler NI, Phillips CB, Webber BL. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press: New York, Oxford, 1993.
50. White III AA, Panjabi MM. *Clinical Biomechanics of the Spine,* 2nd edn. J.B. Lippincott Company: Michigan, USA, 1990.
51. Monheit G, Badler NI. A kinematic model of the human spine and torso. *IEEE Computer Graphics and Applications* 1991; **11**(2): 29–38.
52. Rijpkema H, Girard M. Computer animation of knowledge-based human grasping. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, New York, NY, USA, 1991; 339–348.
53. Murray RM, Sastry SS, Zexiang L. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc.: Boca Raton, FL, USA, 1994.
54. Kaimakis P, Lasenby J. Gradient-based hand tracking using silhouette data. In *Proceedings of the 3rd International Symposium on Visual Computing (ISVC),* volume 1, Lake Tahoe, NV/CA, USA, November 26–28, 2007; 24–35.

55. Brown J, Latombe J-C, Montgomery K. Real-time knot-tying simulation. *The Visual Computer* 2004; **20**(2): 165–179.
56. Harrison J, Rensink RA, van de Panne M. Obscuring length changes during animated motion. *ACM Transactions on Graphics* 2004; **23**(3): 569–573.
57. Aristidou A, Cameron J, Lasenby J. Predicting missing markers to drive real-time centre of rotation estimation. In *Proceedings of the V Conference on Articulated Motion and Deformable Objects,* volume 5098. LNCS, Mallorca, Spain, 2008; 238–247.
58. Cameron J, Lasenby J. A real-time sequential algorithm for human joint localization. In *ACM SIGGRAPH Posters*. ACM Press, New York, USA, 2005; 107.

SUPPORTING INFORMATION

Supporting information may be found in the online version of this article.

AUTHORS' BIOGRAPHIES

Andreas Aristidou is a post-doc researcher associated with the Department of Computer Science, University of Cyprus. He had been a Cambridge European Trust fellow, at the Department of Engineering, University of Cambridge, where he obtained his PhD. Andreas has a BSc in Informatics and Telecommunications from the National and Kapodistrian University of Athens (2005), and he is an honour graduate of Kings College London (2006), where he obtained his MSc degree in Mobile and Personal Communications. His main interests are focused on 3D human motion analysis and classification and human animation and involve optical motion capture, motion synthesis, real-time marker prediction and centre-of-rotation estimation, inverse kinematics and applications of geometric algebra in engineering.

Yiorgos Chrysanthou is an Associate Professor at the Computer Science Department of the University of Cyprus, where he is heading the Graphics and Hypermedia Lab. He was educated in the UK (BSc and PhD from Queen Mary and Westfield College) and worked for several years as a research fellow and a lecturer at University College London. He has been a Visiting Researcher at the University of California at Berkeley, USA (1992), and at Tel-Aviv University, Israel (1997). His research interests are in the general area of computer graphics and virtual/augmented reality and applications.

Joan Lasenby received her BA and MMath in Mathematics from the University of Cambridge (1983) and completed her PhD in Radio Astronomy (at the Mullard Radio Astronomy Observatory, Cambridge) in 1987. She is currently a senior lecturer in the Signal Processing and Communications Group, Cambridge University Engineering Department. Her research interests include motion capture, computer vision, geometric algebra and image processing.