Inverse Kinematics Solutions Using Conformal Geometric Algebra

3

Andreas Aristidou and Joan Lasenby

Abstract
This paper describes a novel iterative Inverse Kinematics (IK) solver, FABRIK, that is implemented using Conformal Geometric Algebra (CGA). FABRIK uses a forward and backward iterative approach, finding each joint position via locating a point on a line. We use the IK of a human hand as an example of implementation where a constrained version of FABRIK was employed for pose tracking. The hand is modelled using CGA, taking advantage of CGA's compact and geometrically intuitive framework and that basic entities in CGA, such as spheres, lines, planes and circles, are simply represented by algebraic objects. This approach can be used in a wide range of computer animation applications and is not limited to the specific problem discussed here. The proposed hand pose tracker is real-time implementable and exploits the advantages of CGA for applications in computer vision, graphics and robotics.

3.1 Introduction

This paper describes a fast iterative Inverse Kinematics (IK) solver which is implemented using Conformal Geometric Algebra (CGA). Geometric Algebra (GA) [1] provides a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. The conformal model of GA extends the usefulness of the 3D GA by expanding the class of rotors to include translations, dilations and inversions, as well as being able to express lines, planes, circles and spheres as elements of the algebra. Rotors are more numerically stable and more

A. Aristidou (✉) · J. Lasenby
Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK
e-mail: aa462@cam.ac.uk

J. Lasenby
e-mail: jl221@cam.ac.uk

efficient than rotation matrices, making GA popular for applications in computer graphics and robotics. A more detailed treatment of GA can be found in [2].

The CGA geometric representation and its algebraic richness offer great flexibility in the process of modelling virtual or mechanical objects. In this paper, a method for solving the IK problem of a 3D human hand, which uses the CGA framework, is presented. The model is highly constrained with both rotational and orientational constraints, allowing motion only within a feasible set. Using data from a markered optical motion capture system, the 3D hand pose was efficiently tracked and reconstructed. It is important to note that this is not a system designed specifically for the task of hand tracking and reconstruction, but rather to provide a framework for many IK applications in computer vision and robotics. Both the IK solver and the hand model are real-time implementable, and the system produces motion which is smooth and natural.

3.2 Background

Inverse Kinematics is defined as the problem of determining a set of appropriate joint configurations for which the end effectors move to desired positions as smoothly, rapidly and as accurately as possible. Several models have been implemented for solving the IK problem from different areas of study. A detailed review of IK solvers is given in [3]. Most of the literature which uses CGA to address the IK problem presents kinematic solutions which focus on the advantages that the CGA model offers, rather than presenting a complete IK solver. For instance, [4] gives a simple framework solution for a robot arm, underlining the generality and the efficiency of the CGA mathematical model for solving the IK problem. Corrochano and Kähler [5] used a language of points, lines and planes (which are later replaced by spheres in [6]) to solve the IK problem of a specific robot arm. Similar solutions were given by [7–10], where CGA was used to deal with forward kinematics, dynamics and projective geometry problems. In [11], a technique for the combination of very efficient algorithms, based on two different optimisation approaches using Gaigen 2 and MAPLE, is presented. CGA therefore appears to be a promising mathematical tool for computing the IK of a robot arm and solving the problem of visually guided grasping. Recently, [12] described an application of CGA to the analysis of a parallel manipulator with limited mobility. [13] gives a brief introduction to CGA and describes basic geometric entities; it also gives a synopsis of different IK framework solutions and grasping processes of a robot arm. Finally, [14] proposed an optimised algorithm to provide IK solutions using reconfigurable hardware, leading to very efficient implementations. In summary, most of these methods are applied to the simple kinematic problems of a robot arm with 5 degrees of freedom (DoF). They mainly describe how to constrain the movement of the arm to a feasible set within a framework rather than describing a solver itself. In this paper we describe a heuristic algorithm that solves the IK problem in an iterative fashion, akin to the popular CCD method [15]. FABRIK (Forward And Backward Reaching Inverse Kinematics) is a reliable iterative algorithm that uses points, lines and spheres to solve the IK problem. It divides the problem into two phases, a forward and a backward reaching approach, and it can treat most of the joint types and supports biomechanical constraints on chains with both single and multiple end effectors. It is fast, computationally efficient and provides visually smooth results.

3.3 FABRIK: An Iterative Inverse Kinematics Solver

FABRIK uses the previously calculated positions of the joints to find the updates in a forward and backward iterative manner. The proposed IK solver starts from the last joint of the chain and works forwards, adjusting each joint along the way. Thereafter, it works backwards in the same way, in order to complete a full iteration.

Therefore, assume that $\mathbf{p}_1, \ldots, \mathbf{p}_n$ are the joint positions of a manipulator. For the simple case where only a single end effector exists, take $\mathbf{p}_1$ as the root joint and $\mathbf{p}_n$ as the end effector. The target is $\mathbf{t}$, and the initial base position is $\mathbf{b}$. First calculate the distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$ for $i = 1, \ldots, n-1$. Then, to check whether the target is reachable or not, find the distance between the root and the target, *dist*, and if this distance is smaller than the total sum of all the inter-joint distances, $dist < \sum_1^{n-1} d_i$, the target is within reach; otherwise, it is unreachable. If the target is within reach, a full iteration is constituted by two stages. In the first stage, the algorithm estimates each joint position starting from the end-effector, $\mathbf{p}_n$, moving inwards to the manipulator base, $\mathbf{p}_1$. So, let the new position of the end-effector be the target position, $\mathbf{p}'_n = \mathbf{t}$. The new position of the $(n-1)$th joint, $\mathbf{p}'_{n-1}$, is assigned as the nearest point on the sphere Σ_{n-1}, with centre the joint position $\mathbf{p}'_n$ and radii the distance d_{n-1} from the joint position $\mathbf{p}_{n-1}$. Similarly, the new position of the $(n-2)$th joint, $\mathbf{p}'_{n-2}$, is selected as the nearest point on sphere Σ_{n-2}, with centre the joint position $\mathbf{p}'_{n-1}$ and radii the distance d_{n-2} from the joint $\mathbf{p}_{n-2}$. The algorithm continues until all new joint positions are calculated, including the root, $\mathbf{p}'_1$. The nearest point on a sphere from a point in space is clearly found by simply taking a point along the line joining the centre of the sphere to the point, which has distance from the centre equal to the sphere radius. An entirely CGA solution is also given in Sect. 3.4.2.1.

A full iteration is completed when the same procedure is repeated but this time starting from the root joint and moving outwards to the end effector. Thus, let the new position for the first joint, $\mathbf{p}''_1$, be its initial position $\mathbf{b}$. Then, the new joint position $\mathbf{p}''_2$ is assigned as the nearest point on the sphere Σ_1, with centre the $\mathbf{p}''_1$ and radii the distance d_1 from the joint $\mathbf{p}'_2$. This procedure is repeated for all the remaining joints, including the end effector. FABRIK is illustrated in pseudo-code in Algorithm 1, and a graphical representation of a full iteration of the algorithm is demonstrated in Fig. 3.1.

The forward and backward procedure is then repeated for as many iterations as needed, until the end effector is identical or close enough (to be defined) to the desired target. FABRIK can easily handle end effector orientations and supports, to the best of our knowledge, all chain classes. It can also cope with cases where the model has multiple chains and end effectors and is applicable to problems with

Algorithm 1: A full iteration of the FABRIK algorithm using CGA.

```
Input: The joint positions $\mathbf{p}_i$ for $i = 1, \dots, n$, the target position $\mathbf{t}$ and the
       distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$ for $i = 1, \dots, n-1$
Output: The new joint positions $\mathbf{p}_i$ for $i = 1, \dots, n$.
% The distance between root and target
$dist = |\mathbf{p}_1 - \mathbf{t}|$
% Check whether the target is within reach
if $dist > d_1 + d_2 + \cdots + d_{n-1}$ then
    % The target is unreachable
    for $i = 1, \dots, n-1$ do
        % Find the nearest point on sphere, with centre the joint position $\mathbf{p}_i$ and radius
        the distance $d_i$, from a point is space, $\mathbf{t}$
        $\mathbf{p}_{i+1} = \mathbf{NearestPointSphere}(\mathbf{p}_i, d_i, \mathbf{t})$;
    end
else
    % The target is reachable; thus, set as $\mathbf{b}$ the initial position of the joint $\mathbf{p}_1$
    $\mathbf{b} = \mathbf{p}_1$
    % Check whether the distance between the end effector $\mathbf{p}_n$ and the target $\mathbf{t}$ is greater
    than a tolerance.
    $dif_A = |\mathbf{p}_n - \mathbf{t}|$
    while $dif_A > tol$ do
        % STAGE 1: FORWARD REACHING
        % Set the end effector $\mathbf{p}_n$ as target $\mathbf{t}$
        $\mathbf{p}_n = \mathbf{t}$
        for $i = n-1, \dots, 1$ do
            % Find the nearest point on sphere, with centre the joint position $\mathbf{p}_{i+1}$ and
            radius the distance $d_i$, from a point is space, $\mathbf{p}_i$
            $\mathbf{p}_i = \mathbf{NearestPointSphere}(\mathbf{p}_{i+1}, d_i, \mathbf{p}_i)$;
        end
        % STAGE 2: BACKWARD REACHING
        % Set the root $\mathbf{p}_1$ its initial position.
        $\mathbf{p}_1 = \mathbf{b}$
        for $i = 1, \dots, n-1$ do
            % Find the nearest point on sphere, with centre the joint position $\mathbf{p}_i$ and
            radius the distance $d_i$, from a point is space, $\mathbf{p}_{i+1}$
            $\mathbf{p}_i = \mathbf{NearestPointSphere}(\mathbf{p}_i, d_i, \mathbf{p}_{i+1})$;
        end
        $dif_A = |\mathbf{p}_n - \mathbf{t}|$
    end
end

% Where the function $\mathbf{NearestPointSphere}(X, Y, Z)$, finds the nearest point on a sphere
from a point in space. X is the sphere's centre, Y is the sphere's radii and Z is the point
in space.
```

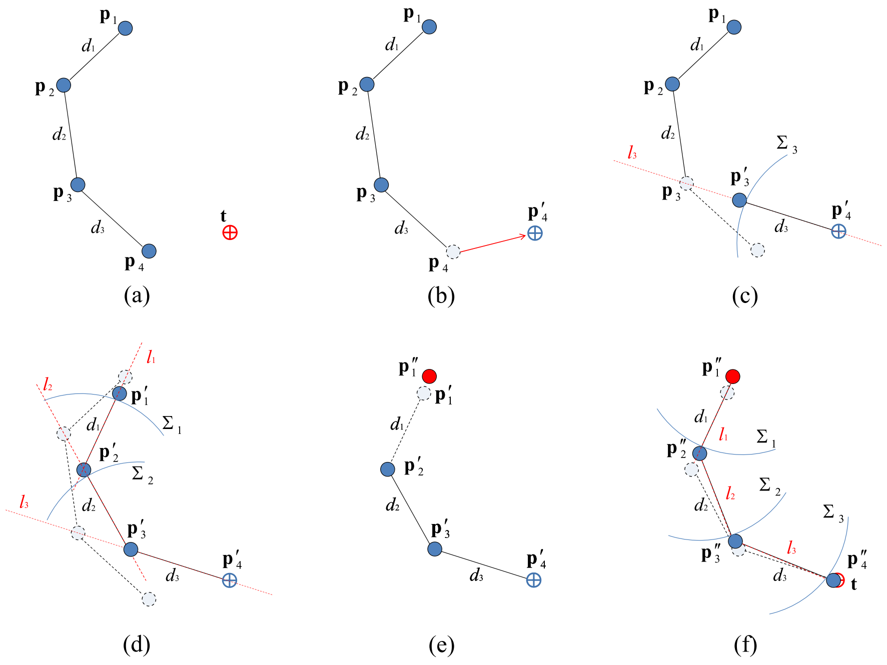

Fig. 3.1 A full iteration of FABRIK for the case of a single target and 4 joints using CGA. (**a**) The initial position of the manipulator and the target, (**b**) move the end effector $\mathbf{p}_4$ to the target, (**c**) find the joint $\mathbf{p}'_3$ which is the intersection of the sphere Σ_3 and the line l_3 which passes through the points $\mathbf{p}'_4$ and $\mathbf{p}_3$, (**d**) continue the algorithm for the rest of the joints, (**e**) the second stage of the algorithm: move the root joint $\mathbf{p}'_1$ to its initial position, (**f**) repeat the same procedure but this time start from the base and move outwards to the end effector. The algorithm is repeated until the position of the end effector reaches the target or gets sufficiently close

closed loops. A reliable method for incorporating constraints is presented in [16]; the main idea is the repositioning and reorientation of the target to be within the allowed range of motion. In this paper we give details of how these constraints can be applied to a hand model, restricting the hand motion to a feasible set. It is worth noting that FABRIK, as described in Fig. 3.1, can be implemented simply by taking distances along lines rather than intersecting with spheres [16]. However, when we wish to incorporate constraints, we often need the sphere-line information; so we choose to work entirely in this unified framework. Also, the CGA framework offers several algorithm optimisations such as for cases where the 'end effector' is not positioned at the end of the chain (i.e. it is a leaf). For instance, assume that the joint positions $\mathbf{p}_i$ and $\mathbf{p}_{i-2}$ are known and that we want to estimate the joint position $\mathbf{p}_{i-1}$. This can be done by finding the intersection of the spheres Σ_1 and Σ_2 with centres the known joint positions $\mathbf{p}_i$ and $\mathbf{p}_{i-2}$ and radii the distances $d_i = |\mathbf{p}_i - \mathbf{p}_{i-1}|$ and $d_{i-2} = |\mathbf{p}_{i-2} - \mathbf{p}_{i-1}|$, respectively. If the intersection is a circle, then the estimated joint position can be assigned as the nearest point on that circle from its previous position (as described in Sect. 3.4.2.2). If the intersection is a single

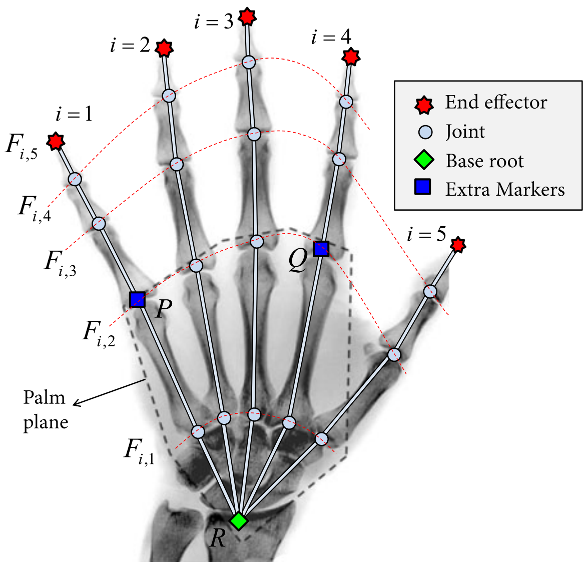

Fig. 3.2 The hand's model geometry used in our implementation

point, the estimated joint position is assumed to be that point; otherwise, if the two spheres do not intersect, the estimated joint position is equal to $\mathbf{p}_{i-1} = \frac{\mathbf{p}_i + \mathbf{p}_{i-2}}{2}$. Another simple optimisation is the direct construction of a line pointing towards the target, when the latter is unreachable. In that case, each joint $\mathbf{p}_i$ is assigned to be the nearest point on the sphere, with centre the previous joint $\mathbf{p}_{i-1}$ and radii the distance d_{i-1} from the target.

3.4 Using FABRIK for Hand Pose Tracking

In this section, an example of FABRIK implementation and how it performs on hand pose tracking is presented. The hand rotational and orientational limitations have been incorporated using CGA. The proposed approach is an example of object modelling for kinematic solutions, and we note that it can be adjusted to solve a variety of different modelling problems.

3.4.1 The Hand Geometry

It is assumed that the hand geometry, meaning the initial joint configuration of the hand, is known a priori. An example of a hand model is graphically represented in Fig. 3.2. The proposed hand model consists of 25 joints and has in total 25 DoFs. The end effector positions are captured using an optical motion capture rig, such as the Phasespace Impulse System [17]. Since our hand model does not have a mesh which defines its external shape, constraints such as self collisions are not considered here. The markers are identified (e.g. in the Phasespace system, each

LED marker is pulsed at a different frequency) so that it is known a priori on which finger each marker is placed. It is also important to know the orientation of the hand in order to efficiently incorporate constraints. This can be achieved by attaching two extra markers at specific positions, p and q, on the back of the hand (reverse palm). Assuming that the palm is always flat, we can find the plane describing the orientation of the hand using p, q and the position of the base root, r, which also lies on the palm plane. For simplicity, markers p and q can be placed at the joint positions $F_{1,2}$ and $F_{4,2}$ respectively, as shown in Fig. 3.2.

Before employing the IK solver, it is crucial to find the fingers' orientations, the chain roots and the end effectors for each chain; the target positions are assumed to be known since they are tracked by the motion capture system. The procedure is simple. Firstly, we estimate the hand orientation; thereafter, we calculate the palm joints and the finger orientations at each time step. When each finger orientation is known, the finger joints at the previous time step are translated and rotated in such a way that all joints belong to the current finger plane. Finally, a constrained version of FABRIK, with rotational limitations, is incorporated to fit the joints of each finger. This procedure is given in detail in the following paragraphs.

The first step is to find the hand orientation; hence, by accepting that the hand plane Φ_x is similar to the palm plane and that the markers p, q and r are lying on that plane, the hand orientation, meaning the plane Φ_x, can be estimated. Therefore,

$$\begin{aligned} P &= \frac{1}{2}\left(p^2 n_\infty + 2p - \bar{n}\right) \\ Q &= \frac{1}{2}\left(q^2 n_\infty + 2q - \bar{n}\right) \\ R &= \frac{1}{2}\left(r^2 n_\infty + 2r - \bar{n}\right) \end{aligned} \tag{3.1}$$

where P, Q and R are the 5D null vectors representing points p, q and r, respectively, and n_∞ and $\bar{n}$ are the null vectors in CGA.[1] The plane Φ_x is given by

$$\Phi_x = P \wedge Q \wedge R \wedge n_\infty = \left\langle \left\langle \langle PQ \rangle_2 R \right\rangle_3 n_\infty \right\rangle_4 \tag{3.2}$$

Note that the form given on the right-hand side of (3.2), and other relevant equations, is useful for implementation purposes and so is included here.

Calculating the Palm Joints The next step is to incorporate constraints to obtain other palm joints. Thus, by assuming that the inter-joint distances (for the joints $F_{i,1}$ where $i = 1, \ldots, 5$ and $F_{j,2}$ where $j = 1, \ldots, 4$) are fixed over time and that all these joints lie on the palm plane, we can easily locate them using basic geometric entities such as planes, circles and spheres. An example of palm constraints is given

[1] *Editorial note*: Note here that in this chapter CGA equations are given in terms of n_∞ and $\bar{n}$, where $\bar{n} = -2n_o$.

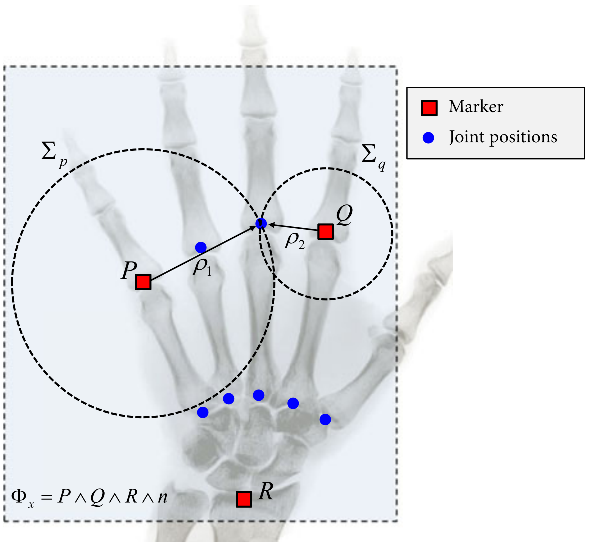

Fig. 3.3 The palm plane constraints: the hand plane can be calculated using the marker positions P, Q and R, accepting that the markers lie on that plane and that the hand and palm planes are similar. The rest of the palm joints can be estimated, assuming that the inter-joint distances remain constant over time, by intersecting the spheres Σ_p and Σ_q with centres at the marker positions P and Q and radii of the distance between their centre and the joint position we are looking for

in Fig. 3.3. For instance, the joint position we are working on can be estimated by intersecting the spheres with centres being the marker positions p and q and radii being the distance between the marker and that joint position (taken from the model). Therefore, find the sphere with its centre at the marker position P and radius equal to the distance between the marker P and the joint we are working on

$$\Sigma_p = \left(P - \frac{1}{2}\rho_1^2 n_\infty\right) I \tag{3.3}$$

where ρ is the sphere radius. Similarly, find the sphere with centre the marker position Q and radius equal to the distance between the marker Q and the joint we are working on

$$\Sigma_q = \left(Q - \frac{1}{2}\rho_2^2 n_\infty\right) I \tag{3.4}$$

The intersection of the two spheres gives a circle or a single point or no intersection. The meet between the two spheres is given by

$$C = \Sigma_p \vee \Sigma_q = \left[\langle \Sigma_p \Sigma_q \rangle_2\right]^* \tag{3.5}$$

- If $C^2 > 0$, then C is a circle. In that case, the possible solutions are given by intersecting the circle C and the palm plane Φ_x:

$$B = C \vee \Phi_x = \left[\langle C \Phi_x \rangle_3\right]^* \tag{3.6}$$

- If $B^2 > 0$, the meet between C and Φ_x gives two points which can be extracted via projectors, as described in [18]. The new joint position is assigned as the point that is closer to the previous joint position (at time $k-1$).
- If $B^2 = 0$, the intersection is a single point $X = Bn_\infty B$.
- If $B^2 < 0$, the intersection does not exist. For that instance, the new joint position is then taken as the nearest point on circle, C, from the previous joint position (at time $k-1$, see Sect. 3.4.2.2).

• If $C^2 = 0$, the intersection is a single point $X = Cn_\infty C$.
• If $C^2 < 0$, the two spheres do not intersect. In that case, the final joint position is given by averaging the distance between the two markers, $x = (p+q)/2$.

Calculating the Finger Joints In order to estimate the finger joints, we need to find the finger planes Φ_i for $i = 1, \dots, 4$. Each Φ_i can be calculated using the known joint positions $F_{i,2}$, the marker positions $F_{i,5}$ and by assuming that they are perpendicular to the palm plane Φ_x (note that this does not hold for the thumb plane Φ_5). Since both points from each finger are known (the motion capture system tracks the end effector positions $F_{i,5}$, and the finger roots $F_{i,2}$ lie on the palm plane with constant distance from the attached markers p and q, as explained in previous paragraphs), each finger plane can be estimated at the current time frame. The vector that is perpendicular to the hand plane Φ_x is given by

$$\hat{n} = \Phi_x^* - \frac{1}{2}\big(\Phi_x^* \cdot \bar{n}\big)n_\infty \tag{3.7}$$

as explained in [18]. The finger planes can then be calculated as

$$\Phi_i = F_{i,2} \wedge F_{i,5} \wedge \hat{n} \wedge n_\infty = \big\langle\big\langle\langle F_{i,2}F_{i,5}\rangle_2 \hat{n}\big\rangle_3 n_\infty\big\rangle_4 \quad \text{for } i = 1, \dots, 4 \tag{3.8}$$

The thumb orientation Φ_5 can be estimated using the marker position $F_{5,4}$ and the joint positions $F_{1,2}$ and $F_{5,2}$ that lie on the palm, assuming that when the thumb bends to the ventral side of the palm, it always points at the joint $F_{1,2}$ (approximately true in practice).

The next step is to estimate the rotation between the previous and the current frame of each finger plane. This can be done using rotors; the rotor R which expresses the rotation between the plane in the previous frame and the plane in the current frame, for each finger, can be found using the closed-form expression given in [19].[2] Then each finger joint at time $k-1$ is translated and rotated in such a way that all joints of a given finger lie on the plane of the current frame k, as demonstrated in Figs. 3.4 and 3.5. Hence,

$$\hat{F}_{i,j}^{k} = RF_{i,j}^{k-1}\tilde{R} \tag{3.9}$$

[2] *Editorial note*: This is essentially (4.2) for $n = 3$.

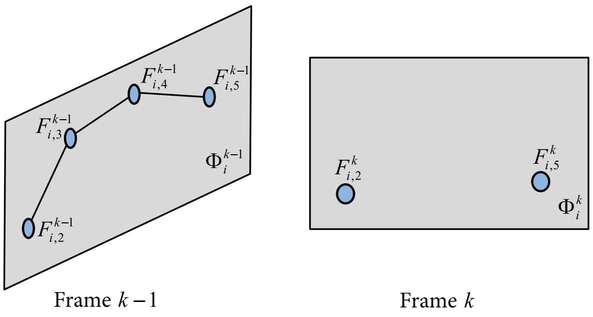

Fig. 3.4 The joint positions at times $k-1$ and k. Each finger joint at time $k-1$ needs to be rotated in such a way that all joints of that finger lie on the plane of the current frame k

where $i = 1, \ldots, 4$ and $j = 3, 4, 5$ (except for the thumb where $i = 5$ and $j = 2, 3, 4$).

All joints now lie on plane Φ_i^k. Lastly, FABRIK is applied to each finger chain, assuming that the root of the chain is $F_{i,2}^k$, the end effector is the rotated point $\hat{F}_{i,5}^k$, and the target is the current marker position $F_{i,5}^k$, as shown in Fig. 3.5. The inter-joint distances are constant over time; thus, for computational efficiency, they can be calculated and stored at the first frame. It is important here to note that the marker occlusion problem is considered solved using constrained prediction algorithms, such as [20].

The resulting posture can be further improved in accuracy and naturalness by incorporating properties of the fingers, muscles, skin and individual joints via constraints [21].

3.4.2 Trigonometric Solutions

This section presents trigonometric solutions, using CGA, to problems which appear during the implementation of the proposed methodology.

3.4.2.1 Nearest Point on a Sphere from a Point in Space

This section shows how to calculate the nearest point on a sphere from a point in space using CGA. Assume that a sphere has centre $\underline{c}$ and radius ρ. The sphere Σ_1 can be expressed as a blade in CGA as follows:

$$\Sigma_1 = \left(c - \frac{1}{2}\rho^2 n_\infty\right) I \tag{3.10}$$

where $c = \frac{1}{2}\underline{c}^2 n_\infty + \underline{c} - \frac{\bar{n}}{2}$.

Assume a point in space q. In order to find the nearest point on the sphere from that point, we need to find the intersection of the line L_1 that passes through the point q and the sphere centre $\underline{c}$. Thus,

$$L_1 = Q \wedge c \wedge n_\infty = \big\langle \langle Qc \rangle_2 n_\infty \big\rangle_3 \tag{3.11}$$

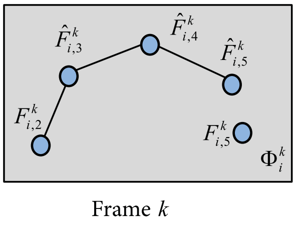

Fig. 3.5 The current joint positions, after rotating them in order to lie on the current finger plane Φ^k_i. The problem of orientation is therefore solved, and FABRIK can then be utilised assuming that the root of the chain is $F^k_{i,2}$, the end effector is the point $\hat{F}^k_{i,5}$, and the target is the current marker position $F^k_{i,5}$

where $Q = H(q)$ is the Hestenes mapping of q. The intersection between the line L_1 and the sphere Σ_1 always returns two possible solutions, which are given by the bivector $X_1 \wedge X_2$.

$$X_1 \wedge X_2 = L_1 \vee \Sigma_1 \tag{3.12}$$

Finally, the vectors X_1 and X_2 can be extracted from $X_1 \wedge X_2$ using projectors. Then, the nearest point on the sphere is assigned as the point that returns the minimum distance from the point in space.

We note here that although the nearest point on a sphere from a point in space can be found very easily using distance along lines, because we are working entirely in the CGA framework (in order to easily incorporate constraints), it is generally more computationally efficient to do all calculations in CGA.

$$X = \arg\big(\max(X_1 \cdot X, X_2 \cdot X)\big) \tag{3.13}$$

3.4.2.2 Nearest Point on a Circle from a Point in Space

This section describes how to find the nearest point on a circle from a point in space. In particular, the minimum distance on a circle from a point in space is related to the projection of that point onto the plane Φ of the circle. This can be achieved by reflecting the point in the plane and finding the mid-point of the reflected and the original point. Hence, let the circle $C = H(b) \wedge H(c) \wedge H(d)$, where b, c and d are points that lie on the circle. The centre c of the circle C can be calculated as

$$c = Cn_\infty C \tag{3.14}$$

and the plane Φ of the circle can be formulated as

$$\Phi = C \wedge n_\infty = \langle Cn_\infty \rangle_4 \tag{3.15}$$

Having the plane Φ and the point $X = H(x)$ in space, the nearest point on the circle can be found by reflecting that point in the plane Φ.

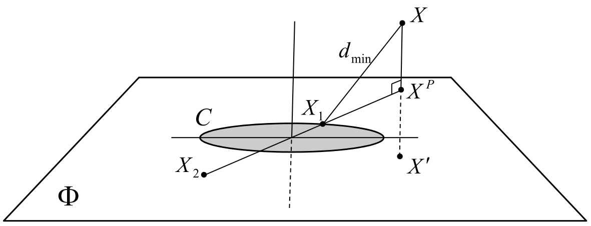

Fig. 3.6 The nearest point on circle to point in space. The point X is projected on the circle's plane Φ. A line is then formed through the midpoint of X and its projected counterpart and the centre of the circle. The intersection between the line and the circle returns two possible solutions; the one that is shorter to the point X is chosen

$$X' = \Phi X \Phi \tag{3.16}$$

The mid-point X^P is then calculated as

$$X'^P = X^P + \alpha n_\infty = H\left(\frac{1}{2}\big(H^{-1}\big(X'\big) + x\big)\right) \tag{3.17}$$

Then, a line, L, is formed through this midpoint and the centre of the circle,

$$L = X^P \wedge c \wedge n_\infty \tag{3.18}$$

The intersection between line L and circle C will return a bivector, $A \wedge B$, which represents the shortest and longest distances on the circle from the point in space. The vectors X_1 and X_2 can be extracted from $X_1 \wedge X_2$ using projectors. The nearest point is then selected using a simple distance comparison method. This method is also illustrated in Fig. 3.6.

$$\begin{aligned}(X_1, X_2) &= L \vee C \\ X &= \arg\big(\max(X_1 \cdot X, X_2 \cdot X)\big)\end{aligned} \tag{3.19}$$

3.5 Experimental Results

Experiments were carried out using a 10 camera Phasespace motion capture system, capturing data at 100 Hz [17]. The implemented methodology was able to process up to 70 frames per second, using MATLAB [22]. Our dataset comprises markered motion capture data; data captured using colour video cameras are also used to compare the reconstruction quality between the estimated and the true hand postures. The reconstructed hand postures were visualised using a mesh deformation algorithm.

The proposed method is real-time implementable, requiring only 1.43 ms per frame for tracking and fitting 25 joints. FABRIK is able to fit the joints and reconstruct the hand accurately; the rotational and orientational constraints ensure that

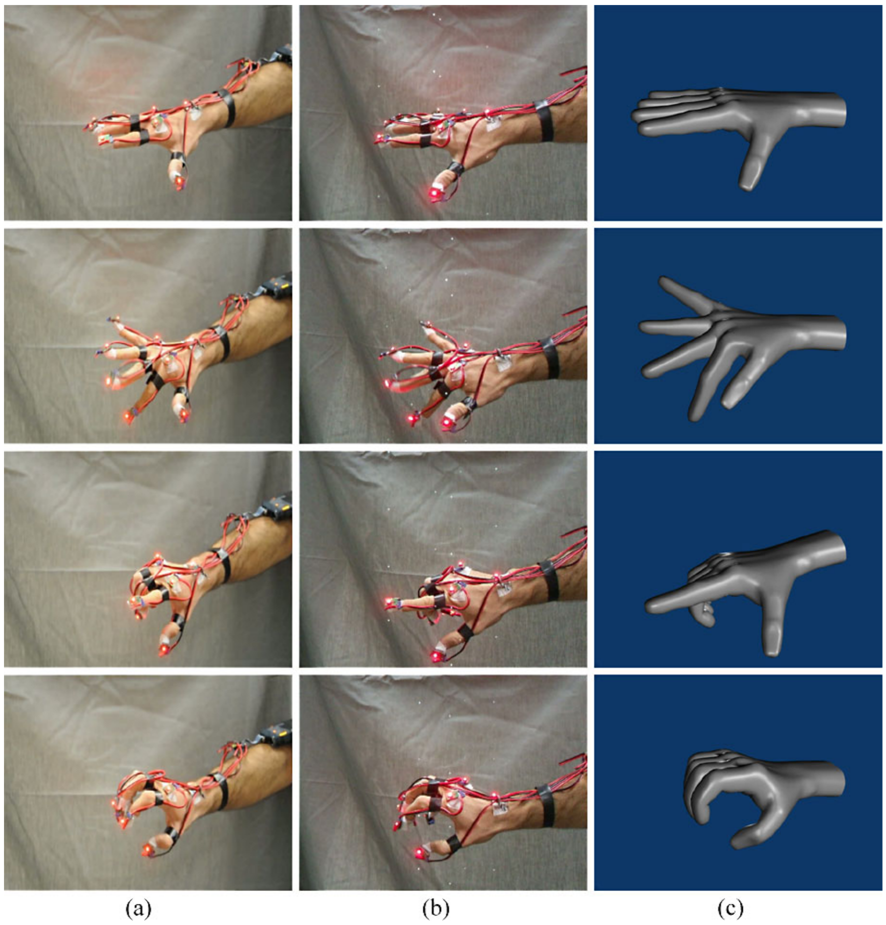

Fig. 3.7 An example of hand reconstruction using our methodology at a frame rate of 100 Hz. (**a**) View of the hand from RGB camera 1, (**b**) a different view of the hand from RGB camera 2 and (**c**) the final visualised posture. The resulting poses are visually natural and biomechanically correct

each finger movement remains normal without showing asymmetries, or irregular bends and rotations.

The implemented system can smoothly track the hand movements. The reconstruction quality can be checked visually by comparing the generated 3D hand animations with the data captured using a colour video camera, as seen in Fig. 3.7. It is difficult to illustrate the reconstruction quality in still images, but the resulting motion does not suffer from oscillation or discontinuities, and each finger smoothly moves to the target.

Despite the accuracy in performance, the resulting postures of our approach are not unique; several possible poses could result from the 3D articulated hand tracking. However, the advantages of this method are its efficiency and ability to return natural and feasible motion, which meets the user constraints, with low computational cost. It is also important to note here that FABRIK results in poses which are closely related to previous states. Therefore, the final joint configuration might be different when the IK problem is solved with the end effectors in different initial positions but with similar final states. Nevertheless, these differences are minimal causing only a small decrease in performance.

3.6 Conclusions and Future Work

In this paper, we presented an iterative Inverse Kinematics solver that was implemented using CGA. Rotational and orientational constraints were then incorporated for hand modelling; using a minimum number of available markers, we were able to track the 25 DoF hand relying on optical motion capture data. One labelled optical marker was attached to the end of each finger, treated as an end effector, and 3 more markers were placed at strategic positions on the hand reverse-palm to help us identify the root and orientation of the hand. The proposed methodology produced smooth and natural hand postures over time; the required processing time remained low enabling an effective real-time hand motion tracking and reconstruction system. The results were precise, producing visually natural, smooth and biomechanically correct movements, without oscillations or discontinuities.

This application exploits the advantages of CGA for incorporation of constraints in IK problems and proves that it is a useful mathematical tool which can be successfully used for applications in computer vision, graphics and robotics. In general, CGA gives us the ability to describe algorithms in a geometrically intuitive and compact manner. More particularly, it simplifies the mathematical model of the IK solver, since basic entities, such as spheres, lines, planes and circles, are simply represented by algebraic objects. In addition, the structure and elegance of CGA leads to low computational cost and real-time performance.

In future work, a more sophisticated model will be implemented which takes into consideration, in addition to the joint rotational and orientational restrictions, physiological constraints such the flexion, inertia, abduction, the finger's intradigital and transdigital correlation, the rigidity and the friction of the hand, as described in [21].

3.7 Exercises

3.1 What is the complexity of a simple unconstrained version of FABRIK for a six-joint kinematic chain?

3.2 Similarly to finding the nearest point on a circle from a point in space, find trigonometric solutions for: (a) the nearest point on a line from a sphere, (b) the nearest point on a line from a circle.

3.3 Describe a model with joint constraints for a human arm using the FABRIK Inverse Kinematics algorithm and CGA as the mathematical framework (hint: assume that the shoulder and the joint connecting the palm with the arm are ball and socket joints with rotational and orientation limits, and that elbow is a hinge joint. For simplicity, the hand should be considered as a solid limb segment).

References

1. Hestens, D., Sobczyk, G.: Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics. Reidel, Dordrecht (1984)
2. Doran, C., Lasenby, A.: Geometric Algebra for Physicists. Cambridge University Press, Cambridge (2003)
3. Aristidou, A., Lasenby, J.: Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. Cambridge University Department of Engineering Technical Report, CUED/F-INFENG/TR-632 (2009)
4. Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry. Morgan Kaufmann, San Mateo (2009)
5. Bayro-Corrochano, E., Kähler, D.: Motor algebra approach for computing the kinematics of robot manipulators. J. Robot. Syst. **17**(9), 495–516 (2000)
6. Bayro-Corrochano, E.: Robot perception and action using conformal geometric algebra. In: Handbook of Geometric Computing, pp. 405–458, Chap. 13. Springer, Berlin (2005)
7. Zamora, J., Bayro-Corrochano, E.: Inverse kinematics, fixation and grasping using conformal geometric algebra. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS '04), vol. 4, pp. 3841–3846 (2004). doi:10.1109/IROS.2004.1390013
8. Hildenbrand, D.: Tutorial: Geometric computing in computer graphics using conformal geometric algebra. Comput. Graph. **29**(5), 795–803 (2005)
9. Zamora, J., Bayro-Corrochano, E.: Kinematics and grasping using conformal geometric algebra. In: Lenarčič, J., Roth, B. (eds.) Advances in Robot Kinematics, pp. 473–480. Springer, Berlin (2006)
10. Hildenbrand, D., Zamora, J., Bayro-Corrochano, E.: Inverse kinematics computation in computer graphics and robotics using conformal geometric algebra. Adv. Appl. Clifford Algebras **18**, 699–713 (2008)
11. Hildenbrand, D., Fontijne, D., Wang, Y., Alexa, M., Dorst, L.: Competitive runtime performance for inverse kinematics algorithms using conformal geometric algebra. In: Proceedings of Eurographics Conference, 2006
12. Tanev, T.K.: Geometric algebra approach to singularity of parallel manipulators with limited mobility. In Lenarcic, J., Wenger, P. (eds.) Advances in Robot Kinematics: Analysis and Design, pp. 39–48. Springer, Dordrecht (2008)
13. Bayro-Corrochano, E., Zamora, J.: Differential and inverse kinematics of robot devices using conformal geometric algebra. Robotica **25**(1), 43–61 (2007)
14. Hildenbrand, D., Lange, H., Stock, F., Koch, A.: Efficient inverse kinematics algorithm based on conformal geometric algebra (using reconfigurable hardware). In: Proceedings of the 3rd International Conference on Computer Graphics Theory and Applications, Madeira, Portugal, 2008
15. Wang, L.-C.T., Chen, C.C.: A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. IEEE Trans. Robot. Autom. **7**(4), 489–499 (1991)
16. Aristidou, A., Lasenby, J.: FABRIK: a fast, iterative solver for the inverse kinematics problem. Graph. Models **73**(5), 243–260 (2011)
17. PhaseSpace Inc: Optical motion capture systems. http://www.phasespace.com
18. Lasenby, A.N., Lasenby, J., Wareham, R.: A covariant approach to geometry using geometric algebra. Cambridge University Department of Engineering Technical Report, CUED/F-INFENG/TR-483 (2004)

19. Lasenby, J., Fitzgerald, W.J., Lasenby, A.N., Doran, C.J.L.: New geometric methods for computer vision: an application to structure and motion estimation. Int. J. Comput. Vis. **26**(3), 191–213 (1998)
20. Aristidou, A.: Tracking and modelling motion for biomechanical analysis. PhD Thesis, University of Cambridge, Cambridge, UK (October 2010)
21. Kaimakis, P., Lasenby, J.: Physiological modelling for improved reliability in silhouette-driven gradient-based hand tracking. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, Miami, USA, 25 June 2009, pp. 19–26
22. The Mathworks—MATLAB and Simulink for technical computing. http://www.mathworks.com