Received March 21, 2021, accepted March 29, 2021, date of publication April 2, 2021, date of current version April 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3070693

FABRIK-R: An Extension Developed Based on FABRIK for Robotics Manipulators

MATHEUS C. SANTOS[1], LUCAS MOLINA[1], ELYSON A. N. CARVALHO[1], (Member, IEEE), EDUARDO O. FREIRE[1], JOSÉ G. N. CARVALHO[1], AND PHILLIPE C. SANTOS[2]

[1]Department of Electrical Engineering, Federal University of Sergipe, São Cristóvão 49100-000, Brazil
[2]Department of Electrical Engineering, Federal University of Campina Grande, Campina Grande 58429-900, Brazil

Corresponding author: Matheus C. Santos (matheus.santos@ul.ie)

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

ABSTRACT This paper develops on the process of inverse kinematics (IK) for manipulator robots based on a recent technique from the computer graphics area that has been highlighted by its simplicity and low computational cost. This IK solver is known as Forward And Backward Reaching Inverse Kinematics (FABRIK) suffers from singularities when applied in kinematic chains composed of one degree of freedom (1-DOF) joints. Some extensions of this method have been proposed to incorporate the constraints in some of the most common joints, however, their application over kinematic chains with only 1-DOF joints is still not possible. Since several manipulators have kinematic chains composed of 1-DOF joints, this work presents a new method, named FABRIK-R, to extend the original method for applications in the robotics manipulators field.

INDEX TERMS FABRIK, inverse kinematics, manipulators, robots.

I. INTRODUCTION

Since the twentieth century, robots are no longer just the idea of science fiction movie writers. They have become the focus of research studies, part of industrial processes, and even used in human surgery [1]. The evolution of industrial automation has led to advances in manipulator robotics research due to the ease of robots to fit into the production process, performing tasks efficiently and accurately to optimize work and reduce costs [2].

A manipulator robot can be as precise and accurate as human operators, associating speed and smoothness to its movements [3]. Due to these characteristics, the realistic and plausible movement of bodies has long been a problem for scholars in many fields, including robotics and computer graphics. This problem has been solved by inverse kinematic (IK) methods aimed at animating or controlling different virtual creatures [4].

In the area of computer graphics, a recent technique has been highlighted due to its efficiency to solve the inverse kinematics problem, producing smooth movements without discontinuities with a low computational cost. One of its best advantages is the simplicity due to the fact the algorithm treats each joint independently, eliminating the need to build the entire kinematic chain of the body [4]. This method is known as Forward And Backward Reaching Inverse Kinematics (FABRIK) and it has been the target of several researches in the last years. FABRIK has gained a high relevance in character animation and it has been used in modern graphics engines such as Unity3D, Unreal and Maya [5].

Despite all these advantages, this method has limitations when applied to the field of robotics. Most manipulator robots have a kinematic chain with joints of only one degree of freedom (1-DOF). This class of manipulators can be easily found in industrial applications, such as cylindrical, SCARA and KUKA robots [6]–[9], in assistive robotic with the JACO robot [10] and medical surgeries, represented by Da Vinci and Navi robots [11], [12]. The movement restriction of these robots implies a move dependency between two sequential joints. Nevertheless, FABRIK treats each joint independently, which makes its solution unfeasible.

Although the 1-DOF joint problem has been investigated by other extensions of this method [4], their application over kinematic chains with only 1-DOF joints is still not possible and this kind of chain is the problem addressed in this paper.

Therefore, we propose an extension for FABRIK to solve the problem of inverse kinematics for kinematic chains with

The associate editor coordinating the review of this manuscript and approving it for publication was Yangmin Li.

only 1-DOF joints. This approach has as main contributions the following points:

- Development of an extension to the FABRIK algorithm. This paper extends the FABRIK for pivot and hinge joints, by taking into consideration the restrictions on the previous (parent) or next (child) joint. In each step, we project the new joint onto a plan that contains the previous and actual joint and respects their respective movement constraints.
- The mathematical formalism for applying restrictions. In order to be able to apply the restrictions of each joint, a way of calculating the plan that respects the restrictions of the previous joint and allows the current joint to move in the direction of the next joint is presented and formalized.
- Application for robotics of manipulators. With the possibility of applying the technique in 1-DOF kinematic chains, the technique becomes applicable in robotics of manipulators, considering that many of the models of manipulator robots have this characteristic.

The paper is organized as follows: Section II presents the state of the art of inverse kinematics; Section III is about the algorithm FABRIK; Section IV is about the limitations of FABRIK extension to solve the movements constraints problem; Section V is dedicated to the method proposed in this paper, which incorporates the movement restrictions by adding the constrictions of the previous (parent) or next (child) joint; The analysis of the obtained results is presented in Section VI; At last, conclusions and propositions for the future are presented in Section VII, followed by the bibliographic references.

II. WORK RELATED

The inverse kinematics problem puzzled researchers for many years in the field of robotics and computer graphics. Over the past decades, various approaches have been implemented to find a robot configuration for a given task. Because of this variety, inverse kinematics algorithms can be divided into four main categories: analytical, numerical, data-based, and hybrid [13].

Craig, in [2], presents a review with various analytical methods developed mainly in the field of robotics as they generally do not suffer from uniqueness problems, offer a global solution, and are reliable. Faria et. al. propose a method to uniquely solve IK of 7-DOF manipulators while avoiding joint limits and singularities [14]. However, the nonlinear nature of kinematic equations and the low scalability make the analytical methods less suitable for redundant systems, in which they generally fall into local minimums and cannot handle prioritized constraints.

Data-based methods learn a space of natural deformation from examples. Using the learned space, they generate new shapes that respect the deformations shown by the examples but still satisfy the constraints imposed by the user. Zhang proposes an approach that mimics human experts' behaviors in solving closed-form inverse kinematics using Behavior Tree [15]. In [16] it is presented a data method learning by demonstration to model the surgical operation skills in the Cartesian space. After that, it proposed an improved recurrent neural network (RNN) to perform the trajectory control of redundant robot manipulators with constraints.

This category also has been used in applications for which the joint torque sensors are often unavailable, such as the biomedical robots, as shown in [17]. The disadvantages of Data-based methods are the requirement of an offline training procedure and the results high dependence on the training data and limitation to the models and movements in which the system was trained.

Attempting to reduce the complexity of the optimization problem, hybrid methods split the problem into numerical and analytical components. Some of these works solve the inverse kinematics problem from a statistical point of view, such as in [18]. However, these statistical methods have a high computational cost as a disadvantage.

Among the numerical approximation methods, the best known are the Jacobians methods that linearly model the end-effector movement in relation to changes that occur in the joint angles. Within this group, several approaches have been developed to calculate or approximate the inverse of the jacobian, such as the transposed Jacobian, pseudoinverse-based, damped least squares (DLS), singular Value Decomposition (SVD) [13], [19]. The solutions presented based on the Jacobian matrix produce smooth positions, however, most of these approaches suffer from high computational cost, complex matrix calculations, and singularity problems [20].

Numerical methods also have several approaches that are characterized by their probabilistic search. This search is made in the configuration space and each sample has an associated value calculated by a fitness function. The main methods of this category are Ant Bee Colony, Firefly Algorithm, and Particle Swarm Optimization [21]. These approaches are recommended for solving complex problems and have the advantage of not falling into singularities, as they do not invert the Jacobian matrix [22]. However, many of these methods can fall into the local minimum problem and present inconsistent results.

Heuristic-based methods are the simplest and fastest numerical approaches, among which can be highlighted the Cyclic Coordinate Descent (CCD) and FABRIK [4]. [23] introduces CCD, a widely used approach in graphic animation and the gaming industry due to its rapid convergence and low computational cost. This method attempts to minimize position and orientation errors by transforming one joint variable at a time. However, it often biases in describing unrealistic movements for robot manipulators.

The FABRIK solution, proposed by [24], consists of an iterative method that treats each joint value as points in the Cartesian plane and thus searches for subsequent joints as points belonging to lines, representing the manipulator links. This approach has the main advantages of high convergence percentage and low computational cost. Nevertheless, even with all the advantages presented FABRIK has some limitations when applied to manipulator robotics. As we aim to propose a solution for these limitations, we present the FABRIK algorithm and its limitations in sections III and IV respectively, followed by the proposed approach in section V.

III. FORWARD AND BACKWARD REACHING INVERSE KINEMATICS (FABRIK)

FABRIK uses previously calculated joint positions to find new positions in an iterative way that aims to minimize system error by adjusting each joint angle one at a time. The algorithm starts initially in the base/end-effector direction and returning in the end-effector/base direction. Fig. 1 shows an illustration of the FABRIK algorithm where $p_1, \ldots, p_n$ are positions of the manipulator joints, $d_1, \ldots, d_{n-1}$ the manipulator links and t corresponds to the target position.

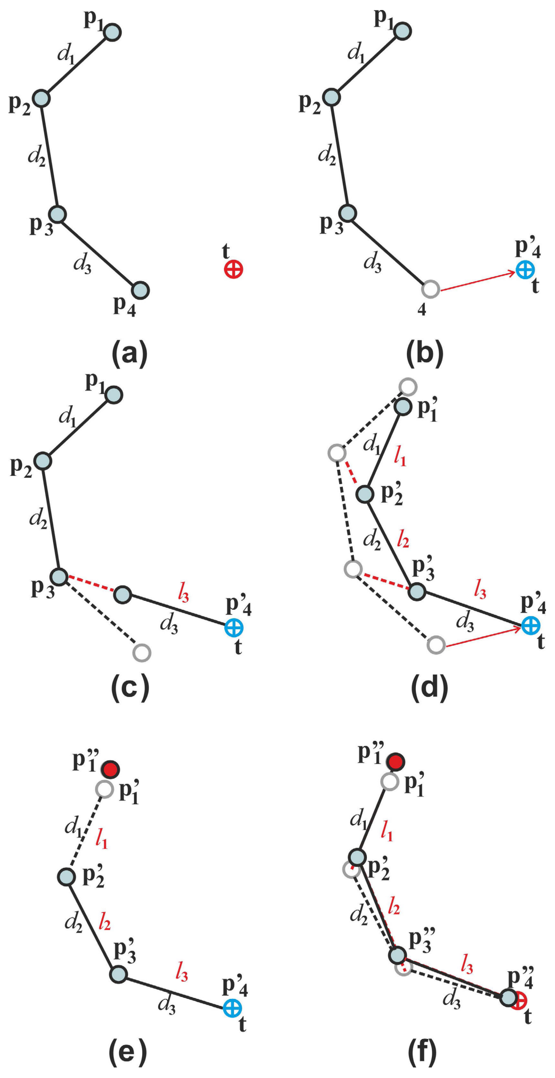


FIGURE 1. Example of a complete FABRIK iteration for a single target and a 4-DOF manipulator. (a) manipulator starting position and target, (b) p_4 end-effector move to target, (c) determination of new joint position p_3, (d) end of forward reaching stage, (e) beginning of backward reaching, (f) target position reached [24].

The first step in performing FABRIK is to calculate the distances between joints and verify that the target is reachable. When it is reachable, the approach is implemented in two stages. In the first stage, known as **Forward Reaching**, joint positions are found starting from the end-effector to base, assigning the end-effector a new value p'_n that corresponds to the target point. Then a connection is made between p'_n and p_{n-1} and the new value of p_{n-1} is given by the size of the d_{n-1} link (which connect the joints p_n and p_{n-1}) starting at the point p'_n. This procedure is repeated until reaching the manipulator base, as presented in Fig. 1(b), (c) and (d). However, this process can move the base out of its real position, as presented in Fig. 1(d). Then, a second phase is necessary and the same process is performed in the base-effector direction. In this stage, called **Backward Reaching**, the base is fixed in its original position and the other joints are re-positioned one by one until the end-effector, as shown in Fig. 1(e) and (f). The two phases of FABRIK are performed until the end-effector position reaches the target or gets close enough.

FABRIK has several advantages over existing iterative heuristic algorithms [24]. The computational cost for each set per iteration is low, which means the solution is reached quickly. The method is also easy to implement, as it is simply a problem involving points, distances and lines, and it always returns a solution when the target is reachable. Another advantage of this approach is the fact it does not require complex calculations or matrix manipulations, does not suffer from singularity problems and returns smooth motion without erratic discontinuities. Besides, its emphasis on joint movements closer to the base ensures a closer simulation of natural movements than that observed with other methods such as CCD.

Despite the advantages presented, several methods have been proposed to extend FABRIK. [25] proved that FABRIK can handle different priorities for its targets. In [26], a data method was applied before the start of FABRIK for producing more natural poses for the human eye, while [27] extended FABRIK to handle collision-free tasks. Finally, in the work developed by [4] FABRIK extensions are presented to meet the most varied types of joints models.

IV. FABRIK EXTENSION LIMITATIONS FOR MOVEMENTS CONSTRAINTS

Aristidou *et al.* [4] presents the six most common anthropometric joints and describes how to incorporate joint constraints using FABRIK. Among these joints, the Ball-and-Socket, hinge and pivot joints are the most common in manipulator robotics.

The Ball-and-Socket joint allows rotational movements in any direction. This is the type of joint in the human body that allows as many different movements as flexion, extension, rotation, abduction, adduction and circumduction. Hinges, on the other hand, are the simplest type of joint, allowing flexion and extension, i.e. movements limited to a flat/direction around a single axis. They can be found on the knees and elbows. Finally, the pivot joint is one in which a bone rotates around another, allowing only rotational movement. This type of joint can be found at the neck, allowing a lateral turn of the head.

Considering the types of movements allowed by each joint type, the extension proposed by [4] suggests solutions to integrate the constraints in FABRIK, complementing those presented in [24]. They propose that hinge joint restriction must be applied using the root and target, as shown in Fig. 2. In this example, the author uses a body of only three points: the hinge joint, where the constraint will be applied, the root and the target. However, it is unclear how to apply this method to a body with more than 3 joints or how to define who is the root and the target. The lack of definition allows different interpretations of this approach applied in a p_i joint in the forward reaching stage, such as:

1) Set root as the base of the manipulator and target as the end-effector.
2) Set root as p_{i+1} and target as p_{i-1}.
3) When in a hinge chain the method should be applied joint after joint, i.e, sequentially constrained, p_i, p_{i-1}, p_{i-2} until the end of the chain.
4) When in a hinge chain the method should set p_i and p_{i-1}, i.e, sequentially constrained, p_i, p_{i-2}, p_{i-4} until the end of the chain.

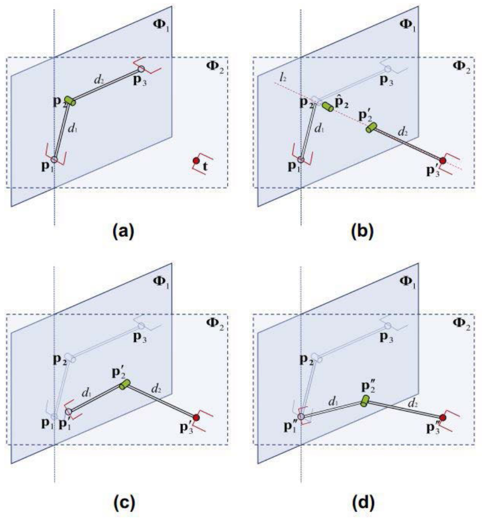


FIGURE 2. Applying hinge joint constraints. (a) Initial configuration and target. Φ_1 represents the plan where p_2 joint movement is allowed. Φ_2 is set by the root and target, which is oriented. (b) Reallocate and reorient the p'_3 joint to the target t. Then, project p_2 onto the plane Φ_2, generating $\widehat{p}_2$ and find p'_2 on line l_2 which passes from the joint p'_3 and point $\widehat{p}_2$ and has distance d_2 of p'_3. Reorient the new joint according to Φ_2. (c) Move and reorient p_1 to p'_1 which is on the line l_1, between the joint p'_2 and p_1 and has distance d_1 from p'_2. (d) Now it is a 2D problem, since all joints are in the plan. [4].

Despite the uncertainty in the method application, we assume that the method is applied following hypotheses 2 and 3 to discuss other points that also have flaws in this approach.

The definition of the Φ_i plane is one of the controversial topics, as it disregards possible movement restrictions in the joints connected to the hinge. Thus, the method can be functional only if the root and the target have total freedom of movement. This restriction is possibly not considered in the algorithm because FABRIK was originally developed for computer graphics applications. However, when considering the use of FABRIK for manipulator robotics, most robot models have a kinematic chain with joints of only 1-DOF, being impossible to apply the approach as presented.

The limitation of the method in a 1-DOF joint sequence can be better understood by analyzing its application in a manipulator similar to the one in Fig. 3(a). Fig. 3(b) shows the forward reaching stage to set the new position of p_3, which should be found after the Φ_3 plan definition. This definition is supposed to be done as presented in Fig. 2 (a) and (b). However, since the joint p'_3 and p'_4 have the same direction vector and p_2 is out of the plane defined by this vector, it is impossible to define the plane Φ_3 as proposed by [24].

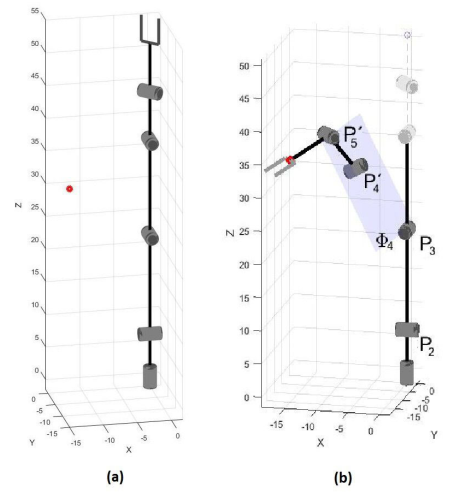


FIGURE 3. Hinge joint constraint fails. (a) Initial configuration of the robot manipulator and the target in red. (b) Is not possible to set Φ_3 once p'_3 and p'_4 are parallel and p_2 is out of the plane defined by these vector.

Although the application can not define the new position of p_3, a new value is assigned to p'_3 assuming that p'_4 did not rotate, so that the method can be analyzed in terms of p_2. In this case, the joints do not have the same direction of action, but the definition of the Φ_2 plan is not as simple as described in the method, because, p'_3's orientation does not make it obvious to choose the plan, as shown in Fig. 4. Therefore, there is a need for Φ_2 to be a plan which goes through p_1 and p'_3 and respects both the p'_3 and p_2 restriction.

Another major limitation of the approach can be noticed by analyzing Fig. 2 (b) where it can be concluded the point $\widehat{p_2}$ does not necessarily point in the same direction as the end-effector. Thus p'_2 point forces a d_2 move, which is only possible with full freedom of movement by the end-effector or changing the desired final orientation. Nevertheless, changing orientation is not an option for many applications in robotics, such as fine handling of any type of object or grasping activities.

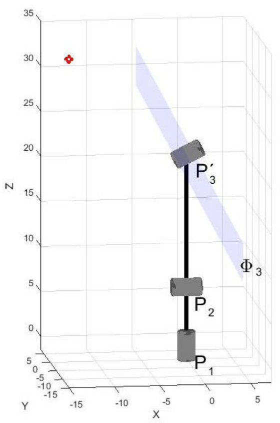


FIGURE 4. In this case, the definition of Φ_2 is not obvious.

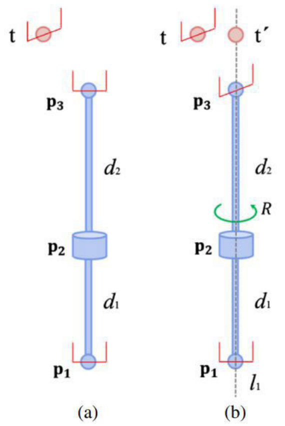


FIGURE 5. Application of pivot joint constraint. (a) Initial configuration and target. (b) The target t is projected in line l_1 which passes from p_1 to p_2. p_3 reoriented to meet destination orientation. [4].

The extension proposed by aristidou *et al.* to constrain pivot joints is shown in Fig. 5. The author proposes to project the destination on a line (l_1) that passes through the joint to be constrained (p_2) and the previous joint (p_1) and reorient p_2 to meet destination orientation. However, the destination can have an orientation that cannot be executed by p_2. Furthermore, if the pivot joint is not the end-effector of the chain and its next joint is a hinge, the reorientation as proposed can risk the capacity of reaching a solution.

The problems mentioned in this section can be noticed using the manipulator model presented in Fig. 6. This model consists of a robot with a pivot joint at its base and a sequence of hinge joints with the same direction vector. After the forward reaching stage, presented in Fig. 6(b), a new point p_2 must be reoriented to reach p_2'. Nevertheless, it cannot be executed due to the manipulator model shown in Fig. 6(a). Even if the direction from p_2' pointing to p_3' were used to reorient the new p_2, this plane couldn't reach the target because all the next joints have the same direction vector. In this case the algorithm can fall into a singularity.

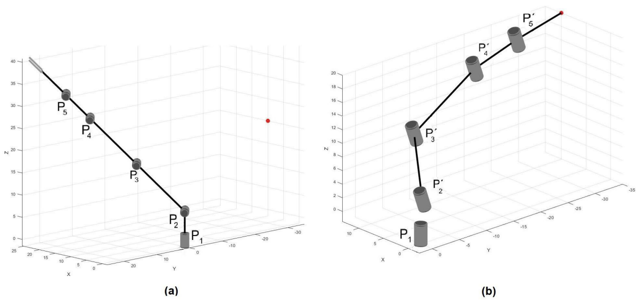


FIGURE 6. Problems from [4] approach for pivot joint. (a) Robot manipulator model and target. (b) Begin of backward reaching stage.

V. FABRIK-R: AN EXTENSION FOR MANIPULATOR ROBOTS

Based on the problems of the extension proposed by Aristidou *et al.* in [4], this section presents alternative approaches to solve the problems of movement restriction of the pivot and hinge joints, when applied for manipulators with chains composed by joints of only 1-DOF.

At forward reaching stage, we propose the appliance of the p_i joint constraint considering the possible constraints of the previous joint $p_{i+1}(p_{prev})$. Thus, we aim to determine a plane Φ_i that respects two rules:

1) p_{prev} constraints will not be violated.
2) Φ_i contains p_{next} and p'_{prev} (Fig. 7 (a)).

The first step is to define the plane Φ_{prev}, which is known once p'_{prev} is set (Fig. 7 (b)). Then, Φ_{prev} generates a $\widehat{p}_i$, according to this plane constraint, that has distance d_{prev} from p'_{prev} (Fig. 7 (c)). The new p_i is calculated by rotating $\widehat{p}_i$ around the vector $\vec{n}_{prev}$ normal to the plane Φ_{prev}, which ensures that p'_{prev} constraints will not be violated.

The second rule is achieved when the actuation vector of $\widehat{p}_i$ is orthogonal to $\overrightarrow{p_{next}p_{prev}}$, which set the new plane Φ_i (Fig. 7 (d) and (e)). After determining Φ_i, the d_i link must be rotated around the joint p'_{prev} using quaternions [28] to position p_i in the Φ_i action plane (Fig. 7 (f)). The procedure is described at algorithm 1 and has analogous behavior at backward reaching stage considering p_{prev} as p_{i-1}.

Algorithm 1 FABRIK-R

```
Require: Joint positions $p_i (i = 1, \ldots, n)$, target $\mathbf{t}$, distance
    between joints $d_i = |p_{i+1} - p_i| (i = 1, .., n)$.
Ensure: $p'_{next}$
 1: DEFINE_$\Phi_{prev}$()
 2: $[\widehat{p}_i\ \vec{\widehat{v}}_i]$ = CREATE_NEW_Pi($\Phi_{prev}, p'_{prev}$)
 3: $\theta$ = DEFINE_$\Phi_i(\vec{v}_{prev}, p'_{prev})$
 4: $p'_{next}$ = $ROT\_QUATERNIONS(\vec{v}_{prev}, \widehat{p}_i, \vec{\widehat{v}}_i, \theta)$ {This
    function rotates $\vec{\widehat{v}}_i$ and $\widehat{p}_i$ around $\vec{v}_{prev}$ by $\theta$ degrees}
```

As mentioned in the previous section, finding the Φ_i plan that respects constraints without mathematical calculations is a non-trivial task, so this article presents an equation to determine $\vec{n}_i$. The first requirement imposed by the limitation is

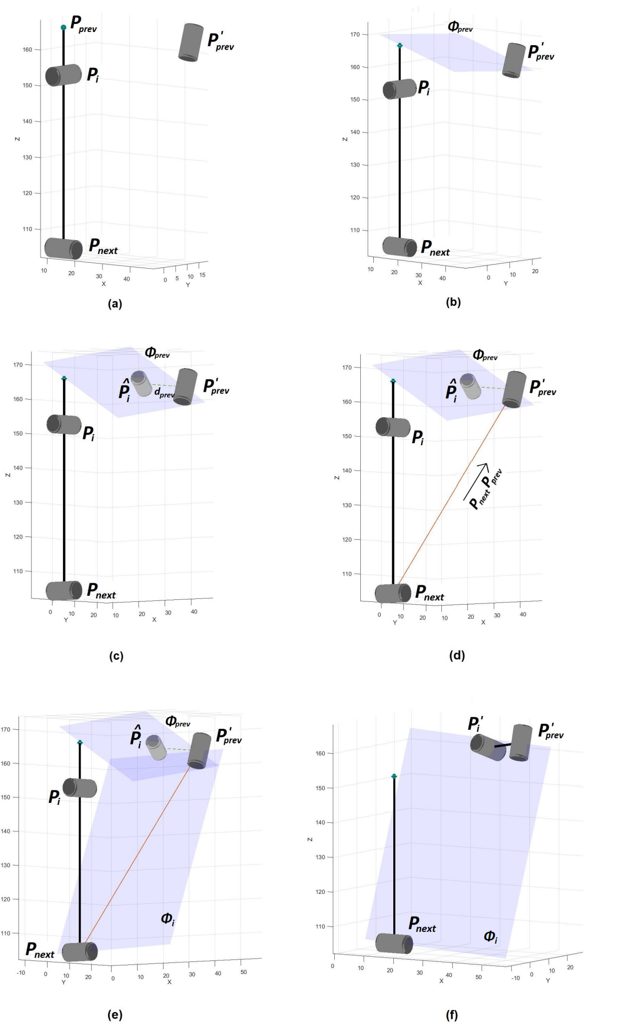


FIGURE 7. Example of a FABRIK-R step in the Forward Reaching phase. (a) The new p'_{prev} value is known before the new definition of the p_i joint; (b) Define Φ_{prev} based on p'_{prev}; (c) Generate a new $\widehat{p}_i$ that respects p'_{prev} constraints; (d) Create the vector $\overrightarrow{p_{next}p_{prev}}$ to calculate the plane Φ_i that contains p_{next} and p_{prev}; (e) Define Φ_i orthogonal to $\overrightarrow{p_{next}p_{prev}}$ and that respects the constraint from p'_{prev}; (f) Set the new value of p'_i in the Φ_i action plane.

that $\vec{n}$ must be orthogonal to $\overrightarrow{p_{next}p_{prev}}$. Therefore, the internal product properties between vectors will be used to guarantee the orthogonal relationship as

$$cos(\Theta) = \frac{\vec{n} \cdot \overrightarrow{p_{next}p_{prev}}}{\|\vec{n}\| \, \|\overrightarrow{p_{next}p_{prev}}\|}. \quad (1)$$

Since vectors must have an orthogonal relationship, equation (1) can be simplified as

$$\vec{n} \cdot \overrightarrow{p_{next}p_{prev}} = 0, \quad (2)$$

considering all vectors as unitary.

In order to respect the predefined joint orientation relationship between p'_{prev} and p_i, the properties of quaternary algebra are used since the use of quaternions allows a vector to be freely rotated around any axis. As p'_{prev} is known, a random acting vector $\vec{v}$ is generated, which respects the relationship between p'_{prev} and p_i. Thus, by rotating the $\vec{v}$ actuation vector around the p'_{prev} actuation vector ($\vec{l}$), it is intended to determine $\vec{n}$ according

$$\vec{n} = cos(2\theta)\vec{v} + (1 - cos(2\theta))(\vec{l} \cdot \vec{v})\vec{l} + sen(2\theta)\vec{l} \times \vec{v}, \quad (3)$$

ensuring the restriction imposed by the p_{prev} joint is respected.

The system comprised by equations (2) and (3) has as solution the vector $\vec{n}$. To solve the proposed system, the calculations are performed for each vector component $\vec{n}$ using the following definitions for the vectors:

$$\begin{aligned} \vec{n} &= (n_1, n_2, n_3) \\ \vec{v} &= (v_1, v_2, v_3) \\ \vec{l} &= (l_1, l_2, l_3) \\ \overrightarrow{p_{i-1}p_{i+1}} &= (\alpha, \beta, \gamma) \\ \vec{l} \times \vec{v} &= (t_1, t_2, t_3). \end{aligned}$$

Applying these definitions to equations (2) and (3), we get

$$n_1 = -\left(\frac{n_2\beta + n_3\gamma}{\alpha}\right), \quad (4)$$

$$\begin{aligned} n_1 &= cos(2\theta)v_1 + (1 - cos(2\theta))(\vec{l} \cdot \vec{v})l_1 + sen(2\theta)t_1 \\ n_2 &= cos(2\theta)v_2 + (1 - cos(2\theta))(\vec{l} \cdot \vec{v})l_2 + sen(2\theta)t_2 \\ n_3 &= cos(2\theta)v_3 + (1 - cos(2\theta))(\vec{l} \cdot \vec{v})l_3 + sen(2\theta)t_3. \quad (5) \end{aligned}$$

Replacing the equation (4) in (5) yields

$$0 = cos(2\theta)K_1 + (1 - cos(2\theta))K_2 + sen(2\theta)K_3, \quad (6)$$

with:

$$\begin{aligned} K_1 &= \alpha v_1 + \beta v_2 + \gamma v_3 \\ K_2 &= \vec{l} \cdot \vec{v}\,(\alpha l_1 + \beta l_2 + \gamma l_3) \\ K_3 &= \alpha t_1 + \beta t_2 + \gamma t_3. \end{aligned}$$

In equation (6) the only parameter to be determined is the value of θ, so it is possible to find the vector $\vec{n}$, which is normal to the Φ_i plane and perform step 3 of the proposed algorithm. However, since (6) is a nonlinear equation, it is common to find more than one possible solution to the problem, as seen in the example of Fig. 8.

In Fig. 8(c), two different radius circumferences may be noticed. They represent the range of each new position p'_4, considering the d_3 link is large enough to find the p_3 joint. This approach was used to visually prove the solution's validity. The choice of the point p'_4 should have as a deciding factor the circumference that has the smallest radius, with solution 1 being chosen to proceed with the algorithm.

When joints have the same direction of action, the equation (6) can present no solution (Fig. 3). Therefore, an additional step is placed in the Φ_i plan definition defining $\vec{v}$ as the next joint that has a different acting vector, as shown in algorithm 2.

The procedures described in algorithms 1 and 2 allow a kinematic chain composed of only 1-DOF joints to reach an

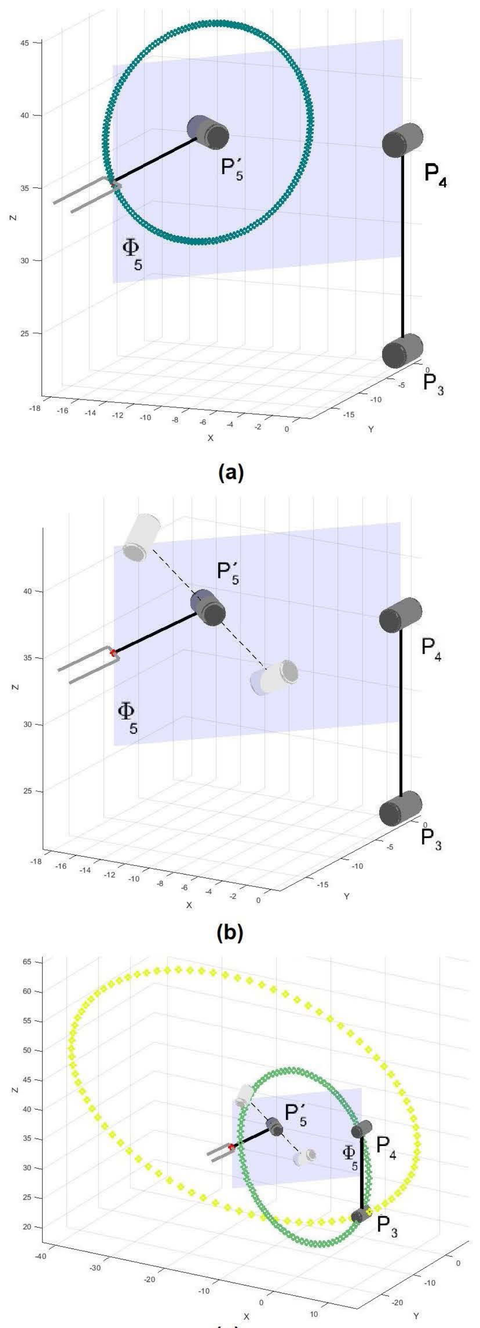


FIGURE 8. Joint definition p'_4. (a) Possibility set of the new joint p'_4 (in green). (b) Using equation (6) we find these two possible solutions. (c) Possibility set for joint p'_3 given the solutions found in (b), disregarding the size of the link (green and yellow set represents solution 1 and 2, respectively).

inverse kinematic solution which would not be possible using the original approach of FABRIK or its extension presented in [4].

VI. EXPERIMENTAL RESULTS

This section presents the results of implementations and tests with a manipulator model for validation purposes. First, we show simulated experiments using a 5-DOF robot manipulator composed only with hinge and pivot joints. Afterward, we compare our method, through simulation, with other IK solvers using a 10-DOF robot based on [24] experiments. At last, a real robot experiment is performed using the Pioneer Arm.

FABRIK-R algorithm, as well as the original FABRIK, does not deal with obstacle avoidance. Thus, all experiments performed in this paper have an obstacle-free environment. Some extensions have been developed to deal with this problem by adding a probabilistic component when a collision is detected, but this extension has not been considered in our method yet.

Algorithm 2 DEFINE_Φ_i

```
Require: $\vec{l}, p_{prev}$
Ensure: $\theta$
 1: $j = FIND\_CONCURRENT(i, \vec{v}_{init})$
 2: $(\alpha, \beta, \gamma) = p_{prev} - p_j$
 3: $\vec{v} = GENERATE\_RANDOM(\vec{v}_{prev}, \vec{v}_j)$
 4: $t = \vec{l} \times \vec{v}$
 5: $K_1 = \alpha v_1 + \beta v_2 + \gamma v_3$
 6: $K_2 = \vec{l} \cdot \vec{v}\,(\alpha l_1 + \beta l_2 + \gamma l_3)$
 7: $K_3 = \alpha t_1 + \beta t_2 + \gamma t_3.$
 8: $A.SOLVE(cos(2\theta)K_1 + (1 - cos(2\theta))K_2 + sen(2\theta)K_3)$
 9: $S.ROT\_QUATERNIONS(\vec{l}, \vec{v}, A)$
10: $\theta = S.COMPARE\_SOLUTION()$
```

A. SIMULATED EXPERIMENTS

The first experiment uses a robot model that has a single pivot joint at his base followed by 4 hinge joints as shown in Fig. 9. The first step in the forward reaching stage is to position the end-effector at the target, followed by a new definition of the joint p_5, which should be constrained, once Φ_5 is set. A bad choice to Φ_5 can risk the method capacity to achieve a solution, such as proposed by [24] with the root and the oriented target. In this case, the algorithm cannot reach the target after several iterations, converging to a different point such as a local minimum. Following the algorithm proposed in this paper, Φ_5 must be set connecting p_6 and the next joint with a different direction vector. In this case p_4, as shows Fig. 10.

The next step is to set the new value of p_4. Once the joint before is a hinge joint, Φ_5 is defined by its direction vector and $\widehat{p_4}$ is created respecting this constraint, as shown in Fig. 11(a). Following the algorithm, p_1 is set as the next joint concurrent of p_4 and then Φ_4 is calculated using equation (6). As a result, 2 solutions were obtained, as represented in Fig. 11(b), and the closest one to p_3 is chosen. Finally, $\widehat{p_4}$ is rotated around p'_5 to reach the solution in Fig. 11(c).

Fig. 12 presents the last forward reaching steps. Since p'_4 and p'_3 have the same direction vector, the plane Φ_4 is equal to Φ_3. In this case, all rotations around Φ_4 can't change Φ_4. Therefore p_3 must be projected onto Φ_3 and FABRIK's

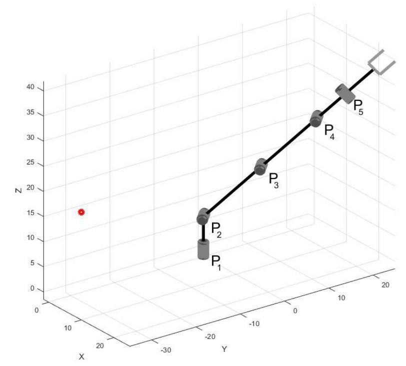


FIGURE 9. Robot Manipulator model and the target.

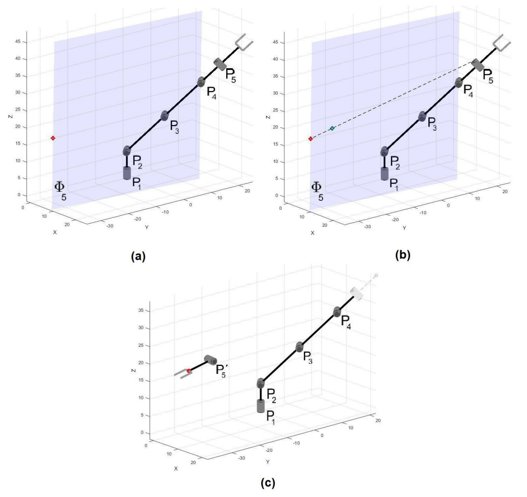


FIGURE 10. Constraining step of p_5 joint. (a) Set of Φ_5 as a plane that passes through t and p_4. (b) Find new p_5 as the point in the line l_5 which passes from t and the joint p_4 projected onto the plane Φ_5 that has distance d_5 of t. (c) Joint p_5 constrained.

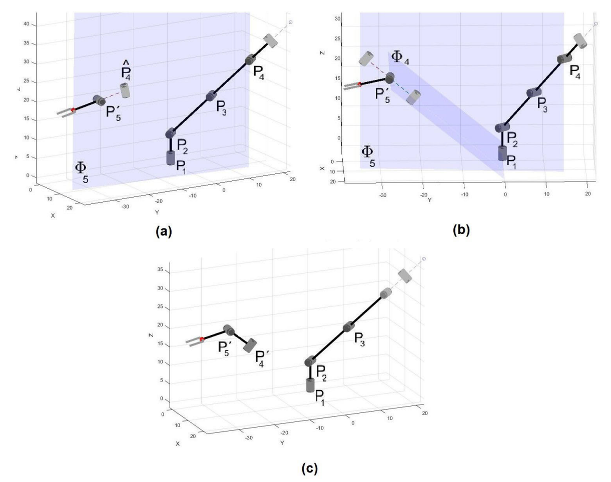


FIGURE 11. Constraining step of p_4 joint. (a) Φ_5 is defined by p_5 and $\widehat{p}_4$ is created based on the manipulator model. (b) Set of Φ_4 and possible solutions of equation (6). (c) New joint p'_4.

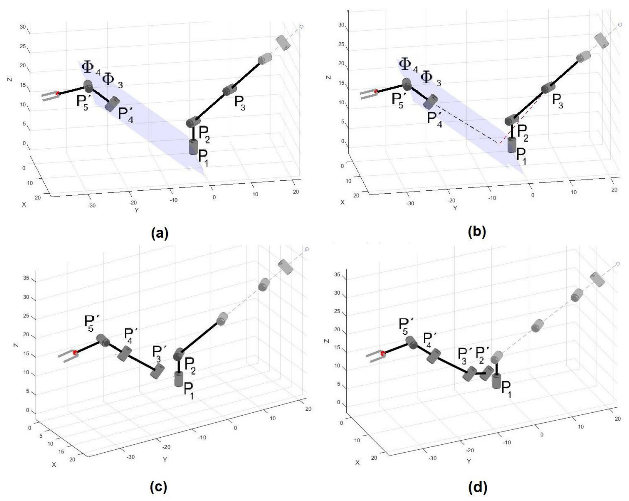


FIGURE 12. Forward Reaching stage. (a) p'_4 and p'_3 have the same direction vector which results in Φ_4 equals Φ_3. (b) Projection of p_3 onto plane Φ_3 (c) p'_3 is set as the point in the line l_3 which passes from p_4 and the joint p_3 projection onto Φ_3. (d) New joint p'_2.

original procedure is applied. Fig. 12(b) presents the projection and Fig. 12(c) the new value of p'_3. Similarly p'_2 is calculated as shown in Fig. 12(d).

The backward Reaching stage starts defining p_1 as the fixed point of the manipulator. Then, algorithm 1 is used to define the new p_2. First, Φ_{bef} (Φ_1) is set by p_1 joint. Then a random joint $\widehat{p}_2$ is created respecting p_1 constraints as shown in Fig. 13(a) and (b). The method proceeds to the definition Φ_2, which aims to connect p_1 and the next joint with different direction vector (p'_5), Fig. 13(c).

The FIND_CONCURRENT function in algorithm 2 is essential to allow this approach to reach a solution because if the joint immediately after p_2 were used to set Φ_2, the method could fall into a local minimum. Aristidou *et al.* method also produces a bad solution, once p'_2 projection in the line which passes from p_1 and $\widehat{p}_2$ represents an impossible direction vector and, even if a rotation was applied pointing to projection, the created plane would probably result in a singularity configuration.

A similar procedure to the forward reaching stage is then applied to the following joints. Fig. 14 presents the last steps until the end of the first iteration and the final solution founded after 3 iterations.

This experiment shows the efficacy of the proposed method to reach a solution when solving the IK problem to a kinematic chain composed of only 1-DOF joints. The model used in the tests (Fig. 9) has all the problems described in section IV, such as pivot-hinge joint sequence, hinge-hinge joint sequence with same actuation direction and hinge-hinge joint sequence with different actuation direction.

We tested 200 different target points, randomly distributed in the reachable space, and the FABRIK-R was always able to find a solution. In these tests, the original chain is $340mm$ long and the termination tolerance is $10^{-3}mm$. Table 1 presents the average runtimes of the FABRIK-R, as well as the number of

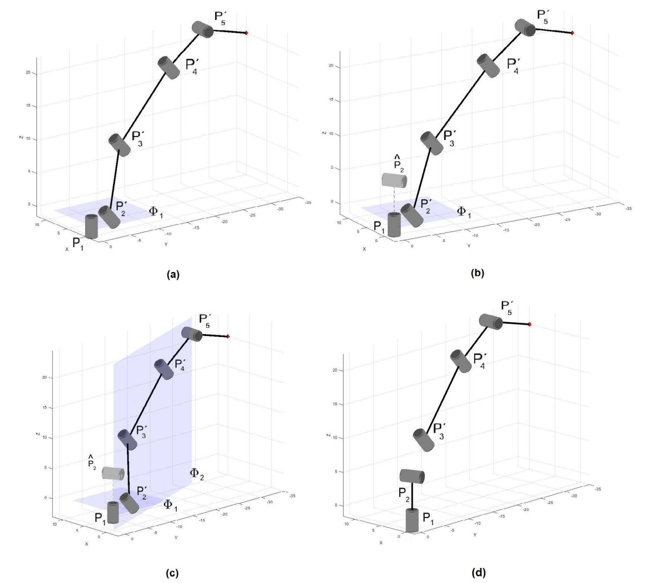


FIGURE 13. First step of backward reaching stage. (a) Set of plane Φ_1. (b) Creation of a random $\widehat{p}_2$ that respects p_1 constraints. (c) Set of plane Φ_2. (d) New joint p_2.

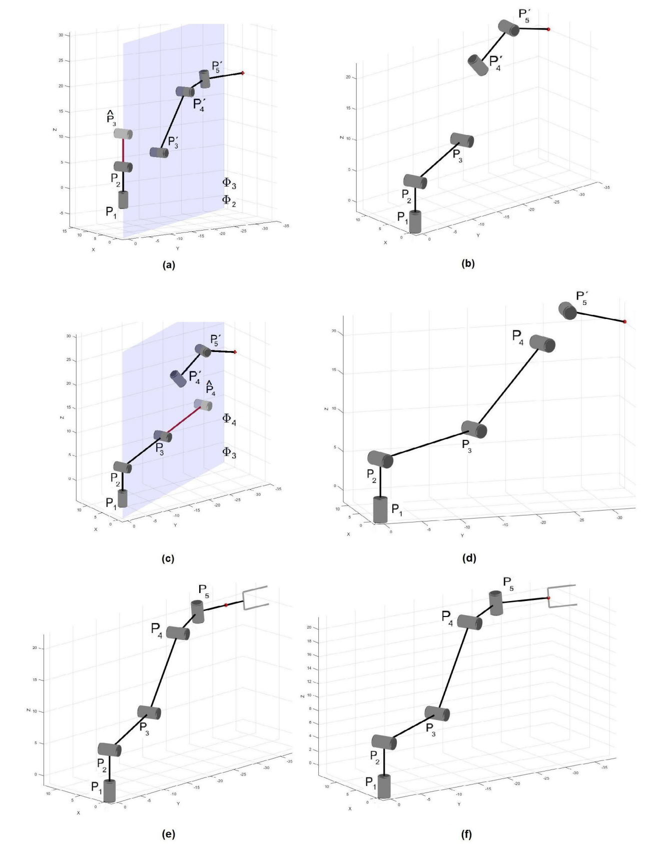


FIGURE 14. Set of joints p_3 and p_4. (a) Application of algorithm 1 for joint p_3. (b) New joint p_3. (c) Application of algorithm 1 for joint p_4. (d) New joint p_4. (e) End of the first iteration. (f) Solution found after 3 iterations.

iterations needed to reach the target. Runtimes are in seconds and were measured with custom MATLAB code on a Core i5 2.5GHz.

TABLE 1. Average results (over 200 runs) for a 5-DOF kinematic chain with joints of only 1-DOF.

Performance metrics	Average
Matlab Execution Time	0.03179s
Number of Iterations	19.4706
Time per Iterarion	0.00147s

The performance metrics in Table 1 show that even with the addition of the plane calculations complexity, the FABRIK-R keeps the features of fast convergence and a low number of iterations.

To compare our method with the main algorithms of the inverse kinematics area, we performed an experiment similar to the one described in the paper of the original FABRIK [24]. In this article, we compare FABRIK-R with CCD, Jacobian Transpose, and Jacobian DLS, using performance metrics as convergence time, number of iterations, and time per iteration. The experiment was performed with an original chain with 10 joints and $900mm$ long without movement constraints, and a termination tolerance of $10^{-3}mm$.

The main difference between the experiment described in [24] and ours is that we added movement constraints in our manipulator in order to make it a kinematic chain with only 1-DOF joints, as shown in Fig. 15. The model of the robot used in the experiment is described using the Denavit-Hartenberg parameters in Table 2.

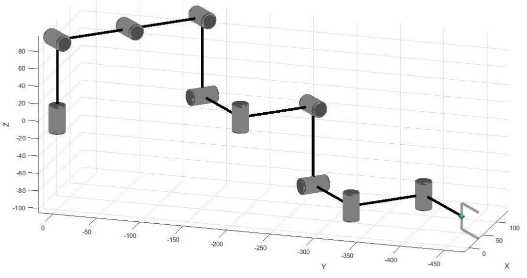


FIGURE 15. Manipulator model based on the DH parameters of Table 2.

TABLE 2. Denavit-Hartenberg table for the 10-DOF manipulator. Length parameters are in terms of millimeters (mm).

Joint	a	α	d	θ
1	0	$-\pi/2$	90	θ_1
2	90	0	0	θ_2
3	90	0	0	θ_3
4	90	$-\pi/2$	0	θ_4
5	90	$-\pi/2$	0	θ_5
6	90	$-\pi/2$	0	θ_6
7	90	$-\pi/2$	0	θ_7
8	90	$-\pi/2$	0	θ_8
9	90	0	0	θ_9
10	90	0	0	θ_{10}

We tested 200 different target points, randomly distributed in the reachable space, considering the same restriction used

TABLE 3. Average results (over 20 runs) for a single kinematic chain with 10 joint obtained by Aristidou in [24] and average results of FABRIK-R (over 200 runs) for a single kinematic chain with 10 joints constrained to allow only 1-DOF per joint.

IK Solver	Number of Iterations
FABRIK	15.461
FABRIK-R	20.4297
CCD	26.308
Jacobian Transpose	1311.190
Jacobian DLS	998.648

TABLE 4. Average results (over 200 runs) for a single kinematic chain with 10 joints constrained to allow only 1-DOF per joint.

Performance metrics	FABRIK-R	CCD
Matlab exe. Time	0.072375s	1.079251s
Number of Iterations	20.4297	106.7959
Time per Iterarion	0.003544s	0.009341s

in [24] in which the distance between the target and the end effector must be at least 600mm. The FABRIK-R was able to find a solution for all targets.

The performance of our method over these 200 runs was measured in terms of convergence time, number of iterations, and time per iteration. Regarding the number of iterations, we present a comparison in Table 3, based on the results presented in [24]. It is important to highlight that all these approaches, except the one proposed in this paper (FABRIK-R), are dealing with an unconstrained chain, which makes the task easier. Despite this, our solver performed with a lower number of iterations than the other methods even dealing with a constrained chain.

Concerning the convergence time, a comparison with the data presented in [24] is not suitable, due to the difference in the processor used for the experiments. Therefore, the CCD algorithm was implemented for the same kinematic chain presented in Table 2, and the experiments were made under the same conditions as the FABRIK-R. The Jacobian methods were not considered in this analysis, since these approaches present a significantly slow convergence time, as stated in [24].

Table 4 shows that our method is approximately fifteen times faster than CCD and needs approximately 5 times fewer iterations to reach the target. These results show that the proposed method in this paper reaches a solution, when possible, for the IK problem applied to a kinematic chain composed of only 1-DOF joints maintaining the main advantages of the algorithm of FABRIK.

B. REAL EXPERIMENTS

We applied the FABRIK-R in the Pioneer manipulator robot to build a case for real test validation. This robot has seven joints, three pivot joints and four hinge joints, and its Denavit-Hartenberg parameters are shown in Table 5. The experiment was performed for ten different target points, with the same initial configuration which has the end-effector at (0cm, 0cm, 59cm) and termination tolerance of 1cm. The Pioneer robot is presented in Fig. 16 in the initial configuration used for the experiments.

TABLE 5. Denavit-Hartenberg table for the Pioneer Arm. Length parameters are in terms of millimeters (mm).

Joint	a	α	d	θ
1	0	$-\pi/2$	150	θ_1
2	0	$-\pi/2$	0	θ_2
3	0	$-\pi/2$	150	θ_3
4	0	$-\pi/2$	0	θ_4
5	0	$-\pi/2$	145	θ_5
6	75	$\pi/2$	0	θ_6
7	75	0	0	θ_7

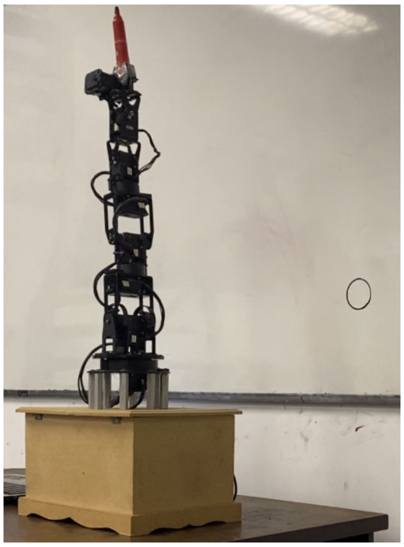


FIGURE 16. Pioneer manipulator - initial configuration.

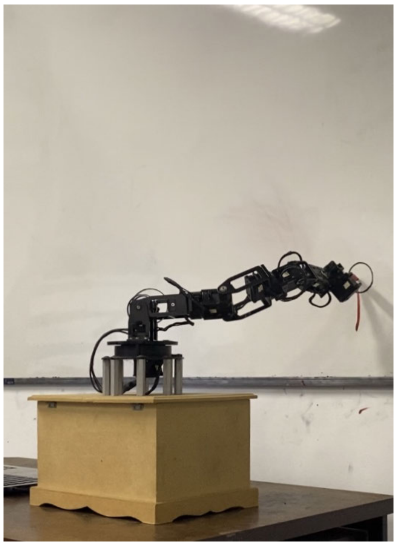


FIGURE 17. Pioneer manipulator - Final configuration.

TABLE 6. Results for real experiments performed with a Pioneer Manipulator using FABRIK-R to solve the IK problem for 10 target points.

Target [x, y, z]	θ_1 (°)	θ_2 (°)	θ_3 (°)	θ_4 (°)	θ_5 (°)	θ_6 (°)	θ_7 (°)	$\mathcal{E}$
[-10cm, 20cm, 15cm]								
Measured	72.4340	231.6715	44.8680	68.3284	173.0205	223.4604	137.8299	2.4576cm
Calculated	72.0357	232.9128	43.3716	69.2559	172.3524	223.2029	138.8450	0.0040cm
[0cm, 30cm, 20cm]								
Measured	289.1496	233.4310	200.8798	97.9472	239.5894	230.7918	173.0205	1.6959cm
Calculated	289.1147	235.2849	202.8857	100.5722	240.3555	232.0421	172.8121	0.8720cm
[30cm, 30cm, 10cm]								
Measured	272.7273	234.6041	161.2903	127.5660	220.5279	174.1935	118.1818	2.8404cm
Calculated	273.7206	238.8962	162.5669	130.0883	221.3856	174.0569	117.9238	0.3881cm
[30cm, 0cm, 15cm]								
Measured	299.4135	211.1437	97.0674	58.3578	97.3607	187.9765	181.2317	2.5531cm
Calculated	300.6636	211.9604	94.5865	59.9430	96.6166	188.9827	181.7716	0.1894cm
[10cm, 20cm, 30cm]								
Measured	149.2669	158.6510	272.1408	57.4780	223.4604	215.5425	150.4399	1.3029cm
Calculated	149.8723	159.8053	273.1128	60.5343	224.4086	216.4675	150.1219	0.0002cm
[-20cm, -25cm, 10cm]								
Measured	149.8534	200.5865	115.5425	72.7273	130.4985	180.6452	168.9149	1.4112cm
Calculated	150.0939	200.3210	114.4607	75.1049	130.3213	180.7206	169.4944	0.0181cm
[30cm, -25cm, 20cm]								
Measured	173.9003	212.3167	208.5044	81.8182	0.5865	184.1642	170.0880	0.8254cm
Calculated	174.1269	212.2136	210.1987	83.6652	-0.2260	184.7386	171.2706	0.0162cm
[10cm, -30cm, 5cm]								
Measured	187.0968	231.3783	172.7273	117.5953	59.2375	229.9120	137.8299	0.5781cm
Calculated	187.3628	231.8674	172.7498	119.6384	58.4168	230.7213	138.7609	0.9812cm
[-20cm, -30cm, 5cm]								
Measured	145.4545	228.7390	114.9560	107.3314	112.9032	191.4956	141.0557	1.0279cm
Calculated	146.0524	229.3575	113.6657	109.7320	112.1990	192.1306	141.2146	0.0022cm
[0cm, -30cm, 15cm]								
Measured	132.8446	234.0176	137.8299	118.1818	298.8270	233.7243	223.1671	1.6758cm
Calculated	133.0709	239.5468	138.8185	121.5291	320.9745	260.8913	223.0277	0.2807cm

For all 10 target points, the FABRIK-R algorithm was able to generate valid solutions that took the robot to the destination point correctly in few iterations. However, the Pioneer arm actuators are not able to perform the calculated joint angles perfectly. In Table 6 we present a comparison between the joint angles calculated by the algorithm and ones performed by the manipulator, which were measured according to the Pioneer joint sensors. The results show that small errors are associated with each joint and for this reason, the end-effector position estimated according to the measured angles has a higher error than it was supposed to be according to the position calculated by the algorithm. The errors ($\mathcal{E}$) between the target point and the end-effector are presented in the last column of the table for both calculated and measured end-effector positions.

Nevertheless, it is important to highlight that the higher errors of the measured values are not caused by the algorithm, but due to hardware limitations. FABRIK-R provided an error of 0.2752 cm and 1.7 iterations on average in 0.006243s of execution time. Better results can be obtained by using a robot with a better response or using visual feedback to get the real end-effector position.

Despite the hardware limitation, the mean error of the real experiment simulated is only 1.6368 cm, which shows that even with an error higher than expected, all solutions are close to the target points. In order to have a better visualization of the algorithm performance, we attached a whiteboard marker to the end-effector of the manipulator and set the target point on a whiteboard with a circle around it. Fig. 16 shows the task is completed successfully.

The results obtained with the real experiments corroborate the objective of FABRIK-R that is to solve the IK problem for 1-DOF kinematic chains, since the application of the original FABRIK to this robot, or some of the main industrial robots as the SCARA, would not be possible without a modification on the method. Therefore, FABRIK-R solves IK for a kinematic chain with joints of only 1-DOF and maintains most of the advantages from the original FABRIK, such as simplicity, low computational cost and fast convergence.

VII. CONCLUSION

This paper presents a new approach for inverse kinematics of manipulators robots based on FABRIK in order to deal with the most commom kinematic chain manipulator robots that is a chain with joints of only 1-DOF. The method was named FABRIK-R, and it maintains the main advantages of the original algorithm, a fast IK solver with low computational cost, which allows real-time applications for manipulators.

In [4] was proved FABRIK always find a solution for unconstrained bodies, once the target is reachable. However, with constrained bodies, this approach can suffer from singularities. The method we present shows a better performance in these cases finding a solution for all tests performed. Nevertheless, once it is necessary to set planes based on restrictions, some calculations were added which makes this approach a little more complex than the original method.

Despite the increased complexity, our method still keeps the main characteristics of FABRIK which are the fast convergence and a small number of iterations. We also showed it to be faster than important IK solvers for manipulators robots as CCD, Jacobian Transpose, and Jacobian DLS. The FABRIK-R treats each joint considering the restrictions of the previous (parent) and next (child) joint. However, this attribute doesn't result in a dependency of the entire kinematic chain and permits the algorithm to solve each step locally. Therefore, changes in the number of joints of the manipulator won't decrease its performance in comparison with other methods.

FABRIK-based approaches have limitations in applications that require orientation control and an environment with obstacles. These methods only provide a unique solution, not considering the orientation. Therefore, future works may investigate the possibility to extend the approach to deal with obstacle avoidance and orientation control problem. Another direction for future works is to provide a formal proof that this approach can always find a solution if reachable.

Moreover, this paper presents a mathematical formalism that will serve as a basis for future works with FABRIK in robotics of manipulators.

REFERENCES

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. Cambridge, MA, USA: MIT Press, 2011.

[2] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. London, U.K.: Pearson, 2009.

[3] M. W. Spong *et al.*, *Robot Modeling and Control*. Hoboken, NJ, USA: Wiley, 2006.

[4] A. Aristidou, Y. Chrysanthou, and J. Lasenby, "Extending FABRIK with model constraints," *Comput. Animation Virtual Worlds*, vol. 27, no. 1, pp. 35–57, Jan. 2016.

[5] S. Starke, N. Hendrich, and J. Zhang, "Memetic evolution for generic full-body inverse kinematics in robotics and animation," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 406–420, Jun. 2019.

[6] L. Zhang, X. Yan, and Q. Zhang, "Design and analysis of 3-DOF cylindrical-coordinate-based manipulator," *Robot. Comput.-Integr. Manuf.*, vol. 52, pp. 35–45, Aug. 2018.

[7] C. Urrea and J. Cortés, "Design, construction and control of a SCARA manipulator with 6 degrees of freedom," *J. Appl. Res. Technol.*, vol. 17, no. 2, pp. 92–103, Oct. 2019.

[8] C. López-Franco, J. Hernández-Barrágn, A. Y. Alanis, N. Arana-Daniel, and M. López-Franco, "Inverse kinematics of mobile manipulators based on differential evolution," *Int. J. Adv. Robotic Syst.*, vol. 15, no. 1, Jan. 2018, Art. no. 1729881417752738.

[9] A. El-Sherbiny, M. A. Elhosseini, and A. Y. Haikal, "A comparative study of soft computing methods to solve inverse kinematics problem," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 2535–2548, Dec. 2018.

[10] A. Campeau-Lecours, V. Maheu, S. Lepage, H. Lamontagne, S. Latour, L. Paquet, and N. Hardie, "JACO assistive robotic device: Empowering people with disabilities through innovative algorithms," in *Proc. Annu. Conf. Rehabil. Eng. Assistive Technol. Soc. North Amer. (RESNA)*. Washington, DC, USA: RESNA Press, 2016.

[11] M. Perrelli, P. Nudo, D. Mundo, and G. Danieli, "Robotic control of the traditional endoscopic instrumentation motion," in *New Trends in Mechanism Science*. Dordrecht, The Netherlands: Springer, 2010, pp. 449–456.

[12] E. I. Veliev, R. F. Ganiev, V. A. Glazunov, G. S. Filippov, and A. N. Terekhova, "Formulation and solution of the problem of the positions of a mechanism with a parallel–series structure used in surgery as an alternative to the DA VINCI robot," *J. Mach. Manuf. Rel.*, vol. 48, no. 4, pp. 283–291, Jul. 2019.

[13] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," *Comput. Graph. Forum*, vol. 37, no. 6, pp. 35–58, 2018.

[14] C. Faria, F. Ferreira, W. Erlhagen, S. Monteiro, and E. Bicho, "Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance," *Mech. Mach. Theory*, vol. 121, pp. 317–334, Mar. 2018.

[15] D. Zhang and B. Hannaford, "IKBT: Solving symbolic inverse kinematics with behavior tree," *J. Artif. Intell. Res.*, vol. 65, pp. 457–486, Jul. 2019.

[16] H. Su, Y. Hu, H. R. Karimi, A. Knoll, G. Ferrigno, and E. De Momi, "Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results," *Neural Netw.*, vol. 131, pp. 291–299, Nov. 2020.

[17] H. Su, W. Qi, C. Yang, J. Sandoval, G. Ferrigno, and E. D. Momi, "Deep neural network approach in robot tool dynamics identification for bilateral teleoperation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2943–2949, Apr. 2020.

[18] N. Courty and E. Arnaud, "Inverse kinematics using sequential Monte Carlo methods," in *Proc. Int. Conf. Articulated Motion Deformable Objects*. Berlin, Germany: Springer, 2008, pp. 1–10.

[19] A. Reiter, A. Muller, and H. Gattringer, "On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1681–1690, Apr. 2018.

[20] X. Wang, J. Cao, X. Liu, L. Chen, and H. Hu, "An enhanced step-size Gaussian damped least squares method based on machine learning for inverse kinematics of redundant robots," *IEEE Access*, vol. 8, pp. 68057–68067, 2020.

[21] P. Savsani, R. L. Jhala, and V. J. Savsani, "Comparative study of different metaheuristics for the trajectory planning of a robotic arm," *IEEE Syst. J.*, vol. 10, no. 2, pp. 697–708, Jun. 2016.

[22] M Ayyıldız and K. Çetinkaya, "Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 825–836, May 2016.

[23] A. A. Canutescu and R. L. Dunbrack, "Cyclic coordinate descent: A robotics algorithm for protein loop closure," *Protein Sci.*, vol. 12, no. 5, pp. 963–972, May 2003.

[24] A. Aristidou and J. Lasenby, "FABRIK: A fast, iterative solver for the inverse kinematics problem," *Graph. Models*, vol. 73, no. 5, pp. 243–260, Sep. 2011.

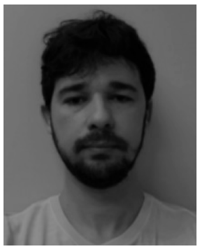
[25] S. Moya and F. Colloud, "A fast geometrically-driven prioritized inverse kinematics solver," in *Proc. 24th Congr. Int. Soc. Biomech. (ISB)*, Aug. 2013, pp. 1–3.

[26] S. Agrawal and M. van de Panne, "Task-based locomotion," *ACM Trans. Graph.*, vol. 35, no. 4, p. 82, 2016.

[27] Y. Xie, Z. Zhang, X. Wu, Z. Shi, Y. Chen, B. Wu, and K. A. Mantey, "Obstacle avoidance and path planning for multi-joint manipulator in a space robot," *IEEE Access*, vol. 8, pp. 3511–3526, 2020.

[28] S. C. de Biasi and M. Gattass, "Utilização de quatérnios para representação de rotações em 3-D," *Relatório Técnico, Tecgraf–Pontifícia Universidade Católica do Rio de Janeiro*. Rio de Janeiro, Brazil: PUCRIO. Disponível, 2007. [Online]. Available: http://www.tecgraf.pucrio.br/~mgattass

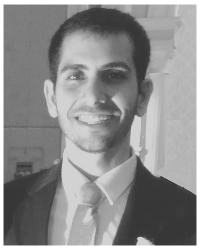
MATHEUS C. SANTOS received the bachelor's degree in computer engineering and the master's degree in electrical engineering from the Federal University of Sergipe, in 2019 and 2021, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Limerick. He is also a member of the Robotics Research Group of the Federal University of Sergipe (GPR-UFS).



LUCAS MOLINA received the bachelor's degree in electronic engineering from the Federal University of Sergipe (UFS), in 2007, the master's degree in control, automation and robotics from the Federal University of Rio de Janeiro, in 2010, and the Ph.D. degree in electrical engineering from the Federal University of Campina Grande, in 2014. He is currently an Adjunct Professor with UFS. He is also a Researcher with the Robotics Research Group of UFS (GPR-UFS).

ELYSON A. N. CARVALHO (Member, IEEE) received the bachelor's degree in electronic engineering from the Federal University of Sergipe, in 2006, and the master's and Ph.D. degrees in electrical engineering from the Federal University of Campina Grande, in 2007 and 2012, respectively. He is currently an Associate Professor III with the Federal University of Sergipe. He is also a Permanent Member of the Graduate Program in Electrical Engineering (PROEE).

EDUARDO O. FREIRE received the bachelor's degree in electrical engineering from the Federal University of Paraíba, in 1995, and the master's and Ph.D. degrees in electrical engineering from the Federal University of Espírito Santo, in 1997 and 2002, respectively. He is currently an Associate Professor IV with the Federal University of Sergipe.



JOSÉ G. N. CARVALHO received the bachelor's and master's degrees in electrical engineering from the Federal University of Sergipe, in 2010 and 2012, respectively, and the Ph.D. degree in automation and systems engineering from the Federal University of Santa Catarina, in 2016. He is currently an Assistant Professor with the Federal University of Sergipe.

PHILLIPE C. SANTOS received the bachelor's degree in electronic engineering and the master's degree in electrical engineering from the Federal University of Sergipe, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the Federal University of Campina Grande. He is also a Professor with the Federal Institute of Sergipe. He is also a member of the Robotics Research Group of the Federal University of Sergipe (GPR-UFS).