IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 16, 2020, accepted November 4, 2020, date of publication November 17, 2020, date of current version December 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038424

M-FABRIK: A New Inverse Kinematics Approach to Mobile Manipulator Robots Based on FABRIK

PHILLIPE CARDOSO SANTOS[1], RAIMUNDO CARLOS SILVÉRIO FREIRE[1], ELYSON ADAN NUNES CARVALHO[2], LUCAS MOLINA[2], AND EDUARDO OLIVEIRA FREIRE[2]

[1]Department of Electrical Engineering, Federal University of Campina Grande, Campina Grande 58428-830, Brazil
[2]Department of Electrical Engineering, Federal University of Sergipe, Aracaju 49100-000, Brazil

Corresponding author: Phillipe Cardoso Santos (phillipe.santos@ee.ufcg.edu.br)

This work was supported in part by the Fundação de Amparo à Pesquisa do Estado da Paraíba (FAPESQ), the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) under Grant Finance Code 001.

ABSTRACT The inverse kinematics of mobile manipulators is a challenging problem due to the high degree of kinematic redundancy added by the coupling of the mobile platform and the manipulator. Some different approaches have been proposed to solve this problem, but most of them are either complex in terms of modelling and matrix calculation, with high computational cost and even with singularity problems, or slow in terms of convergence. This paper proposes a new approach for inverse kinematics of mobile manipulators based on the algorithm FABRIK. This new method, named M-FABRIK, has as main advantages the simplicity to implement, a fast convergence and a low computational cost, allowing real-time applications. Furthermore, this solution allows the robot to be positioned according to various criteria, such as decreasing convergence time, avoiding contact with obstacles, avoiding joint angle limits, increasing robot manipulability or even decreasing joint effort, besides avoiding matrix inversion and being robust to singularities. The proposed approach is illustrated by simulation considering a 5 DOF manipulator mounted on an omnidirectional base for a path-tracking task in different environments, including obstacles. A comparison between the proposed approach and classical methods is also presented.

INDEX TERMS FABRIK, inverse kinematics, mobile manipulators, robots.

I. INTRODUCTION

Mobile manipulator (MM) is a term used for robotic systems consisting of a robot arm mounted on a mobile platform. They have been developed aiming to combine the manipulation ability of robotic arms with the mobility of mobile robots [1]. This combination provides the robot with the ability to perform operations in different parts of the same environment, and even reach inhospitable places, without the need to adapt to the environment for such navigation.

Besides the workspace expansion, reconfigurability, disturbance-rejection capabilities and robustness to failure are some of the benefits of having a manipulator mounted on a mobile platform [1].

Most of these benefits accrue from the redundancy added with the mobile base, which can also improve the system in order to be more robust regarding issues like joint angle limitations and joint failures [2].

The redundancy in the system poses many possible joint configurations for a given end-effector task position. The inverse kinematics (IK) problem for mobile manipulators is then finding both the mobile base position and the manipulator joint configuration for a given end-effector position in Cartesian space coordinates.

According to Ram in [3], ample literature is available about the control of mobile manipulators. However, there is only a limited body of literature dealing with IK of MM. Most of the mobile manipulators IK approaches deal with the inversion of the Jacobian matrix [4]–[7]. However, these approaches suffer from high computational cost, complex matrix calculations or singularity problems [8].

In order to avoid the use of the Jacobian matrix and matrix inversions, in 2010 Huang and Tsai presented an inverse kinematics approach based on metaheuristic algorithms [9]. With this approach, Huang and Tsai obtained good results that

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

inspired other authors to use similar approaches, as can be seen in [10] and [3].

Genetic Algorithms (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO), Cuckoo Search (CS) and Biogeography-Based Optimization (BBO), are examples of the metaheuristic algorithms used for mobile manipulators IK [10]. Tests both in simulation and with real robots have been performed with these algorithms and positive results have been obtained. Moreover, environments with obstacles have also been considered, as presented in [3].

Despite the great advantages and importance of the metaheuristic algorithm methods, these approaches are relatively slow in terms of convergence time, as they create a set of individuals and run hundreds of interactions in order to find the solution.

In summary, besides the limited body of literature dealing with IK of MM, the existent approaches are either complex in terms of modelling and matrix calculation with high computational cost and even with singularity problems, or slow in terms of convergence. Therefore the inverse kinematics for mobile manipulators still lacks a fast and simple method.

For fixed-base manipulators, heuristic approaches such as the CCD [11], Triangulation [12] and FABRIK [13], have presented simple and fast solutions avoiding the use of matrix inversions. The main advantage gained with FABRIK, for instance, is a fast convergence besides being simple to implement.

FABRIK is a fast and iterative algorithm, which avoids the use of rotational angles or matrices, and instead finds each joint position via locating a point on a line. Thus, it converges in a few iterations, has a low computational cost and produces visually realistic poses [8]. Moreover, constraints in terms of joints movement or limits are also supported in the algorithm.

This algorithm was originally developed for the computer graphics area, but due to its advantages, FABRIK has also been used in other areas, including robotics. However, this algorithm had never been extended to mobile manipulator robots.

FABRIK works by minimizing the error between the goal point and the end-effector by controlling the joint angles without the need for matrix inversions. This control is performed firstly in the end-effector - base direction (Forward Reaching stage), where adjusting the positioning of the joints normally causes the base to be moved out from its original position. For this reason, a second step is performed, where the base is fixed in its original position and joints are adjusted from the base to the end-effector (Backward Reaching stage). These two steps are repeated iteratively until the end-effector reaches the target.

The FABRIK feature that allows the base to be displaced from its real position is the key point for our proposed approach. Since the robotic arm is mounted on a mobile robot, the second stage of FABRIK can be performed from a different location, as the base is able to move. This point can be defined by means of several important criteria for navigation, such as decreasing convergence time, avoiding collision with obstacles, avoiding joint angle limits, increasing robot manipulability, or even decreasing joint effort. For that purpose, we introduce the idea of an admissible area, which is based upon creating a certain region from where the robot can reach the target and still attain other objectives during the process of navigation. Therefore, we propose an extension for FABRIK to solve the problem of inverse kinematics for mobile manipulators. This approach has as major contributions the following points:

- **Simplicity**: in this approach, both the configuration of the manipulator and the base position can be located based on projections of lines and points in the space, with no need of using rotational angles or matrices.
- **Few iterations**: one of the main advantages of the algorithm FABRIK is the low number of iterations needed to find a solution. In our approach, as the base is mobile, the Backward Reaching stage always starts from a smaller displacement from the position set by the Forward Reaching stage, then, even fewer iterations are needed to find a solution.
- **Suitability for real-time applications**: Apart from the low number of iterations, M-FABRIK has low computational cost, thus, it has a fast convergence time, which allows real-time applications.
- **Ability to deal with additional navigation objectives**: the proposed approach, through the idea of the admissible area, allows the position of the base to be chosen according to different criteria set by the user.

We tested our approach in simulation using the Kuka Youbot model, a 5 DOF (Degrees of Freedom) manipulator mounted on an omnidirectional base, and compared the results with the IK classical approaches of the Pseudo-inverse Jacobian and the Damped-Least Square (DLS). M-FABRIK was also tested in different scenarios and considering environments with obstacles.

This paper is structured as follows. As the new approach is based on the algorithm FABRIK, the original method is introduced in section II. In section III, the proposed approach is presented. Results are presented in section IV and conclusions in section V.

II. FORWARD AND BACKWARD REACHING INVERSE KINEMATICS (FABRIK)

FABRIK is an algorithm which works based on minimizing the error between the target point and the end-effector by controlling the angles of the joints. This control is carried out by adjusting the joints one at a time [13], first in the end-effector - base direction, and then in the base - end-effector direction.

The first step of the algorithm consists of checking whether the target point is reachable or not, by summing all the links length. If so, the algorithm is performed in two steps, as presented in Fig. 1. In this example, p_1, p_2, p_3 and p_4 represent the joints of a planar manipulator, d_1, d_2, d_3 and d_4 represent the links, and the letter t indicates the destination point.

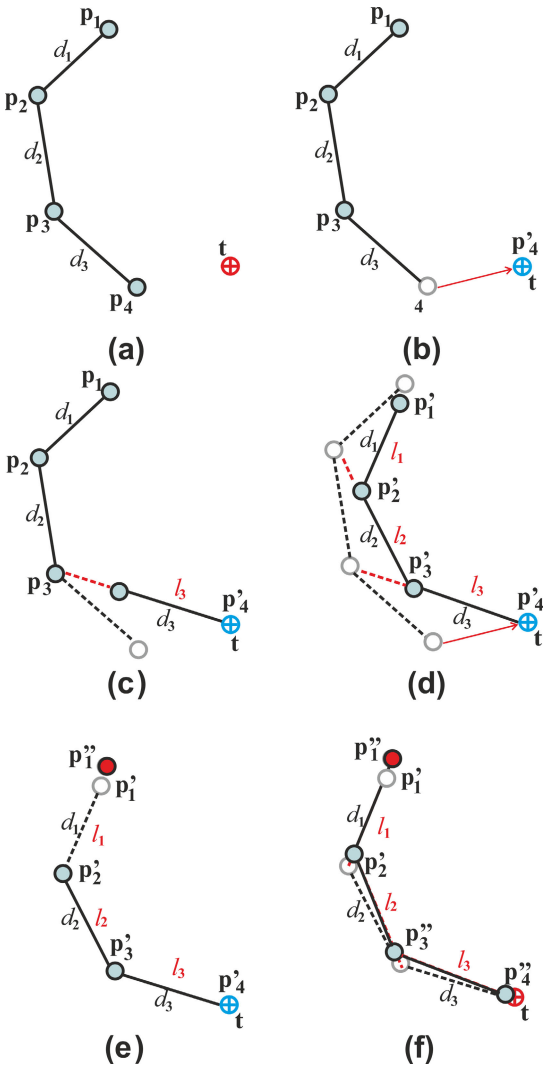


FIGURE 1. An example of a full iteration of FABRIK for a manipulator with 4 joints [13]. (a) The initial position of the manipulator and the target. (b) End-effector p_4 movement to the target. (c) New position of the joint 3 p'_3. (d) Continue the algorithm for the rest of the joints. (e) The second stage of the algorithm: move the base joint p'_1 to its initial position. (f) Repeat the same procedure but this time starting from the base and move outwards to the end-effector.

In the first stage, the algorithm estimates each joint position starting from the end-effector, moving inwards to the manipulator base. Considering the example of Fig. 1, where p_1 is the base and p_4 is the end-effector, the first step of this stage is to assign the end-effector to a new position p'_4, which corresponds to the target point t (Fig. 1b). Then, a line is drawn between p'_4 and the next joint p_3. The new position of joint 3, p'_3, lies on this line, at a distance from p'_4 corresponding to the link d_3 size, (Fig. 1c). Similarly, the new position of joint 2, p'_2, is given on a line drawn between p'_3 and p_2, considering the distance of link 2, d_2. Finally, the process is repeated reaching a new position for the base p'_1 (Fig. 1d).

As shown in Fig. 1d, at the end of the first stage of the algorithm, the manipulator base is out of its initial position. As FABRIK is an approach normally used employing a manipulator with a fixed based, a second stage of the algorithm is necessary for the base to return to its correct position. This second stage is called Backward Reaching and the same procedure is repeated, but this time starting from the base and moving outwards to the end-effector. The first step of this stage is presented in Fig. 1e, with the joint p'_1 returning to the base initial position, being labeled now as p''_1. Then, using the same process as in first stage, a line is drawn from p''_1 to p'_2 and the new position p''_2 is defined based on the link d_1. This process is repeated until it reaches the target.

According to the authors of FABRIK, after a complete iteration, the end-effector will always be closer to the target [8]. Thus, the procedure must be repeated, as many times as necessary, until the end-effector reaches the target.

Although FABRIK was presented with a planar manipulator with 4 degrees of freedom, the algorithm works the same way in a three-dimensional space and for manipulators with more joints. FABRIK was also extended to consider many different types of joint models and their constraints [14].

Other authors have also presented approaches based on FABRIK. In [15], for example, a variation of FABRIK is used to solve the problem of inverse kinematics from an energy transfer perspective. In [16], the algorithm is adjusted to work with joints that have more than two segments. In [17], a data-driven prior to warm start FABRIK is used to produce more natural human poses. In [18], a control system is developed combining the FABRIK and a PID controller for a simple and fast-tracking approach with the end-effector. In [19], a random rotation strategy and local optimum-seeking technology are introduced to improve the FABRIK algorithm in order to avoid obstacles. This approach is combined with a path planning stage based on a bidirectional rapidly exploring random tree (Bi-RRT) algorithm to explore the process postures from a given initial posture to the final grasping posture. More recently, in [20], the RRT algorithm is used to explore the entire workspace and find a feasible path without collision for the end-effector. For each node generated for the end-effector, FABRIK calculates the positions of each joint, and a collision detection method is applied. The node is only added to the tree in case of no collision.

Although FABRIK has been extensively used, all approaches based on this method in the field of robotics are restricted to fixed-base manipulators. In this paper, we introduce an extension of FABRIK in order to take the advantages of this algorithm to the inverse kinematics of mobile manipulators. Thus, in the next section, we present the proposed approach.

III. M-FABRIK: A FABRIK EXTENSION TO MOBILE MANIPULATOR ROBOTS

As presented before, FABRIK is an algorithm performed in two stages, in which at the end of the Forward Reaching phase, the manipulator base is usually out of its proper position. For a fixed-base manipulator, this is corrected by repositioning the base at its original position and executing the algorithm in the inverse order, which is the objective of

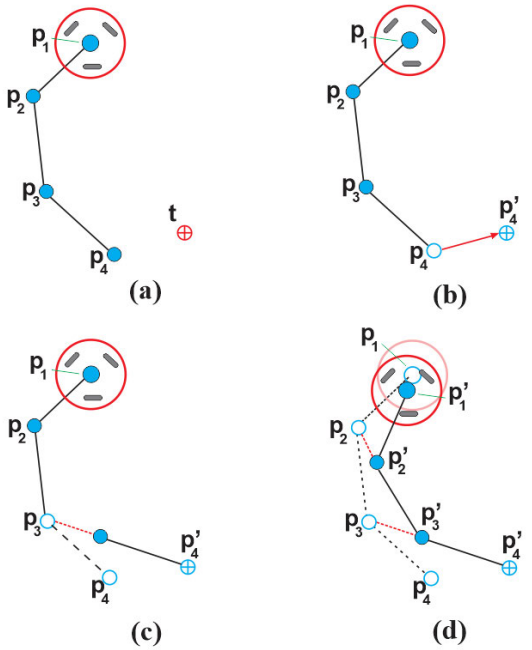


FIGURE 2. An example of a full iteration of M-FABRIK for a planar manipulator mounted on an omnidirectional base. (a) The initial position of the mobile manipulator and the target, (b) Move the end effector p_4 to the target, (c) find the joint p'_3 just as the original algorithm. (d) Continue the algorithm until the base p_1 moves to the new position p'_1, since the base can move, the algorithm finishes and the Backward Reaching is not necessary.

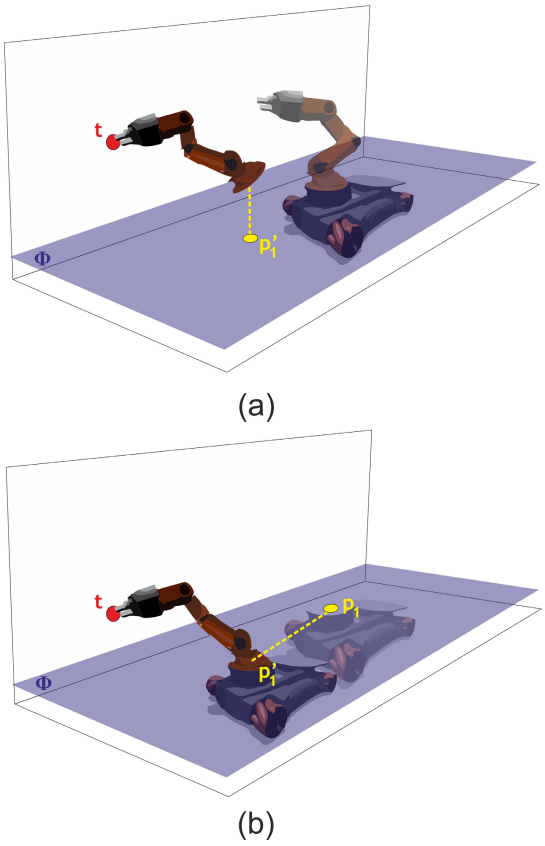


FIGURE 3. M-FABRIK for a manipulator working on a 3D workspace and mounted on a omnidirectional base. (a) Manipulator configuration after finishing the Forward Reaching stage and base projection on the plane Φ where the platform can move. (b) Final configuration with base movement.

the Backward Reaching phase. However, if the manipulator is mounted on a mobile platform, the base can be moved, so that the Backward Reaching phase can be carried out from another point, which can be arbitrarily chosen based on different criteria since it respects the constraints of the robot.

To a better understanding of the proposed methodology, we first present the approach considering a planar manipulator mounted on an omnidirectional base. Thus, both the manipulator and the mobile base work in the same plane, as presented in Fig. 2.

The robot used in this example is similar to the one used in [3] and it is also similar to the manipulator used by Aristidou in [8] to present the methodology of FABRIK. So, p_1, p_2 and p_3 are the joints of a planar manipulator, p_4 is the end-effector and t is the target. However, we consider now p_1 is a joint attached to the center of a mobile base.

In the first stage, the algorithm works exactly as the original FABRIK, as presented in Fig.2a, 2b and 2c. At the end of the Forward Reaching stage, Fig. 2d, the base is out of its original position. Nevertheless, as the base now is mounted on an omnidirectional robot, the base can be easily moved into this new position. For such movement, the robot will have achieved the target point and the base will be at a possible position. Therefore, the Backward Reaching phase is not even necessary in this case, and the final configuration is reached in less than one complete iteration.

Considering now a more realistic situation, we present our approach with a manipulator in a 3D workspace mounted on an omnidirectional platform. The proposed method is illustrated in pseudo-code in Algorithm 1 and in some graphical representations along this section.

For a robot working in the 3D workspace, the movement of the arm in the Forward Reaching stage can move the base not only out of its initial position, but also out of the plane where the omnidirectional platform can be moved. Thus, the Backward Reaching stage is required. However, the first step of this stage does not necessarily need to move the base to its initial position. In this case, there is an entire plane with possible positions where the mobile robot can be moved to, as presented in Fig. 3a.

The question, now, is how to choose the best point among the available possibilities. Some different criteria can be used to set the base position. Aiming to decrease the number of iterations and consequently, the convergence time, we propose choosing the closest point, by projecting the base in the plane Φ, as presented in Fig. 3. This projection is performed before the first step of each Backward Reaching stage as illustrated in the Algorithm 1 by the function *ProjectPoint()*. Thus, the base always starts with a smaller displacement from its previous position. Consequently, the algorithm has a faster convergence, as it needs fewer iterations to reach the final configuration.

Nevertheless, the main advantage of the proposed approach is the possibility to allow the choice of the base position, in order to accomplish additional requirements besides reaching the target, such as avoiding obstacles, respecting joint limits, increasing manipulability or decreasing joint effort. In this paper, we focused on the obstacle avoidance and joint limits, but the same idea can be applied for other different objectives.

For that purpose, we introduce the concept of admissible area, which consists of the region where the base can be positioned to assure the target reachability and the accomplishment of the additional requirements. This area is created by setting the region from where the robot can reach the target and subtracting it by regions where the base would violate the additional objectives.

As an example, in Fig. 4 we present a situation in an environment where the robot has to reach a target keeping a safe distance to the wall. In this figure, the blue area represents the whole region where the robot can navigate, the red area is the danger zone, which the robot cannot enter lest it make a collision, and the green one is the admissible area.

In algorithm 1, this stage is performed by the function *CreateAdmissibleArea()* before the recursive loop used to find the solution. The creation of the admissible area allows inspecting whether the target is reachable or not, since the nonexistence of an admissible area equates the nonexistence of a position from where the robot can reach the target, leading the algorithm to stop. Furthermore, if an admissible area exists, this methodology also allows verifying whether it is possible to navigate from the actual position to the desired one by checking the connectivity between the navigable and the admissible areas.

In the example presented in Fig. 4, the Forward Reaching stage moves the base to a position over the plane where the platform can move. Based on the criterion of decreasing the convergence time, the base would be projected in the plane to the position $\hat{p}_1$. However, this position is not allowed, since the robot would be positioned close to the obstacle, increasing the chances of a collision. Thus, the new position for $\hat{p}_1$ is chosen translating it to the closest point of the admissible area, as presented in Fig. 4a. This movement allows positioning the base in a safe distance from the obstacle, Fig. 4b. The Backward Reaching phase is then started from this new position p'_1 and the algorithm is iteratively repeated until a possible configuration is found. The steps to choose the new base position according to the admissible area are presented in the Algorithm 1 from line 11 to line 17.

In [8], Aristidou et. al. states that the unconstrained version of FABRIK converges to any given chains/goal positions when the target is reachable. He also mentions that neither the joint limits nor the joint movement constraints have reduced the ability of FABRIK to converge to a solution. Therefore, since M-FABRIK works with the same methodology as the original FABRIK and the admissible area is mainly created based on the reachability of the target, we can state that: if there exists an area, connected to the navigable region,

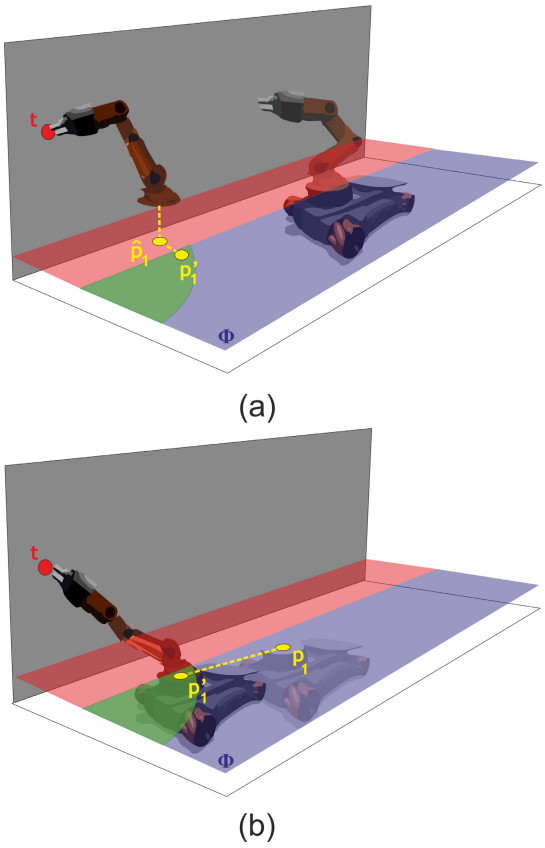


FIGURE 4. M-FABRIK considering the admissible area. (a) After the F-R stage, the base is projected to point $\hat{p}_1$, but this point is located in the rejection area, so it is translated to the admissible area. (b) Final configuration with base movement.

from where the robot can reach the target and accomplish the secondary objectives, M-FABRIK is able to place the base at this area and find a solution just as the original FABRIK.

Nevertheless, M-FABRIK is essentially an inverse kinematics approach rather than a motion planning method. Thus, for applications such as target tracking or path following, depending on the admissible areas created during the navigation, two successive configurations can be generated far from each other and a collision can happen while moving from one to the other, as presented in Fig. 5a.

Aiming to avoid problems with distant successive configurations for applications of target tracking or path following, we added an extra stage on the method, which is presented in the Algorithm 2. This stage limits the movement of the base in a maximum step size ρ, which can be defined by the user, and creates intermediary configurations at these points. Additionally, these configurations are set considering the admissible area, meaning additional requirements can be assured.

As an example, in Fig. 5a the base would move from the point p_1 to the position p'_1. However, the movement is limited by ρ and the first movement of the base is to the point b_1. This step is performed by the function *MoveOneStep()* on Algorithm 2. From b_1, an intermediary configuration q_i is set by running M-FABRIK algorithm and a second step is

Algorithm 1: A Full Iteration of the M-FABRIK Algorithm

Input: The joint positions p_i for $i = 1, \ldots, n$; (with p_n being the end-effector and p_1 the base); the base orientation θ; the target position t; the rejection area A_{rej};distance between each joint $d_i = |p_{i+1} - p_i|$ for $i = 1, .., n$.

Output: The new joint positions p_i for $i = 2, \ldots, n$, the base position p_1 and orientation θ.

```
//Create the admissible area;
A_adm = CreateAdmissibleArea(t,A_rej);
//Check whether there is an admissible area to position the base;
if (A_adm == null) then
    Break;
end
// Check whether the distance between the end effector p_n and the target t is greater than a tolerance.;
dif_A = |p_n − t|;
while (dif_A > tol) do
    // STAGE: FORWARD REACHING;
    p_i = ForwardReaching(p_i,t);
    // Project p_1 on the base actuation plane Φ;
    p̂_1 = ProjectPoint(p'_1, Φ);
    // Check whether p̂_1 is in the admissible area;
    if (p'_1 ⊄ A_adm) then
        p'_1 = MoveToAdmArea(p̂_1,A_adm);
    else
        p'_1 = p̂_1 ;
    end
    // STAGE: BACKWARD REACHING;
    p_i = BackwardReaching(p'_i,t);
    θ = SetBaseOrientation(p_i, θ);
    dif_A = |p_n − t|;
end
```

given towards p'_1. This step leads the robot to the point $\hat{b}_2$, which is located in the rejection area. Thus, $\hat{b}_2$ must be translated to the admissible area. However, besides being in the admissible area, the position of b_2 must be inside of the area where the robot can move one step in a straight line from its current position. This movement is performed on the function *MoveToStepArea()*. A straight line is then traced from b_2 to the point p'_1 and limited again by ρ, creating the point $\hat{b}_3$. As this point is also in the rejection area, the third step of the robot is the point b_3, which is located in the admissible area. From this point, a new position $\hat{b}_4$ can be found in the admissible area. Nevertheless, $\hat{b}_4$ is not in a position where the robot can step to without crossing the rejection area. So, $\hat{b}_4$ is translated to the position b_4, which is the closest point in the region where a straight line step can be performed from b_3. This area is highlighted in figure 5a in light green. From the point b_4, the base can reach the desired position of the base p'_1 in just with one more step in b_5, without the need to cross the rejection area. So, the additional stage is able to create a set of intermediary configurations leading robot from the point p_1 to p'_1, as presented in Fig. 5b.

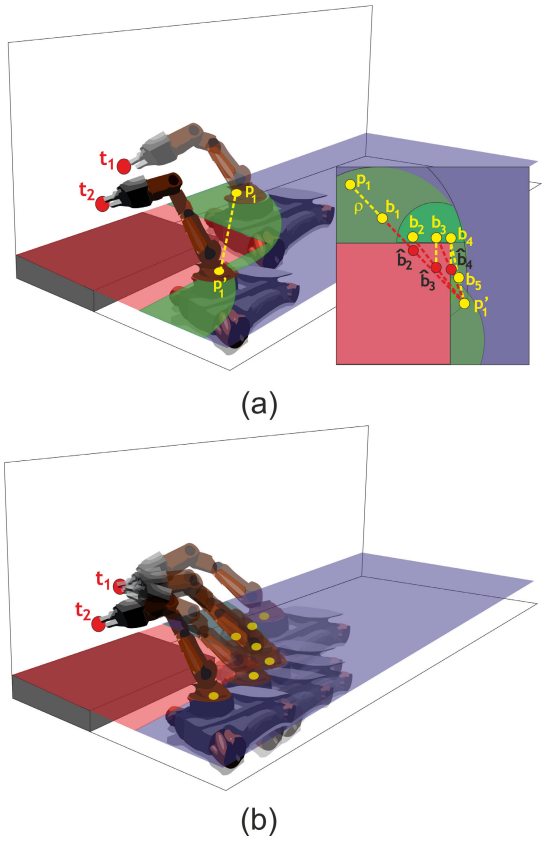


FIGURE 5. Creating intermediary configurations between two configurations far from each other. (a) b_1, b_2, b_3, b_4 and b_5 are the accepted positions for the intermediary configurations, $\hat{b}_2$, $\hat{b}_3$ and $\hat{b}_4$ are rejected since they were not in the area which the robot can step to (in light green is presented the step area for the point b_3). (b) Set of intermediary configurations leading robot from the point p_1 to p'_1.

The admissible areas created for these experiments are just examples. The idea is to allow the users to set their own areas according to the environment, the task objective, or even to use a different criterion. For example, the rejection areas can be either broader or narrower depending on the type of obstacle or the level of safety the user needs for the task. The same approach can also be used in an environment free from obstacles, but with the objective to keep manipulability over a certain threshold. Thus, the admissible area would be the region where the robot can reach the target, keeping the manipulability at the desired level.

It is important to state that the admissible area allows the obstacle avoidance only for the base since further study needs to be conducted in order to apply this approach in joint positions due to its movement constraints. However, other obstacle avoidance approaches can be combined with M-FABRIK in order to avoid collision also for the manipulator. The random rotation strategy and local optimum-seeking technology proposed in [19] or the method based on RRT and FABRIK proposed in [20] are examples of possible combinations.

TABLE 1. DH table for the Kuka Youbot omnidiretional robot.

Joint	θ (rad)	d (mm)	a (mm)	α (rad)
1	θ_1	147	33	$\pi/2$
2	θ_2	0	155	0
3	θ_3	0	135	0
4	θ_4	0	0	$\pi/2$
5	θ_5	217.5	0	0

Another important feature to mention is that the original method of FABRIK solves the IK problem in position space, not considering the orientation. For manipulator robots, in applications which a specific actuation angle for the end-effector is required, having the end-effector mounted on a ball and socket joint allows the method to set both position and orientation. If the robot does not have this kind of joint, fixing the joint connected to the end-effector according to the desired orientation is a possible solution. However, this constraint implies in the loss of a degree of freedom in the manipulator. For the original algorithm, depending on the chain length and others constraints applied to the robot, FABRIK can fail in finding a solution. M-FABRIK is more robust in dealing with this situation however, since this loss can normally be compensated by the redundancy added by the mobile base.

The same constraint imposed when setting the end-effector orientation is also applied for the base. Thus, after performing the Backward Reaching, we let the user choose the base orientation according to the wish on how to perform the task by creating the function *SetBaseOrientarion()* in the Algorithm 1. For example, the base orientation can be defined in order to the robot always heads to the target, to avoid the first joint to reach its limit or even to be fixed during the experiment.

The proposed approach was tested by simulation and the results are presented in the next section.

Algorithm 2: Algorithm to Generate Intermediary Configurations

Input: The previous and the new configurations p_i and p'_i; the target position t; the rejection area A_{rej}; the step size ρ.
Output: The intermediary configurations b_j.

```
// Check whether the distance between the base of the previous and the current configuration is greater than the step;
dif_B = |p'_1 − p_1|;
if (dif_B > ρ) then
    // b_0 receives the base position of the starting configuration;
    b_0 = p_1;
    while (dif_B > ρ) do
        // Move one step from b_{j−1} in the direction of p'_1;
        b̂_j = MoveOneStep(b_{j−1},p'_1) ;
        // Check whether b̂_j is in the admissible area;
        if (b̂_j ⊄ A_adm) then
            b_j = MoveTStepArea(b̂_j,A_adm);
        else
            b_j = b̂_j;
        end
        // Translate the configuration p_i to the base position b_j;
        q_i = TranslateConfig(p_i,b_j);
        // Run M-FABRIK algorithm to find the new intermediary position of the base q_i ;
        q_i = M-FABRIK(q_i,target);
        j++;
        dif_B = |b_j − p'_1|;
    end
end
```

IV. RESULTS

This section aims to show the proposed approach feasibility and advantages by simulating a mobile manipulator performing tasks in different situations and environments and also comparing it with classical approaches. Simulations were performed in MATLAB considering the model of the mobile manipulator Kuka YouBot presented in Fig. 2. This robot has an omnidirectional base and a manipulator with 5 joints, and it has been extensively used in recent mobile manipulators works, as presented in [21] and [10]. The Denavit-Hartenberg (DH) parameters for the arm is presented in the table 1.

For the first experiment, we performed a comparitive study between the proposed approach and two classical IK methods, the Pseudo-inverse Jacobian and the Damped-Least Square (DLS), with parameters set based on the values used in [8]. These classical methods were applied using the methodology presented in [10], where the mobile manipulator is transformed into a fixed manipulator by modelling the base as virtual joints. One rotational and two prismatic joints were used to model the omnidirectional base.

Firstly, we compared the performance of the algorithms based on a set of one thousand random target points, with the mobile manipulator starting from different initial configurations and with a requested tolerance distance varying from $1mm$ to $10^{-3}mm$ to describe when the end-effector reaches the target.

Table 2 presents the average time the algorithms needed to find a solution. For accuracy of 1 *mm*, M-FABRIK needs on average only 0.33 *ms* to find a solution, compared to the 33.12 *ms* the Pseudo-inverse Jacobian and the 30.72 *ms* the DLS take. Apart from being faster, the performance of M-FABRIK is less susceptible to requirements of a higher degree of accuracy, with a difference of only 0.39 *ms* from the time needed to find a solution with 1 *mm* to $10^{-3}mm$ of accuracy. On the other hand, the classical methods varied more than 40 *ms*. The execution was measured on MATLAB with an Intel Core i5 1.8 GHz processor with 6GB RAM. No optimizations were used for the methods reported in the table.

TABLE 2. Average execution time over 1000 runs for random targets and initial configurations.

Average execution time (seconds)			
Accuracy (mm)	Pseudo-inverse Jac.	DLS	**M-FABRIK**
1	0.03312	0.03072	**0.00033**
0.1	0.05084	0.04423	**0.00041**
0.01	0.05781	0.05465	**0.00066**
0.001	0.07709	0.07096	**0.00072**

TABLE 3. Average number of iterations over 1000 runs for random targets and initial configurations.

Average number of iterations			
Accuracy (mm)	Pseudo-inverse Jac.	DLS	**M-FABRIK**
1	116.51	114.90	**2.92**
0.1	160.49	158.73	**4.24**
0.01	205.06	203.03	**5.58**
0.001	249.09	247.65	**6.89**

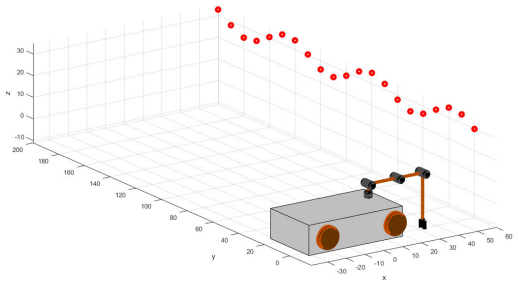


FIGURE 6. Initial configuration of the robot and target setpoints represented by the red dots.

These results are reflected from the number of iterations, presented in table 3. M-FABRIK varies from an average of 2.92 to 6.89 iterations to the requested accuracy of 1 mm and $10^{-3} mm$ respectively. The pseudo-inverse Jacobian varies from 116.51 to 249.09 and the DLS from 114.90 to 247.65.

We also compared the approaches performing an experiment similar to the one presented in [10], in which a sinusoidal path composed of 21 points was defined for the end-effector to follow.

The robot initial configuration as well as the target setpoints are presented in Fig. 6. We defined a tolerance distance of 1 mm to describe when the end-effector reaches the target.

For this experiment, joint limits were not considered and we chose for the M-FABRIK to rotate the base according to the angle of the joint attached to the platform, so that the robot would follow the path always heading to the setpoints. This is not a requirement, but just an option given to the user to choose the base behavior during the experiment. The algorithm would perfectly work with the platform moving only on the $x - y$ axis with no rotation. The result presented in Fig. 7 confirms the algorithm functionality, as the robot follows the entire path correctly.

The same experiment was performed using the classical approaches of the Pseudo-inverse Jacobian and the DLS. In both cases, the robot followed the path correctly.

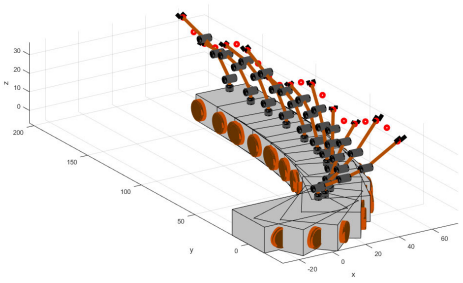


FIGURE 7. Experiment 1: Path-tracking result for an environment free from obstacles using M-FABRIK.

TABLE 4. Comparison of execution time between the proposed approach and classical methods.

	Matlab execution time (seconds)		
Setpoint	Pseudo-inverse Jac.	DLS	**M-FABRIK**
1	0.0295561	0.0357971	**0.0007977**
2	0.0221152	0.0273181	**0.0002571**
3	0.0227257	0.0282108	**0.0004001**
4	0.0220802	0.0268064	**0.0005059**
5	0.0220627	0.0309427	**0.0003863**
6	0.0214217	0.0257867	**0.0005877**
7	0.0184710	0.0242942	**0.0006092**
8	0.0246785	0.0248332	**0.0007263**
9	0.0298927	0.0213003	**0.0005861**
10	0.0356442	0.0189399	**0.0006735**
11	0.0296823	0.0204147	**0.0004212**
12	0.0212183	0.0190808	**0.0004933**
13	0.0190868	0.0173560	**0.0005720**
14	0.0307200	0.0213659	**0.0003807**
15	0.0190632	0.0219592	**0.0006049**
16	0.0167249	0.0131456	**0.0006811**
17	0.0173864	0.0205337	**0.0004263**
18	0.0225367	0.0235163	**0.0003797**
19	0.0136815	0.0132190	**0.0004885**
20	0.0173445	0.0210712	**0.0003693**
21	0.0214173	0.0205138	**0.0003699**
Total time	0.4775099	0.4764056	**0.0107168**

However, as expected, both approaches require a higher execution time and a higher number of iterations when compared with the proposed approach, as presented in tables 4 and 5 respectively.

The results presented in table 4 show M-FABRIK needed less than 1 ms to find a solution for each of the setpoints of the path, summing up a total of around 10 ms to find all the 21 configurations for the robot to follow the path. On the other hand, for each setpoint, the classical methods needed between 13 ms and 35 ms, performing the complete path in around 470 ms, almost 0.5s. Thus, M-FABRIK performed around 47 times faster than the classical approaches. These results can also be qualitatively compared with the following meta-heuristic IK approaches presented by Lopez et. al. in [10]: Differential Evolution (DE), Particle Swarm Optimization (PSO), Hybrid Biogeography-Based Optimization (HBBO), Cuckoo Search (CS) and Teaching–Learning-Based Optimization (TLBO). However, these approaches require a high computational time and they are more likely to be applied for off-line applications [10]. For the same following track

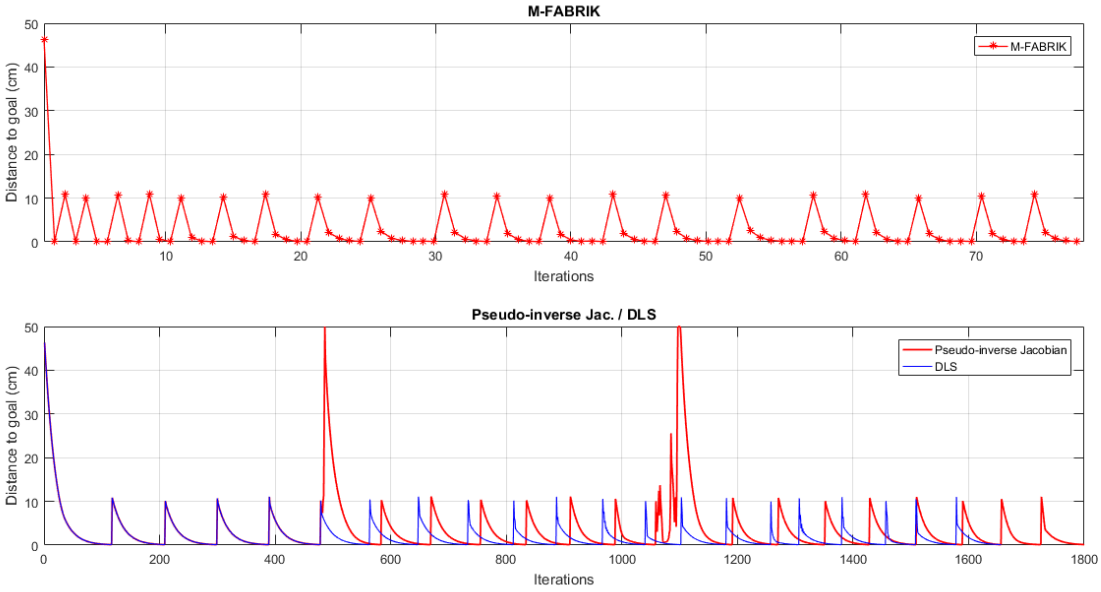


FIGURE 8. Number of iterations needed to reach the target setpoints during the path tracking experiment.

TABLE 5. Comparison of number of iterations between the proposed approach and classical methods.

	Number of iterations		
Setpoint	Pseudo-inverse Jac.	DLS	**M-FABRIK**
1	116	116	**1**
2	91	91	**1**
3	89	89	**2**
4	89	89	**2**
5	88	88	**2**
6	104	84	**3**
7	85	83	**3**
8	85	85	**4**
9	78	78	**4**
10	75	73	**6**
11	77	79	**4**
12	69	73	**4**
13	132	61	**5**
14	78	77	**4**
15	80	76	**6**
16	76	48	**6**
17	80	73	**4**
18	79	75	**4**
19	66	51	**5**
20	68	69	**4**
21	77	77	**4**
Total	1782	1635	**78**

experiment, Lopez mentioned that the DE and PSO algorithms performed below 30 *s* and the others above 50 *s*. These values are much higher than the ones obtained with M-FABRIK and the classical approaches. Therefore, for the rest of the paper we focused the comparison between M-FABRIK and the classical approaches.

Regarding the number of iterations, the results presented in table 5 show that M-FABRIK is able to perform the inverse kinematics with a few iterations. The proposed approach needed a maximum of 6 iterations to find the arm

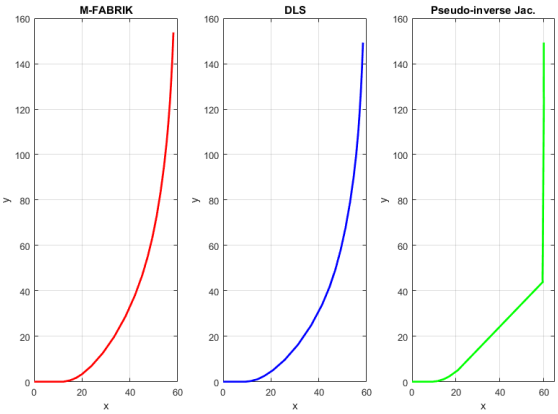


FIGURE 9. Base movement in *xy* plane during the path tracking experiment.

configuration and the base position. In other words, the Forward and Backward Reaching stages have been performed a maximum of 6 times per setpoint. For some targets, they were even performed only once. The total amount of iterations needed to find all the 21 configurations by M-FABRIK was only 78. This number is even lower than a great part of the results obtained with the classical approaches for a single target. The Pseudo-inverse Jacobian approach performed the tasks with the number of iterations varying from 68 to 132 iterations per setpoint and a total of 1782. The DLS varied between 51 and 116 and a total of 1635.

The graphs display the distance from the end-effector to the target for each iteration during the experiments are present in Fig.8.

On these graphs, the peaks represent the moment when the end-effector reaches a target and a new setpoint is chosen. For

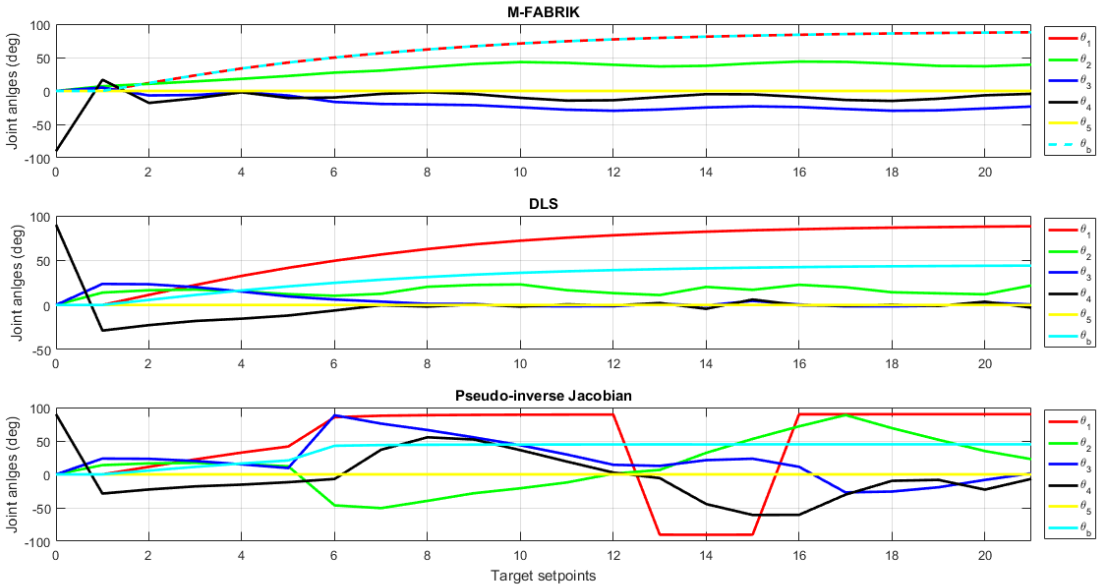


FIGURE 10. Joints movement during the path tracking experiment.

the M-FABRIK, the graph shows a fast convergence, with the distance decreasing close to zero in just a few iterations. For the classical approaches, the convergence process is slower and requires a higher number of iterations, being important to highlight the scale difference on the x axis between the two graphs. Furthermore, for the Pseudo-inverse approach, some high peaks occur when the method computes the inverse kinematics near to singular configurations. Therefore, the joint angles place the end-effector further from the target.

A comparison of the robot movement during the experiment is also presented in 9 and 10. Fig.9 shows that the base movements on the $X - Y$ plane were similar for all 3 approaches. In Fig.10 the movement of all joints and also the angle of the base are presented. For both the M-FABRIK and the DLS, the robot performed smoothly during the experiment. Nevertheless, for the Pseudo-inverse Jacobian approach, when the method computes the inverse kinematics near to a singular configuration, some abrupt movements are performed by the joints.

In summary, the proposed approach presented a smooth result with fewer iterations and lower computational cost when compared to the classical approaches, while also being simple to implement and not susceptible to matrix singularities.

Furthermore, M-FABRIK allows the robot to accomplish additional requirements besides reaching the target. These additional requirements are achieved by creating admissible areas in which the base can be projected in order to accomplish the chosen criterion. Aiming to illustrate this application, we simulated different scenarios in which the mobile manipulator had to navigate with a predefined path for the end-effector in environments with obstacles. For this task, besides moving the end-effector throughout the path,

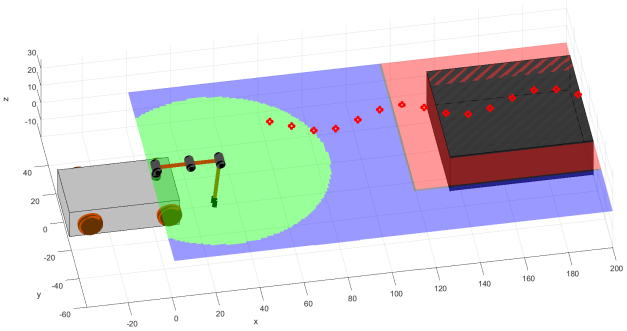


FIGURE 11. Initial configuration for an experiment considering a sinusoidal path passing over a squared obstacle (black). For this experiment we consider the navigable area (blue), admissible area (green) and rejection area (red).

the robot had to keep a safe distance to the obstacles to avoid a collision. From now on, all experiments were performed considering the Kuka Youbot joint limits.

The first scenario is presented in Fig. 11. For this experiment, the targets for the end-effector are disposed on a sinusoidal path passing over a squared obstacle and the rejection area was chosen to keep the robot at least 20 *cm* away from the obstacle (red area in Fig. 11). The blue area in the figure corresponds to the whole area where the robot can navigate and the green one to the admissible area created considering the initial position of the end-effector.

Fig. 12 presents the robot configurations during the experiment, demonstrating that M-FABRIK was able to generate suitable configurations to achieve both objectives of reaching the target and keeping the robot away from the obstacle.

Based on the admissible areas created along with the experiment, when the robot approached the obstacle, two

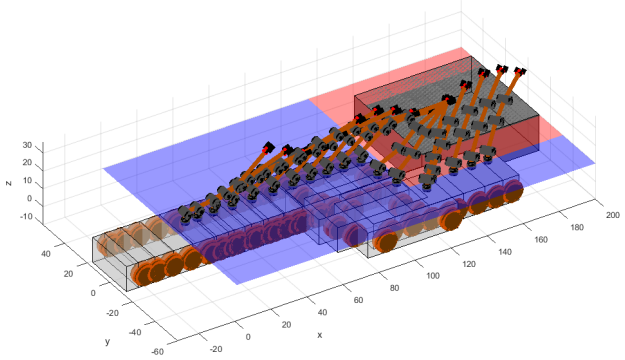


FIGURE 12. Experiment 2: Path-tracking result for an environment with an obstacle.

configurations distant to each other were generated in order to assure the base would not be positioned on the rejection area, Fig. 13a and Fig. 13b. However, besides being far from each other, the movement between these two configurations can lead the robot to the rejection area or even to a collision. Therefore, algorithm 2 is applied in order to create intermediary configurations limiting the step of movement and assuring that the robot will not pass through the rejection area. Fig. 13c presents the intermediary configurations generated in the experiment.

The number of iterations as well as the convergence time for this experiment are presented in table 6. Although the position of the base is now influenced by the admissible area before each stage of Backward reaching, the average number of iterations needed to find a solution was almost the same when comparing to the results presented in table 5, with a maximum of only 7 iterations to find a final configuration and a total of 71 iterations for the entire path tracking. Nevertheless, due to the extra part on the algorithm to seek a safer position to the base, the convergence time increased, but with a maximum value of only 9.7 *ms* per target and a total of 65.5 *ms* for the entire path. These values are even lower than the results performed by the classical approaches in an environment free from obstacles and without joint limit constraints.

For experiment 3, we simulated a situation in which the mobile manipulator had to perform a task such as drawing, painting or drilling, on the wall with a predefined path, and with the restriction that the robot had to act in parallel to the target during the entire path. In addition, the robot had to keep a safe distance of 20 *cm* to the wall and to other obstacles in the environment.

The initial conditions are presented in Fig. 14. Based on the safety requirements, just a small area with a narrow passage towards the end of the environment was available for the robot to perform the task on.

Even with a limited navigation area available, M-FABRIK was able to generate all the configurations and complete the task successfully, as presented in Fig. 15.

Table 6 presents the convergence time and the number of iterations performed in this experiment. In most parts of

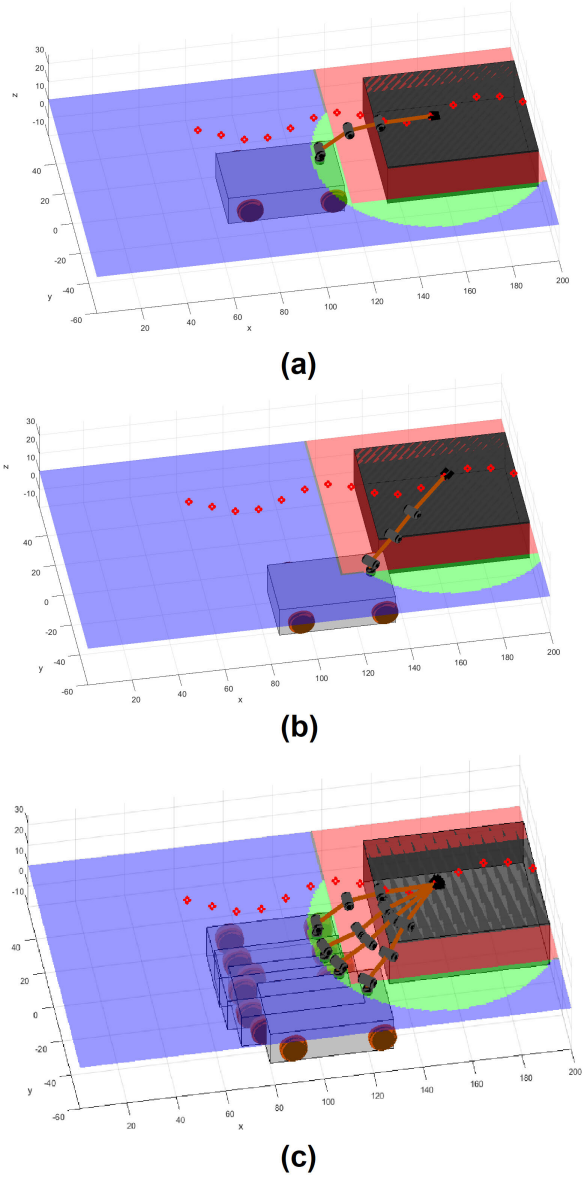


FIGURE 13. Experiment 2: Intermediary configurations for the obstacle avoidance.

the navigation, the configurations were found with less than 6 iterations and convergence time between 2 *ms* and 7 *ms*. The exception was the configuration for the 11^{th} setpoint, which is the first target within the narrowest region. This configuration required 11 iterations in 24 *ms* approximately. However, this is still faster than the most configurations generated by the classical approaches in experiment 1, in which no restriction were imposed.

For the final experiment, the information about the obstacles and the entire path was available in advance. We, however, used a more challenging environment with two obstacles, presented in Fig. 16, one of them being L-shaped and positioned there to entice the robot to move for a position from which it would not be able to continue the task.

TABLE 6. Execution time and number of iterations for the performed experiments using M-FABRIK.

Experiment 2			Experiment 3		
Sp	Conv. time (sec)	Number of iterations	Sp	Conv. time (sec)	Number of iterations
1	0.0029670	1	1	0.025373	4
2	0.0014995	2	2	0.023005	4
3	0.0016795	2	3	0.003983	4
4	0.0020058	3	4	0.004365	3
5	0.0020247	3	5	0.007372	5
6	0.0026943	4	6	0.005197	4
7	0.0040303	6	7	0.002599	3
8	0.0042602	6	8	0.003071	3
9	0.0027630	4	9	0.002482	3
10	0.0030643	5	10	0.003433	2
11	0.0027225	5	11	0.024493	11
11	0.0097924	5	12	0.010996	6
11	0.0013354	4	13	0.002227	3
11	0.0041886	5	14	0.001993	3
12	0.0019221	2	15	0.002647	3
13	0.0040123	3	16	0.002911	3
14	0.0090136	7	17	0.004590	4
15	0.0055318	4	18	0.006649	3
-	-	-	19	0.004230	4
-	-	-	20	0.002887	3
Total	0.0655073	71	Total	0.144502	78

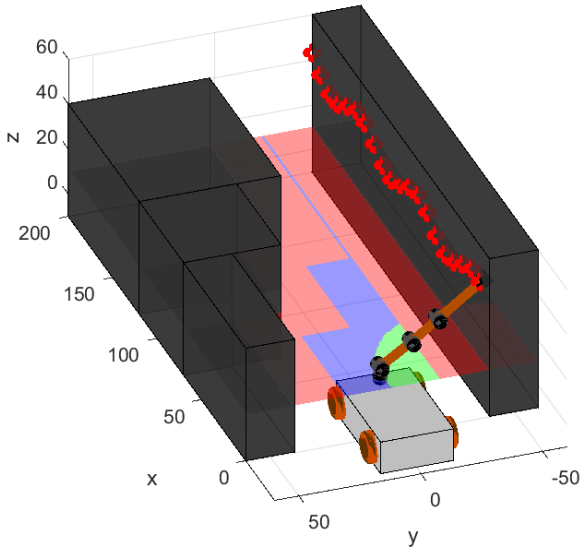


FIGURE 14. Experiment 3: Initial configuration with a predefined path on the wall. For this experiment the navigation area gets narrower towards the end of the environment due to the obstacles position.

As mentioned before, the admissible area can be created based on many different criteria, according to the objectives of the navigation. Another possibility is to create the admissible area for the entire path before the start of navigation, as presented by the green area in Fig. 16. This admissible area created in advance allows the user to detect areas with no connections to the following configurations, just by checking the intersection between them. Thus, these areas can be excluded from the admissible area and only the ones that allow the robot to successfully move through the entire path are accepted.

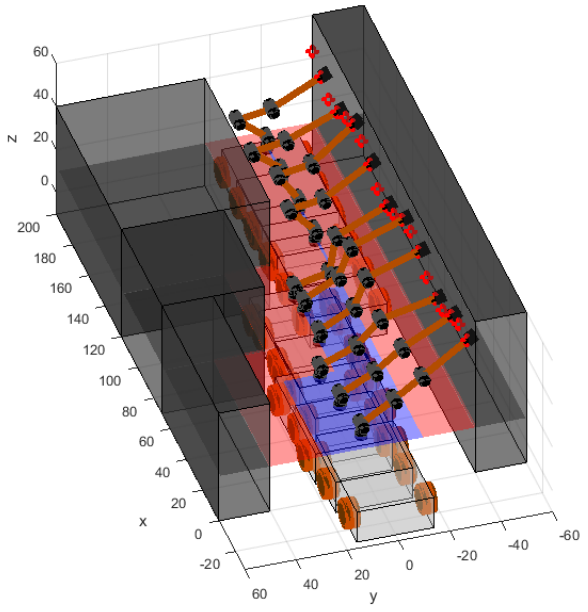


FIGURE 15. Experiment 3: Path-tracking result with obstacle avoidance.

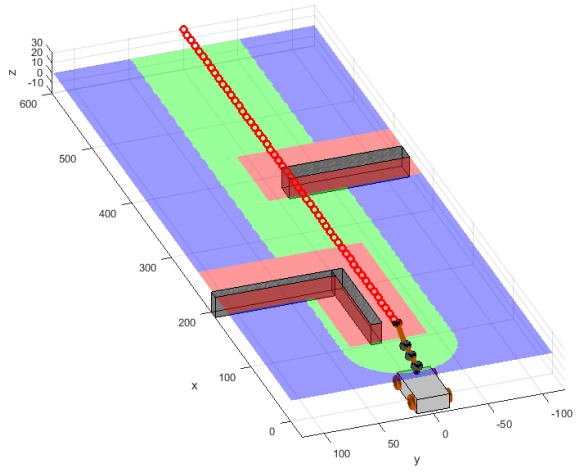


FIGURE 16. Experiment 4: Initial configuration for a more challenging environment with two obstacles. For this experiment the admissible area (in green) is created in advance.

Based on this idea, the green area in the inner part of the first obstacle was excluded from the admissible area and M-FABRIK generated all the configurations for the robot to follow the path and avoid both obstacles successfully, as presented in Fig. 17. For this experiment, the performance in terms of the number of iterations and convergence time is not presented in detail in a table, since a greater number of set-points were used. Nevertheless, the average convergence time per configuration was 9 *ms* and the average number of iteration was 4.

Therefore, the proposed algorithm was able to solve the inverse kinematics problem for a mobile manipulator with few iterations and low computational time, allowing for real-time applications. The algorithm has also been

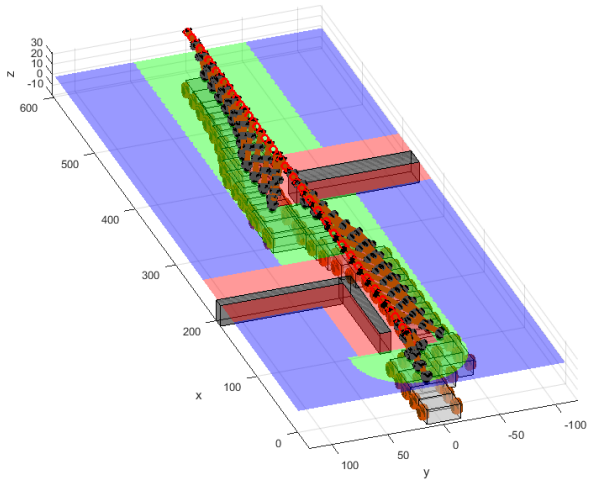


FIGURE 17. Experiment 4: Path-tracking result for an environment with two obstacles.

successfully applied, in environments with obstacles and narrow corridors. Furthermore, the M-FABRIK gives more flexibility for the user to choose additional objectives for the navigation, it is simple to implement and it avoids singularities.

V. CONCLUSION

This paper presented a new approach for inverse kinematics of mobile manipulators robots based on FABRIK. This approach keeps all main features and consequently the advantages of the algorithm. Thus, we propose a simple and fast algorithm with low computational cost, which allows for real-time applications.

Considering the base movement allowed by the mobile robot which the manipulator is attached to, we extended FABRIK by proposing a method to choose the new base position after each Forward Reaching stage, so that the Backward Reaching can be carried out from another point since it respects the constraints of the robot.

The method allows for the new position to be arbitrarily chosen based on different criteria, in order to accomplish additional requirements besides reaching the target in few iterations, such as to avoid obstacles, respect joint limits, increase manipulability or decrease joint effort. For these additional requirements, we proposed the creation of specific areas which the base can be positioned or not during the task.

The proposed approach was illustrated by simulation considering a 5 DOF manipulator mounted on an omnidirectional base. Firstly, the approach was tested with a set of one thousand random target points, being able to find a solution to all of them with few iterations and with a low convergence time. In addition, the method was robust to variations on the accuracy requirement.

M-FABRIK was also tested for a path-tracking task in different environments, including obstacles. For all experiments performed, the algorithm returned configurations, which allowed the mobile manipulator to track the path successfully and smoothly, with a few iterations and with a low convergence time. These results show the feasibility of M-FABRIK to real-time applications. M-FABRIK also presented better performance when compared to the classical approaches of the Pseudo-inverse Jacobian and the DLS.

Furthermore, the creation of the admissible area allowed the robot to navigate with no collision and within the safety region even in really narrow environments. The admissible area is an important feature of the proposed approach since it can be created based on other criteria, according to different navigation requirements the user may be interested in.

M-FABRIK also deals better with joint constraints and predefined end-effector orientation when compared to the classical FABRIK, due to redundancy added by the mobile base.

Future works may investigate the possibility to extend the approach of creating admissible and rejection areas to choose the joints and links position during the execution of M-FABRIK. So, an obstacle avoidance approach can be also applied for the manipulator.

REFERENCES

[1] B. B. V. L. Deepak and D. R. Parhi, "Control of an automated mobile manipulator using artificial immune system," *J. Exp. Theor. Artif. Intell.*, vol. 28, nos. 1–2, pp. 417–439, Mar. 2016.

[2] D. M. Bodily, T. F. Allen, and M. D. Killpack, "Motion planning for mobile robots using inverse kinematics branching," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 5043–5050.

[3] R. V. Ram, P. M. Pathak, and S. J. Junco, "Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling," *Mechanism Mach. Theory*, vol. 131, pp. 385–405, Jan. 2019.

[4] K. Tchoá and J. Jakubiak, "Endogenous configuration space approach to mobile manipulators: A derivation and performance assessment of jacobian inverse kinematics algorithms," *Int. J. Control*, vol. 76, no. 14, pp. 1387–1419, Jan. 2003.

[5] K. Tchón, "Dynamic Jacobian inverses of mobile manipulator kinematics," in *Advances in Robot Kinematics: Motion in Man and Machine*. Dordrecht, The Netherlands: Springer, 2010, pp. 11–21.

[6] M. Galicki, "Inverse kinematics solution to mobile manipulators," *Int. J. Robot. Res.*, vol. 22, no. 12, pp. 1041–1064, Dec. 2003.

[7] W. Li and R. Xiong, "Dynamical obstacle avoidance of task-constrained mobile manipulation using model predictive control," *IEEE Access*, vol. 7, pp. 88301–88311, 2019.

[8] 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.

[9] H.-C. Huang and C.-C. Tsai, "Particle swarm optimization algorithm for optimal configurations of an omnidirectional mobile service robot," in *Proc. SICE Annu. Conf.*, 2010, pp. 2872–2877.

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

[11] L.-C.-T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Trans. Robot. Autom.*, vol. 7, no. 4, pp. 489–499, Dec. 1991.

[12] R. Muller-Cajar and R. Mukundan, "Triangualation—A new algorithm for inverse kinematics," in *Proc. Image Vis. Comput. New Zealand*, New Zealand, Dec. 2007, pp. 181–186.

[13] A. Aristidou and J. Lasenby, "Inverse kinematics: A review of existing techniques and introduction of a new fast iterative solver," Dept. Eng., Univ. Cambridge, Cambridge, U.K., Tech. Rep. F-INFENG/TR.632, 2009.

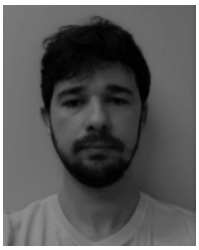
[14] A. Aristidou, Y. Chrysanthou, and J. Lasenby, "Extending FABRIK with model constraints," *Comput. Animation Virtual Worlds*, vol. 27, no. 1, pp. 35–57, Jan. 2016.

[15] J. Huang and C. Pelachaud, "An efficient energy transfer inverse kinematics solution," in *Proc. Int. Conf. Motion Games*. Cham, Switzerland: Springer, 2012, pp. 278–289.
[16] S. Moya and F. Colloud, "A fast geometrically-driven prioritized inverse kinematics solver," in *Proc. Congr. Int. Soc. Biomechanics (ISB)*, 2013, pp. 1–3.
[17] S. Agrawal and M. van de Panne, "Task-based locomotion," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–11, Jul. 2016.
[18] M. A. Y. Abdallah, M. S. Baziyed, R. Fareh, and T. Rabie, "Tracking control for robotic manipulator based on FABRIK algorithm," in *Proc. Adv. Sci. Eng. Technol. Int. Conf. (ASET)*, Feb. 2018, pp. 1–5.
[19] S. Tao and Y. Yang, "Collision-free motion planning of a virtual arm based on the FABRIK algorithm," *Robotica*, vol. 35, no. 6, pp. 1431–1450, Jun. 2017.
[20] 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.
[21] J. Liao, F. Huang, Z. Chen, and B. Yao, "Optimization-based motion planning of mobile manipulator with high degree of kinematic redundancy," *Int. J. Intell. Robot. Appl.*, vol. 3, no. 2, pp. 115–130, Jun. 2019.

PHILLIPE CARDOSO 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, Federal University of Sergipe (GPR-UFS).

RAIMUNDO CARLOS SILVÉRIO FREIRE received the bachelor's degree in electrical engineering from the Federal University of Maranhão, in 1979, the master's degree in electrical engineering from the Federal University of Paraíba, in 1982, the Ph.D. degree in electrical engineering from the Institut National Polytechnique de Lorraine France, in 1988, and the Postdoctoral degree from the Ecole Supérieur des Télécommunications, Paris, France. He is currently a Full Professor with the Federal University of Campina Grande.

ELYSON ADAN NUNES CARVALHO 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, a Permanent Member of the Graduate Program in Electrical Engineering (PROEE), and a Researcher of the Robotics Research Group, Federal University of Sergipe.



LUCAS MOLINA received the bachelor's degree in electronic engineering from the Federal University of Sergipe, 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 and a Researcher with the Robotics Research Group, Federal University of Sergipe (GPR-UFS).

EDUARDO OLIVEIRA 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.