Inverse Kinematics solver using Bezier curve based iterative algorithm with obstacle avoidance ability

Ghani Albar Harahap[1],
[1]Engineering Faculty, International University Liaison Indonesia, Intermark, BSD City, 15310
E-mail: [1]ghani.harahap@iuli.ac.id

Abstract. The inverse kinematics (IK) problem in area of robotics, game and computer graphics is one of the most essential problems. To find the solution, set of joint positions of the articulated model must be calculated so that a specific point of the system can achieve a desired target. Thus, in this paper, the quadratic Bezier curve based iterative algorithm is proposed for IK problem of articulated manipulator. This algorithm doesn't require rotation of angle or matrix computation, has low computational cost and results in position of joints efficiently with high accuracy, smooth and realistic pose. In this paper, the proposed algorithm is tested over a 6 joints articulated planar system. Due to the nature of Bezier curve, this algorithm has a directional obstacle avoidance ability. For 100 tests, the average convergence time for the tested configurations is about (0.0019 s) and the position error between end-effector and target was around (5.0180e-04 mm).

Keywords— *Inverse Kinematics, Bezier curve, articulated manipulator, iterative algorithm*

I. INTRODUCTION

The solution of kinematics is an initial step to carry out a desired task, to generate path and control motion. Forward kinematics calculates the rotation and the displacement of joint angles and the subsequent of the end-effector position can be determined, this can be analyzed using Denavit-Hartenberg parameter. With inverse kinematics, joint parameters are to be determined in order to move the end-effector as smooth and as close as possible to the target and satisfying the constraints of the system. Despite the extensive researches in many areas such as robotics, game, animation and ergonomics, it remains a challenging problem until now. The analytical solution is not always promising as it is confronted with singularity problems, abnormal discontinuities, not unique solutions and high computational cost. Manipulator with high degree of freedom is even more difficult to solve analytically. Thus, the aim of this paper is to propose an iterative algorithm to approximate a good solution which has low computational cost and produces realistic poses. Bezier curve based iterative algorithm creates at first a smooth curve connecting the base and the target. The curve will be discretized and an initial adjusting of the links between the base and the target to the discretized curve can take place. Decrementing the y axis coordinate of the middle control point and repeating the links adjusting will result in the updated joint positions between base and target which solve the IK problem with smooth and realistic pose. Incorporating binary search in decrementing and adjusting ensures that this

algorithm has fast convergence rate and low computational cost.

II. LITERATURE REVIEW

There are various techniques for solving IK problems such as analytical approach, numerical approach, heuristic approach and geometric approach and artificial-intelligence approach. Complete overview can be found in [2, 3, 4], In this paper, the literature overview will be narrowed down to analytical and numerical techniques.

A. ANALYTICAL SOLUTION

Analytical solution incorporates function of the length of the system, initial position and rotation constraints, the solution is typically not unique, thus, some assuptions are usually made to ensure a unique solution [2]. Korein et al. [1] reviewed early techniques of analytical solutions. Early analytical solution of 6R manipulator has been proposed by [5, 6]. IK tool to solve IK equation has been developed by Diankov [7] for motion planning for robotics in real-world applications. An analytical IK method to achieve anthropomorphic arm posture for a given end-effector position and orientation Was proposed by Zhao et al. [8]. Tong et al. [9] proposed a semi-analytical approach for solving redundant sliding manipulators. Nevertheless, non-linear behaviour of kinematics systems of higher degree causes analytical solution not suitable for e.g. 7-DoF redundant systems.

B. NUMERICAL SOLUTION

Numerical solution uses iterative method to minimize a cost function. Typical numerical solutions are based on Jacobian, Newton and heuristic methods. Jacobian based method is a linearization of nonlinear functions and uses Jacobian matrix to change the set of joint positions of kinematic chain, so that the end-effector reaches the target gradually. Gier [10] presented how Jacobian pseudoinverse method applied to 6-DoF for 3D printing Numerous Jacobian based methods using pseudoinverse was presented by Buss [11]. Duleba et al. [12] compared various Jacobian based methods i.e. Jacobian pseudoinverse and transpose method, Damped least square and Modified Damped least square method. Newton method is a solution minimization problem of the target configuration. Thus, the result is a smooth motion without abnormal discontinuities [13]. However, the drawback is that Newton method suffers from high computational complexity. The most popular ones are the Broyden method, the Powell method and the Broyden, Fletcher, Goldfarb, and Shanno method [14]. Duleba and Karcz-Duleba [15] proposed acceleration Newton method of IK problem for robot manipulators. For solving IK problem with prismatic, revolute and spherical joints, a numerical approach for calculating the exact Hessian was presented by Erleben and Andrews [16]. Heuristic methods have been gaining more attention, being simple yet easy to implement, efficient, has fast convergence rate and high accuracy. Typical for heuristic IK solver is CCD [17, 18, 19, 20], Triangulation [21, 22] and FABRIK [13, 23, 24] . CCD is an iterative heuristic IK solver which is computationally fast and can run at interactive frame rates. However, CCD tends to overemphasize the movement of joints closer to the end-effector of the kinematic chain. Triangulation is a non-iterative approach that uses the cosinus law to determine each joint angle from the base to the end-effector. It has a lower computational cost than CCD, since it needs only 1 iteration to reach the target, however, the pose is often unrealistic. Instead of using rotational angles or matrices, FABRIK calculates each joint position via locating a point in a line. Songqiao Tao et al. [23] extended FABRIK algorithm with obstacle avoidance.

III. METHODOLOGY

Bezier curve is a parametric curve and widely used in computer graphics, game and related fields, it is used to create a smooth curve and path, it can move in all direction and perform loops, it's properties like the normals and tangents are convenient to orient object and can be used to create coordinate space around each points which can be used to generate procedural geometry. A Bezier curve is defined by a set of control points from $\boldsymbol{P_0}$ to $\boldsymbol{P_n}$ where n denotes the order (n =1 for linear, n= 2 for quadratic, n= 3 for cubic, etc. (see figure 1)). Bezier curve of order n+1 is represented by

$$C(t) = \sum_{i=0}^{n} \boldsymbol{p_i} B_{i,n}(t), \quad 0 \le t \le 1 \tag{1}$$

Where $\boldsymbol{p_i}$ are the control points and $B_{i,n}$ the basis functions, they together determine the shape of the curve.

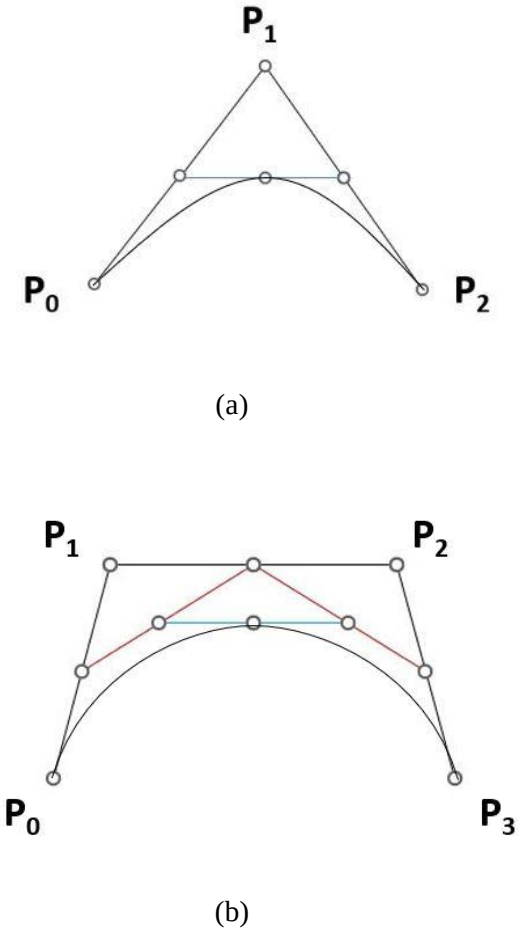


Figure 1: Example of Bezier curve. a) Quadratic Bezier curve with three control points. b) Cubic Bezier curve with four control points.

A. LINEAR INTERPOLATION

Given two points $\boldsymbol{P_0}$ and $\boldsymbol{P_1}$ and a line segment connecting those two points. A point $\boldsymbol{P}$ between $\boldsymbol{P_0}$ and $\boldsymbol{P_1}$ (figure 2) can be described by a linear interpolation

$$p(t) = (1-t)P_0 + tP_1, \quad 0 \le t \le 1 \tag{2}$$

t-value defines the position of $\boldsymbol{P}$, t-value of zero moves the point to $\boldsymbol{P_0}$ and t-value of one moves the point to $\boldsymbol{P_1}$, and any value in between is a blend between those two.

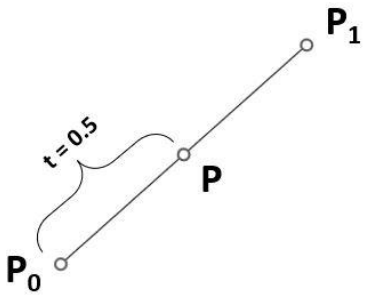


Figure 2: Linear interpolation between points P_0 and P_1.

B. QUADRATIC BEZIER CURVE

A Quadratic Bezier curve can be interpreted as a linear interpolant between a point on a linear interpolation from $\boldsymbol{P_0}$ and $\boldsymbol{P_1}$ and from $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ described by

$$p(t) = (1-t)\big((1-t)P_0 + tP_1\big) + t\,((1-t)P_1 + tP_2),\ 0 \le t \le 1 \quad (3)$$

And can be rearranged to

$$p(t) = (1-t)^2 P_0 + (1-t)2tP_1 + t^2 P_2\,,\quad 0 \le t \le 1 \quad (4)$$

With three control points $\boldsymbol{P_0}$, $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$, equation (1) can be simplified to

$$C(t) = \sum_{i=0}^{2} \boldsymbol{p_i} B_{i,2}(t),\ 0 \le t \le 1 \quad (5)$$

With

$$B_{0,2}(t) = (1-t)^2 \quad (6)$$

$$B_{1,2}(t) = 2t(1-t) \quad (7)$$

$$B_{2,2}(t) = (t)^2 \quad (8)$$

generating the terms of series results in

$$C(t) = \boldsymbol{P_0}(1-t)^2 + \boldsymbol{P_1}2t(1-t) + \boldsymbol{P_2}(t)^2 \quad (9)$$

Let $\boldsymbol{P_0}$ = (x_0 , y_0), $\boldsymbol{P_1}$ = (x_1 , y_1), $\boldsymbol{P_2}$=(x_1 ,y_1), then two separate functions for coordinates $\boldsymbol{x}$ and $\boldsymbol{y}$ can be written

$$x(t) = \boldsymbol{x_0}(1-t)^2 + \boldsymbol{x_1}2t(1-t) + \boldsymbol{x_2}(t)^2 \quad (10)$$

$$y(t) = \boldsymbol{y_0}(1-t)^2 + \boldsymbol{y_1}2t(1-t) + \boldsymbol{y_2}(t)^2 \quad (11)$$

C. ARTICULATED OBJECT

The configuration tested for this paper is a system composed of five serial rigid links connected by revolute joints in a two dimensional plane. The joints consist of base joint, which has no parent joint and has one child joint, interconnected joints, each has one parent joint and one child joint, and an end-effector. The end-effector has no child joint. The rotational axis is the normal axis to the plane, that is, either joints rotation about *x*-axis, *y*-axis or *z*-axis. Figure 3 represents the configuration set up in this paper, which the inverse kinematics solver should be applied to, $\boldsymbol{l_i}$, $1 \le i \le 5$ are the links, $\boldsymbol{l_1}$ ist the link connecting base joint and it's child joint, $\boldsymbol{l_5}$ is the link connecting end-effector and the parent joint before it. $\boldsymbol{j_i}$, $1 \le i \le 6$ are the joints connecting the links, $\boldsymbol{j_1}$ ist the base joint, $\boldsymbol{j_6}$ is the joint of end-effector. Given a target, the end-effector should be placed on the target, all joints positions except base joint should be updated accordingly.

D. ITERATIVE INVERSE KINEMATICS SOLVER BASED ON QUADRATIC BEZIER CURVE

In the following, figure 4 illustrates how the algorithm works for a simple system, the system composed of five links, six joints and one target. Three control points (base joint, end-effector joint and target), which construct quadratic bezier curve, form a control polygon $\boldsymbol{P_0P_1P_2}$ (figure 4a). In the first step, from the initial control polygon, the curve will be partitioned with *n* equidistant points, where the curve points are defined by $C_i \in C(t)$, $i \in [1, n]$, where $C_1 = j_1$ and $C_n = T$ (with T is target), then

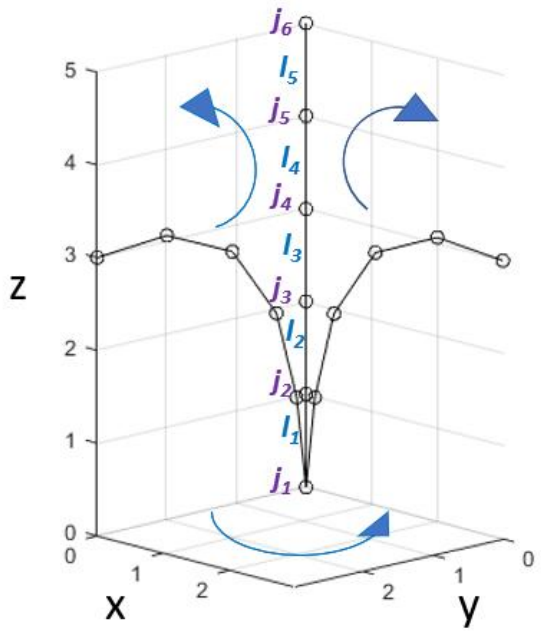


Figure 3: Rotating joints of articulated object about z axis, x axis and y axis

define curve points C_j, $i < j \leq n$, and define d_i, where d_i represents the length of the link between joint i and joint $i+1$. In the second step, the curve length will be estimated by $length\,(C(t)) = \sum_{i=1}^{n} |C_{i+1} - C_i|$ and calculate $d = \sum_{i=1}^{m-1} d_i$, where d_{m-1} is the length of link connecting end-effector joint and the parent joint before it. After moving last joint to the target $j_m = T$, then If $d < length\,(C(t))$ the iteration starts from C_i , find C_j such that $|C_j - C_i| = d_i$ and move j_{i+1} to the position of C_j and set $C_{i+1} = C_j$, the new position of joint j_{i+1} is denoted by j'_{i+1},. The next iteration starts at C_{i+1} and a full iteration ends if j_{m-1} is moved to the curve. This is based on the linear search, however, binary search can be implemented for much faster convergence time and reduction of number of iterations (see the pseudo-code section). Repeating above iteration until the last joint will result in updated joints position (figure 4b). After one complete procedure, the joints are all moved to the curve with the correct length between the joints, except for the link between end-effector joint and the target, which seems longer than it should be $|T - j_{m-1}| \neq d_{m-1}$. Thus, In the third step, the main task is finding the appropriate position of the middle control point $\boldsymbol{P_1}$, which is crucial for determining the satisfactory control polygon and, the related Bezier curve and the joint positions. To get a sequence of control polygons and curves, the y coordinate of middle control point, $\boldsymbol{P_1}$, will be shifted downwards iteratively (figure 4c). In each iteration of new position of $\boldsymbol{P_{1,i}}$, $i \in [1, k]$ (k is the number of equdistant points between y coordinate of base joint and last joint), the first and second steps above will be repeated until the condition $|T - j_{m-1}| = d_{m-1}$ is satisfied and the final position of $\boldsymbol{P_{1,final}}$ found. The result is the final pose of joint positions. However, shifting or reflecting the control point $\boldsymbol{P_1}$ in x axis direction can be considered :

1) To smoothen the final pose.
2) If $length\,(C(t)) \approx d$, that is, if the initial curve length produced by the initial control polygon is still greater than the sum of length of the articulated manipulator links, yet too close.
3) To avoid an obstacle.

Figure 5 represents how the final pose can be smoothened. After $\boldsymbol{P_{1,final}}$ calculated iteratively from $\boldsymbol{P_{1,1}}$ (figure 5a), the pose can be improved by shifting the position of $\boldsymbol{P_{1,1}}$ in positive x axis direction to $\boldsymbol{P_{1,1}^{shifted}}$, thus, after finding resulting $\boldsymbol{P_{1,final}^{shifted}}$, resulting in a new control polygon and the updated improved pose (figure 5b). Point 2) and 3) will be discussed in the next sections.

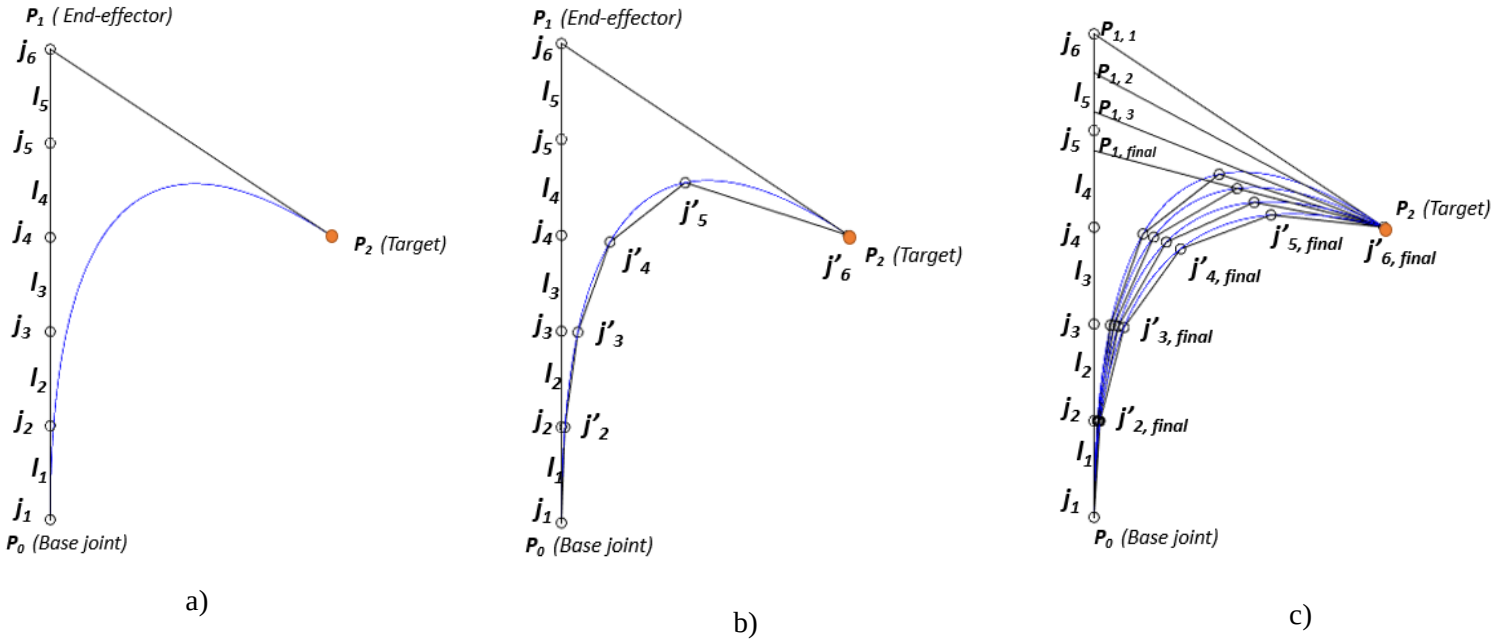


Figure 4: Iterative procedure. a) Construct an initial control polygon and the Bezier curve. b) Adjust the links and joints iteratively to the curve. c) Find the new position $\boldsymbol{P_{1,i}}$ iteratively until the final position $\boldsymbol{P_{1,final}}$ found and the target is reached.

E. CRITICAL ZONE

If the length of the links of the articulated manipulator is close to the curve length, $length\left(C(t)\right) \approx d$, the adjusment of the links to the curve will become less precise. A workaround for this problem is shifting the control point $\boldsymbol{P_{1,1}}$. Exemplifying of this situation will be presented in chapter III (experimental results).

F. SAFE ZONE

If the distance between base joint and target is too close, the length of the links of the articulated manipulator becomes less than the curve, $length\left(C(t)\right) < d$, the adjusment of the links to the curve is not possible anymore. To prevent this, the user might adjust the length of the links or reduce the number of joints or links before performing the IK test.

G. OBSTACLE AVOIDANCE ABILITY

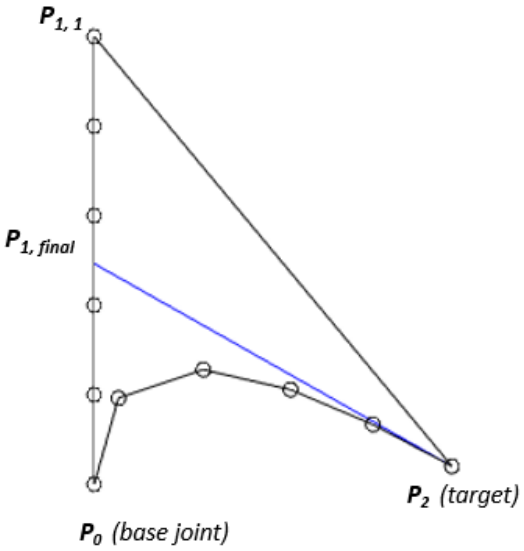
In many situations, articulated manipulators work incomplex environments where interferences with surrounding can occur. In this section, additional strategy will be added to the algorithm and presented to resolve such problem. As mentioned previously, one of the advantages of Bezier curve is that it allows for obstacle avoidance ability. This scenario can be seen in picture , the manipulator with the initial control polygon and control point $\boldsymbol{P_{1,1}}$, confronted with an obstacle. To avoid this, the position of $\boldsymbol{P_{1,1}}$ will be shifted iteratively in negative x axis direction to $\boldsymbol{P_{1,1}^{shifted}}$ and a new control polygon will be produced, $\boldsymbol{P_{1,final}^{shifted}}$ can then iteratively calculated, resulting in a final control polygon and the updated final pose of manipulator with no interference. If shifting the position of $\boldsymbol{P_{1,1}}$ repeatedly doesn't succeed (e.g. due to several obstacles blocking the path), reflecting the control point $\boldsymbol{P_{1,1}}$ can be considered. Figure 7a) illustrates that the point $\boldsymbol{P_{1,1}}$ can be reflected to x axis and afterwards shifted, assumed that there is no constrain underneath the base joint limiting the path of the manipulator. If several attempts to reach the target by shifting the control point $\boldsymbol{P_{1,1}}$ fails, reflecting it and shifting it to $\boldsymbol{P_{1,1}^{reflected\ shifted}}$ can be Performed. After the point $\boldsymbol{P_{1,final}^{reflected\ shifted}}$ and the corresponding

control polygon found, the final attempt to reach target is accomplished (figure 7b).

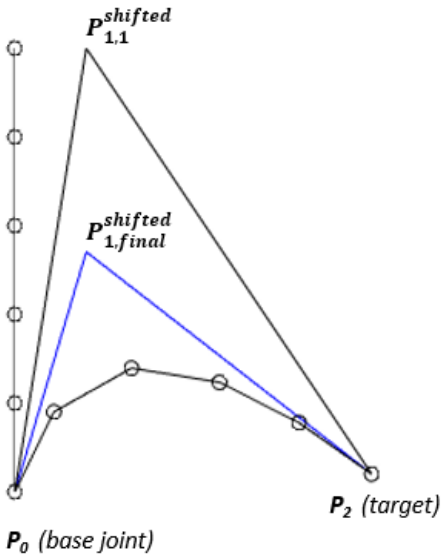
Pseudo Code of the Inverse Kinematics routine

Input: Joint positions j_i, target position $\boldsymbol{T}$ and distance between each joint $d_i = |j_{i+1} - j_i|$ for i = 1,...,m-1.
First control point $\boldsymbol{P_0}$, second control point $\boldsymbol{P_1}$, third control point $\boldsymbol{P_2}$
Output: new joint positions j'_{i+1} ,..., j'_{m-1}

1 % Calculate distance between base and target
2 dist = $|T - j_1|$
3 % If the target is reachable
4 If dist < $d = \sum_{i=1}^{m} d_i$
5 % Create n equdistant points on the curve
t = linearspace(0, 1, n);
6 % Create a discretized Bezier curve
$C(t) \approx C_j$
$C_j = \boldsymbol{P_0}(1-t)^2 + \boldsymbol{P_1}\, 2t(1-t) + \boldsymbol{P_2}(t)^2$
7 % Move last joint to the target $j_m = T$
8 % Initial adjusting of the links between j_i and j_{i+1} to the curve either with linear search or binary search
8a % Linear search
% All joints except base and end-effector the iteration will be conducted for all (m-2)-joints (m = number of joints)
for i = 1 : length(t) , $i < j \leq$ n
if ($|C_j - C_i| = d_i$) ,
$C_{i+1} = C_j$;
End
End
8b % Binary search : starting in the middle of interval [0 n], the iteration will be conducted for all (m-2)-joints (m = number of joints)
up = n; low = 1; mid = floor($\frac{\text{low+up}}{2}$);
While $|C_{mid} - j_i| \neq d_i$ do
If ($|C_{mid} - j_i| > d_i$)
up = mid; mid = floor($\frac{\text{low+up}-1}{2}$);
Else if ($|C_{mid} - j_i| < d_i$)
low = mid; mid = floor($\frac{\text{low+1+up}}{2}$);
End
End
$j_{i+1} = C_{mid}$
9 % Create k equdistant points from y coordinate of $\boldsymbol{P_0}$ to y coordinate of $\boldsymbol{P_1}$
y_P = linearspace(y_{P0}, y_{P1}, k);
10 % Finding the position of $\boldsymbol{P_{1,final}}$ iteratively either with linear search or binary search, and repeat step 6 and 8.
10a % Linear search:
for i = 1 : length(y_P)
$y_{P1} = y_{P1,i}$; repeat step 6 and 8a;
If $|T - j_{m-1}| = d_{m-1}$
Break;
End
End
10b % Binary search: starting in the middle of interval [$y_{midP0P1}$, y_{P1}] with
$y_{midP0P1}$ = floor ($\frac{y_{P0} + y_{P1}}{2}$)
low = $y_{midP0P1}$; up = y_{P1} ; mid= y_{P1};
While $|T - j_{m-1}| \neq d_{m-1}$ do
If ($|T - j_{m-1}| > d_{m-1}$) % This is always the case in the 1st iteration
up = mid; mid = floor($\frac{\text{low+up}-1}{2}$)
Else if ($|T - j_{m-1}| < d_{m-1}$)
low = mid; mid = floor($\frac{\text{low+1+up}}{2}$)
repeat step 6 and 8b;
End
% if convergence requires that $\boldsymbol{P_1}$ to be smaller than $y_{midP0P1}$, then extend the lower bound to y_{P0}
if (mid==1)
y_P = linspace(y_{P0}, y_p(mid), k);
up = length(y_P);
low = y_{P0};
End
y_{P1} = mid;
End
End
$C_i = [Base, j_{i+1}, \ldots, j_{m-1}, Target\,]$;

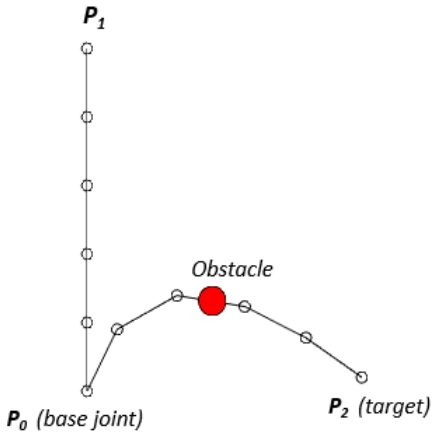


a)

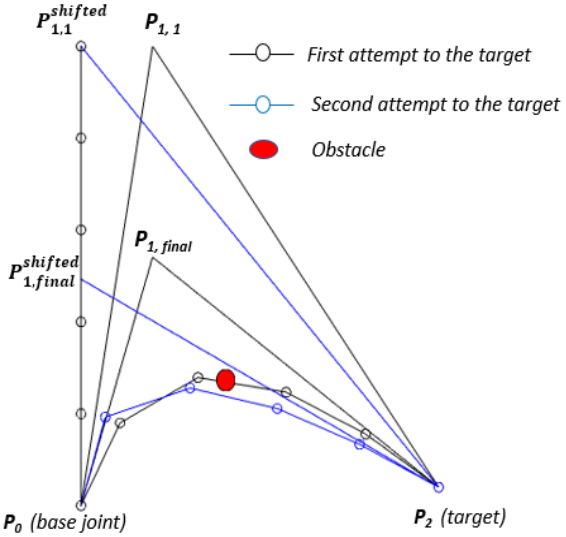


b)

Figure 5: smoothening the curve. a) Final pose without shifting the point $P_{1,1}$. b) Final pose with the point $P_{1,1}$ shifted.

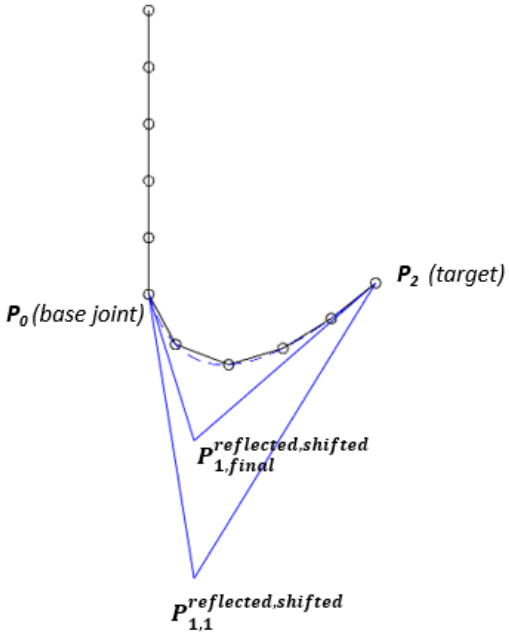


a)

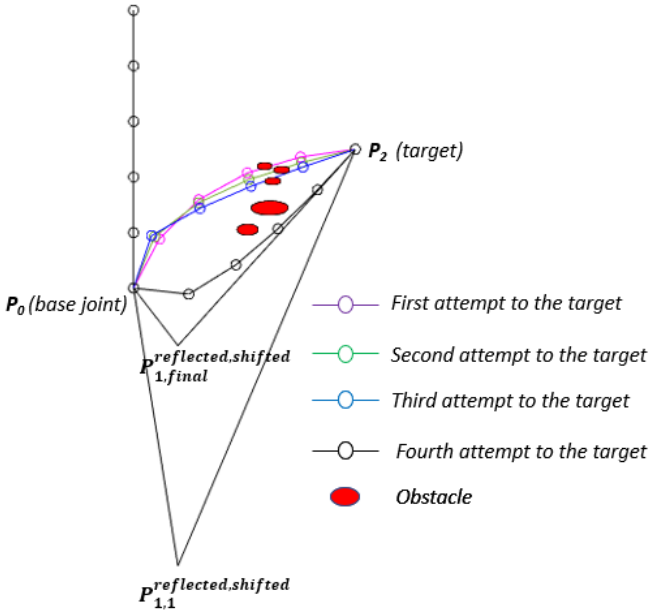


b)

Figure 6: Obstacle avoidance. a) Final pose with an interference. b) Final pose without interference by shifting the control point $P_{1,1}$.



a)



b)

IV. RESULTS

To verify the presented algorithm, Matlab was used to test whether the end-effector of the articulated manipulator can reach a set of target positions and to test whether directional obstacles can be avoided. The 2D model is a six-joint manipulator with the joint coordinates (0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), the base of the manipulator is fixed to the ground, the end-effector can be positioned within the safe zone and critical zone (with shifting of control point $\boldsymbol{P_1}$). Represented in figure 8 are the coordinates of the target positions set at (4, 0), (4, 1.5), (4, 3), (3, 0), (3, 2), (3, 4), (2, 0), (2, 2) and (2, 4) which are labeled as orange circles, the initial state of the manipulator (vertical and labeled as black) and the final postures (labeled as black). The coordinates of the target positions are within the safe zone except (2, 2). Figure 9 represents how directional obstacles can be avoided by shifting the control point $\boldsymbol{P_1}$. The initial state of the manipulator is labeled as black, the first, second and third attempt to the target are respectively labeled as purple, blue and black, the target coordinate used for the obstacle avoidance test is (2, 4) and labeled as orangle circle, the red round objects are representing the obstacles and positioned at (1.5, 3.9) (figure 9a) and at (1.5, 3.9) and (1.7, 3.5) (figure 9b). In the experimental results, adjusting the links to the curve and decrementing the position of $P_{1,i}$ by linear search (target position (4, 0) and (4, 2)) and by binary search (target position (4, 0), (4, 1.5), (4, 3), (3, 0), (3, 2), (3, 4), (2, 0), (2, 2), (2, 4)) are presented. For linear search and binary search, the number of discrete points n on curve and the number of discrete points k between $\boldsymbol{P_0}$ and $\boldsymbol{P_1}$, are respectively 4101 and 800, and 5001 and 900. For obstacle avoidance test, the algorithm used linear search with n and k respectively 4101 and 800. The statistical results of results in figure 8 are respectively shown in table 1. Table 1 presents the number of iterations needed, the lowest distance that can be achieved between end-effector and target positions, run time and both searching methods for links adjusting to the curve as well as for finding $\boldsymbol{P_{1,final}}$. It is shown that finding $\boldsymbol{P_{1,final}}$ by binary search is better than linear search since the requiered number of iterations and the run time of are drastically reduced, though the run time at

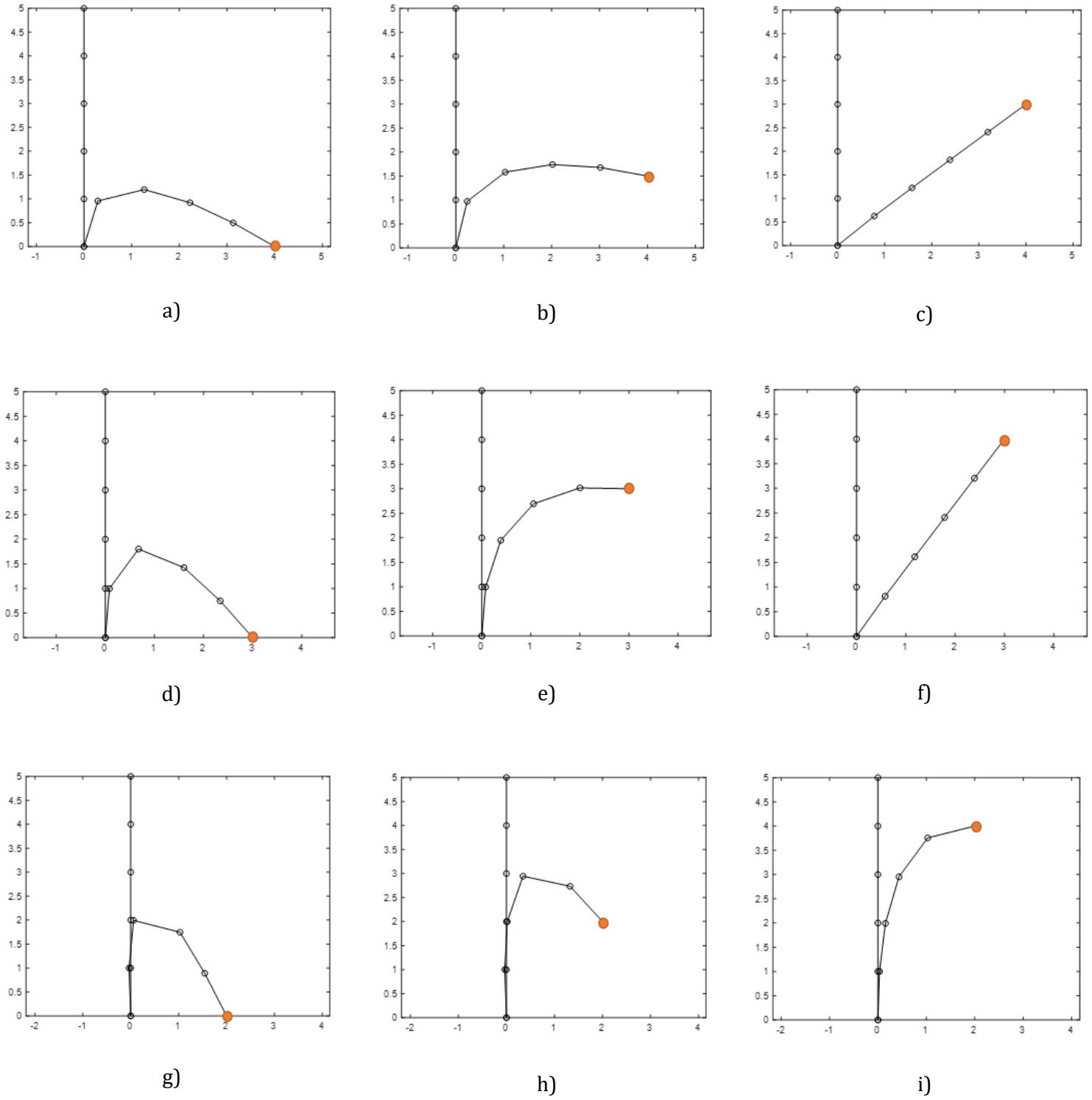
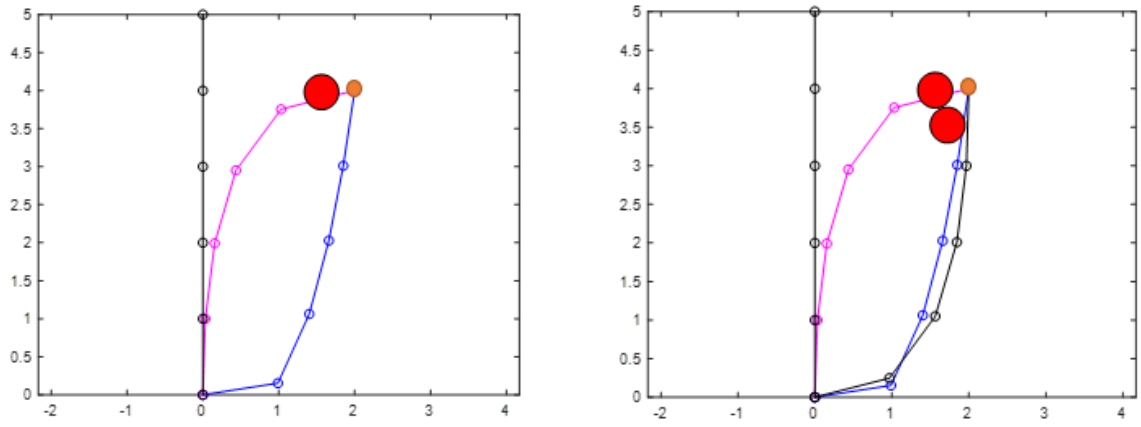


Figure 8: Visualization of experimental results of Bezier curve based iterative solver tested over a kinematic chain with 6 joints



a) b)

Figure 10: Directional obstacle avoidance. a) first attempt failed due to interference with one obstacle, second attempt succeeded. b) first and second attempt failed due to two obstacles, third attempt succeeded.

Table 1: Average results (over 10 runs) for the kinematic chain with 6 joints

	Number of iterations	**Distance between end-effector and target (mm)**	**Matlab run time (s)**	**Method of adjusting links to the discretized curve**	**Method of searching for $P_{1,final}$**	**Whether $P_{1,1}$ shifted ?**	**Result**
1	504	6.7196e-04	0.0012	Binary search	Binary search	No	Figure 8a
2	544	8.5708e-04	0.0026	Binary search	Binary search	No	Figure 8b
3	615	2.4320e-04	0.0025	Binary search	Binary search	No	Figure 8c
4	351	3.3380e-04	0.0028	Binary search	Binary search	No	Figure 8d
5	288	3.6458e-04	0.0018	Binary search	Binary search	No	Figure 8e
6	546	7.3012e-04	0.0019	Binary search	Binary search	No	Figure 8f
7	418	9.0282e-04	0.0018	Binary search	Binary search	No	Figure 8g
8	1	0.0655	3.78e-04	Binary search	Binary search	No	Figure 8h
9	280	3.9340e-05	0.0012	Binary search	Binary search	No	Figure 8i
10	616032	6.7196e-04	0.1752	Linear search	Linear search	No	Figure 8a
11	591150	8.5708e-04	0.1596	Linear search	Linear search	No	Figure 8b
12	272	3.7331e-04	0.0010	Binary search	Binary search	Yes (0.2 to the left)	Figure 8h

linear search is fairly low. Figure 8h represents an initial state of the manipulator and a target position in a critical zone (2, 2), its lowest distance between end-effector and the target, which can be achieved with the given parameters, is clearly not as satisfactory as the other results. By shifting the point P1 by 0.2 to the left, a much more precise distance can be obtained. In figure 9a, in order to avoid the obstacle in the first attempt, the position of P1 was shifted by 1.3 to the right and the second attempt succeeded. In figure 9b, another obstacle is positioned within the path of the second posture of the manipulator, thus, the position of $\boldsymbol{P_1}$ was shifted subsequently by 0.7 to the right and the third attempt prevailed.

IV. CONCLUSIONS

In the previous sections a new method of solving IK problem utilizing Bezier curve was described. The advantage of this method is there is always a smooth realistic curve produced from the base joint to the target where the base joint and the end-effector are excluded from the calculation and only the interior joints need to be considered. This method gains an upper hand compared to Jacobian numerical approaches in terms of being able to produce realistic postures without erratic discontinuities. This IK solver is computationally efficient, especially if binary search is utilized, the number of iterations is equal to $\log_2 k \cdot ((\mathrm{m} - 2) \cdot \log_2 n)$ for binary search and $\mathrm{k} \cdot (m - 2) \cdot \mathrm{n}$ for linear search where k is the number of points between $\boldsymbol{P_0}$ and $\boldsymbol{P_1}$, m the number of joints and n the number of points on the curve. Another advantage is also that this method can avoid directional obstacles by shifting the point $\boldsymbol{P_1}$ as demonstrated in the previous sections. The next milestone is to improve the stability and optimize the test results for the proposed method.

Journal Article

[1] KOREIN J. U., BADLER N. I.: Techniques for Generating the Goal-directed Motion of Articulated Structures. IEEE Computer Graphics and Applications 2, 9 (1982): 71–81.

[2] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, “Inverse kinematics techniques in computer graphics: a survey,” Computer Graphics Forum, vol. 37, no. 6, pp. 35–58, 2018.

[3] J. Craig, Introduction to Robotics: Mechanics and Control, Addison-Wesley Longman Publishing Co., Reading, MA, USA, 3rd edition, 2003.

[4] A. Aristidou and J. Lasenby, Inverse Kinematics: A Review of Existing Techniques and Introduction of a New Fast Iterative Solver, Cambridge University Engineering Department, Cambridge, UK, 2009.

[5] M. Raghavan and B. Roth, “Inverse kinematics of the general 6R manipulator and related linkages,” Journal of Mechanical Design, vol. 115, no. 3, pp. 502–508, 1993.

[6] D. Manocha and J. F. Canny, “Efficient inverse kinematics for general 6R manipulators,” IEEE Transactions on Robotics and Automation, vol. 10, no. 5, pp. 648–657, 1994.

[7] P. R. Diankov, ““Automated construction of robotic manipulation programs”,” Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, USA, 2010, Ph.D. thesis.

[8] J. Zhao, “Analytical inverse kinematics of anthropomorphic movements for 7-DOF humanoid manipulators,” Journal of Mechanical Engineering, vol. 54, no. 21, p. 25, 2018.

[9] Y. Tong, J. Liu, Y. Liu, and Y. Yuan, “Analytical inverse kinematic computation for 7-DOF redundant sliding manipulators,” Mechanism and Machine Theory, vol. 155, Article ID 104006, 2021.

[10] Gier, M. R. de, “Control of a robotic arm: Application to on-surface 3Dprinting,” Delft University of Technology, 2015.

[11] Buss, Samule R, “Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods,” University of California, San Diego, 2009.

[12] Dulęba, Ignacy, and Michał Opałka. "A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators." International Journal of Applied Mathematics and Computer Science 23.2 (2013).

[13] A. Aristidou, C. Yiorgos, and L. Joan, “Extending FABRIK with model constraints,” Computer Animation & Virtual Worlds, vol. 27, no. 1, pp. 35–57, 2016.

[14] R. Fletcher, Practical Methods of Optimization, Wiley-Interscience, New York, NY, USA, 2nd edition, 1987.

[15] I. Duleba and I. Karcz-Duleba, “Accelerating Newton algorithms of inverse kinematics for robot manipulators,” in Proceedings of the 2018 23rd international conference on methods & models in automation & robotics (MMAR), pp. 50–372, Faculty of Electronics, Wroclaw University of Science and Technology, Wroclaw, Poland, August 2018.

[16] K. Erleben and S. Andrews, “Solving inverse kinematics using exact Hessian matrices,” Computers & Graphics, vol. 78, pp. 1–11, 2019.

[17] L.-C. T. Wang and C. C. Chen, “A combined optimization method for solving the inverse kinematics problems of mechanical manipulators,” IEEE Transactions on Robotics and Automation, vol. 7, no. 4, pp. 489–499, 1991.

[18] A. L. Olsen and H. G. Petersen, “Inverse kinematics by numerical and analytical cyclic coordinate descent,” Robotica, vol. 29, no. 4, pp. 619–626, 2011.

[19] A. N. Kamal and H. Jing, “Constrained cyclic coordinate descent for cryo-EM images at medium resolutions: beyond the protein loop closure problem,” Robotica, vol. 34, no. 8, pp. 1777–1790, 2016.

[20] D. Bao, “Exploration of virtual human running based on CCD algorithm,” Concurrency and Computation Practice and Experience, vol. 31, no. 10, Article ID e4901, 2019.

[21] R. Muller-Cajar and R. Mukundan, Triangualation—A New Algorithm for Inverse Kinematics, University of Canterbury Computer Science & Software Engineering, Christchurch, New Zealand, 2007.P. R. Diankov, ““Automated construction of robotic manipulation programs”,” Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, USA, 2010, Ph.D. thesis.

[22] R. Mukundan, “A robust inverse kinematics algorithm for animating a joint chain,” International Journal of Computer Applications in Technology, vol. 34, no. 4, pp. 303–308, 2009.Y. Tong, J. Liu, Y. Liu, and Y. Yuan, “Analytical inverse kinematic computation for 7-DOF redundant sliding manipulators,” Mechanism and Machine Theory, vol. 155, Article ID 104006, 2021.

[23] Tao. S, Tao. H, and Yang. Y “Extending FABRIK with Obstacle Avoidance for Solving the Inverse Kinematics Problem,” Hindawi, Journal of Robotics, vol. 2021, no. 2, pp. 1-10, 2021.Buss, Samule R, “Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods,” University of California, San Diego, 2009.

[24] A. Aristidou and J. Lasenby, “FABRIK: a fast, iterative solver for the Inverse Kinematics problem,” Graphical Models, vol. 73, no. 5, pp. 243–260, 2011.A. Aristidou, C. Yiorgos, and L. Joan, “Extending FABRIK with model constraints,” Computer Animation & Virtual Worlds, vol. 27, no. 1, pp. 35–57,2016.