A Fast Inverse Kinematics Solution for an *n*-link Joint Chain

Ramakrishnan Mukundan, *Senior Member, IEEE*

Abstract—**The Cyclic Coordinate Descent (CCD) is a well known algorithm used for inverse kinematics solutions in applications involving joint-chains and moving targets. Even though a CCD algorithm can be easily implemented, it can take a series of iterations before converging to a solution, and also generate undesirable joint rotations. This paper presents a novel single-pass algorithm that is fast and eliminates problems associated with improper and large angle rotations. Experimental results are presented to show the performance benefits of the proposed algorithm over CCD and the "triangulation" methods, using different types of cost functions.**

Index Terms— **Character animation, Cyclic coordinate descent, Goal-directed motion, Inverse kinematics.**

I. INTRODUCTION

Animation of an articulated structure often requires inverse kinematics (IK) solutions, when only the desired positions of the end-effectors are given. When the number of links in a joint-chain becomes greater than 3, analytical methods usually become complex and difficult to implement. Iterative numerical methods are therefore commonly used in Robotics and Computer Graphics applications [1]-[3]. An important area where IK algorithms are used is character animation where joint angles of 3D character models are needed to be computed for achieving a goal-directed motion [4]. Character animation techniques based on motion capture data also require IK solutions for mapping interpolated data to joint positions [5].

One of the well-known algorithms used for computing joint angles from target positions, is the Cyclic Coordinate Descent (CCD) method [6]. Even though this method can be easily implemented, it has several drawbacks such as the requirement for a number of iterations for certain configurations, and undesirable joint angle rotations performed for certain target positions. In order to eliminate some of these problems, a "triangulation" method was recently proposed [7]. This algorithm tries to reduce an *n*-link IK problem into a two-link IK problem using the method of triangulation, to reach the target position. However, the main drawback of this method is the need to often rotate a joint by a large angle greater than 100 degrees, which may have to be avoided in many situations with joint angle constraints.

This paper presents an improved version of the triangulation algorithm, designed to provide solutions without large angle rotations. The proposed algorithm is a "single-pass" algorithm in the sense that each link is rotated at most once in an attempt to find a solution. The above characteristics make the proposed algorithm both fast and useful for graphics applications involving multi-joint chains. The paper also presents results of experimental analysis comparing CCD and the triangulation method with the proposed algorithm using different types of cost functions. The paper is organized as follows: The next section gives an overview of the CCD algorithm and outlines its drawbacks. Section 3 gives a description of the triangulation method given in [7]. Section 4 presents the proposed algorithm. Experimental results are presented in Section 5. Concluding remarks and possible future extensions are discussed in Section 6.

II. CYCLIC COORDINATE DESCENT

We present below an outline of the CCD algorithm for an *n*-link chain as shown in Fig. 1, with the following notations:

(x_T, y_T) : Position of the target

(x_E, y_E) : Position of the end-effector

(x_i , y_i) : Pivot point of the i^{th} link, i=1, 2, …, n

$\boldsymbol{t}_i$: Target vector for the i^{th} link = $(x_T - x_i , y_T - y_i)$

$\boldsymbol{e}_i$: End-effector vector for the i^{th} link = $(x_E - x_i , y_E - y_i)$

α_i : Angle between vectors $\boldsymbol{t}_i$ and $\boldsymbol{v}_i$.

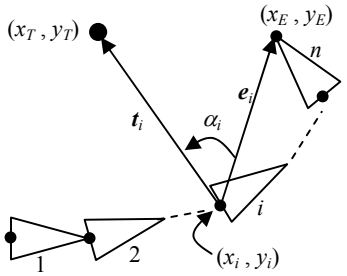


Fig. 1. An *n*-link joint chain.

The CCD algorithm can be concisely given as follows:

1: Set $i = n$.

2: Compute α_i

3: Rotate i^{th} link by angle α_i so that the end-effector meets the target vector $\boldsymbol{t}_i$

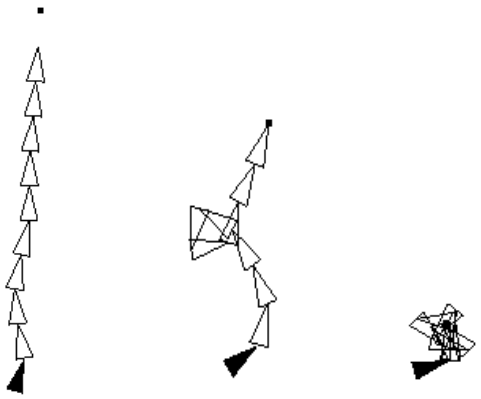
4: Decrement i, and go to step 2 if $i > 0$.

5: Repeat steps 1 to 4 until target position is reached. We count each repetition of the above steps as one iteration.

R. Mukundan is with the Department of Computer Science and Software Engineering at University of Canterbury, Christchurch, New Zealand. (telephone: 643-364-2987 x7770, e-mail: mukundan@canterbury.ac.nz).

ICITA2008 ISBN: 978-0-9803267-2-7

CCD's drawbacks are known to the graphics community. Three typical problems are illustrated in Fig. 2, using a 10-link joint chain. Throughout this paper, the initial configuration of the joint chain is assumed to be such that the base (triangle with solid color) is located at the origin, and every link is axis aligned with respect to the *x*-axis.



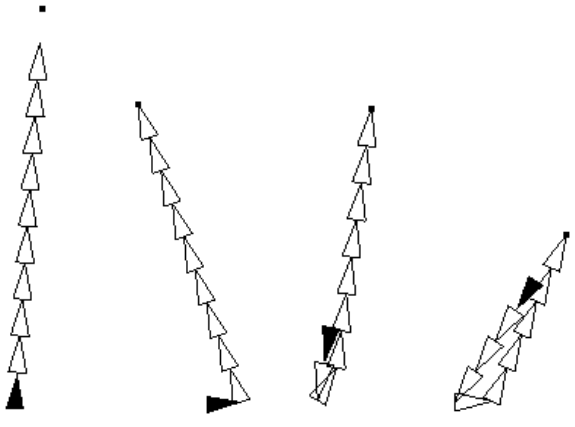
(a) (b) (c)

Fig. 2. Typical problems associated with CCD algorithm.

For the configuration shown in Fig. 2(a), the algorithm requires 100 iterations, though the target could be reached using a single rotation about the base. A simpler solution is possible in the case of Fig. 2(b), whereas the CCD algorithm causes the chain to form a loop, intersecting itself. In Fig. 2 (c), the target position is located close to the base, and the joint chain gets crumbled together to reach the target.

III. TRIANGULATION ALGORITHM

The triangulation algorithm given in [7] takes into account the distance of the target from the base, and rotates the entire chain (*i.e.,* performs a rotation of the base by angle α_1) if the target is not reachable (Fig. 3(a)).



(a) (b) (c) (d)

Fig. 3. The configurations of the joint chain generated by the triangulation algorithm for different target positions.

If the target distance is less than the total length of the chain, we have to consider several possibilities. These are explained with the help of the following diagram (Fig. 4). Here we assume that each link has a length *l*, so that the total length of the chain is *nl*. The distance from the base of a joint chain to the target is denoted by *c*.

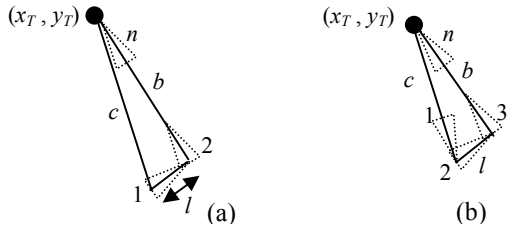


Fig. 4. Triangulation Algorithm.

The triangulation algorithm tries to split the joint chain into two parts consisting of the current link (index 1) of length *l*, and the remaining links forming a single segment of length *b*. As the name implies, the algorithm then tries to form a triangle with *c*, *l*, *b* as sides so that the end-effector can reach the target (Fig. 4(a)). The condition for this to be possible is

$$b - l \le c \le b + l \tag{1}$$

If the target is close to the base of the joint chain where $c < b - l$, then the first link is rotated in the direction opposite to the target vector and is aligned with it (Fig. 4(b)), so that *c* is effectively increased by *l* and *b* reduced by *l*. If condition (1) is still not satisfied, the next link is also rotated to align with the target vector and the process continues till (1) is satisfied. This situation is illustrated in Fig. 3(d). More details about the triangulation algorithm can be found in [7].

IV. PROPOSED ALGORITHM

The triangulation algorithm obviously performs large angle rotations in order to reach the target. For example, in Fig. 3(c), a link is rotated by nearly 165 degrees. Large angle rotations are not acceptable in many situations where joint constraints limit rotations to a maximum value (typically in the range 90-150 degrees). Using the triangulation algorithm, a target can be approached only from the side of the base, whereas a more natural way to approach a target that is located close to the base is to go around it and try to reach it from the opposite side of the base. We take into consideration the above aspects, and propose an improved version of the triangulation algorithm below.

We first try to maximize the minimum angle within the triangle in Fig. 4(a), by splitting the total length $b+l$ evenly. This is done by performing the rotation on a link *k* that is closest to the midpoint of the remaining chain (Fig. 5(a)). Thus we will have the configuration where the sides of the triangle are *a*, *b*, and *c*, with the condition (1) changed to

$$|a - b| \le c \le a + b \tag{2}$$

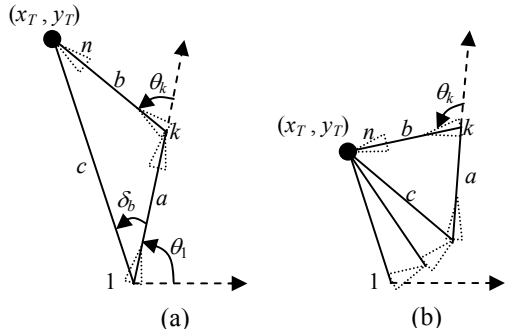


Fig. 5. Improvements to the triangulation algorithm.

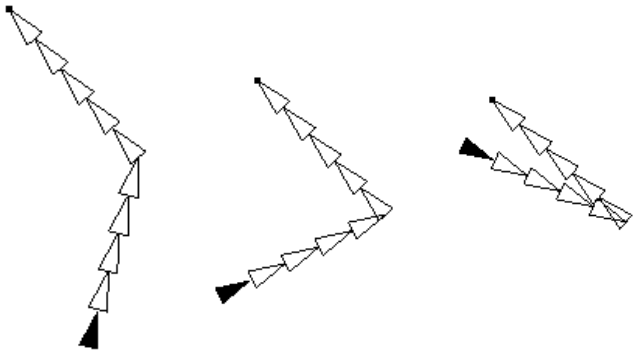
The joint angles at nodes with indices 1 and k are denoted by θ_1 , θ_k respectively, and are computed as follows:

$$\delta_b = \cos^{-1}\left(\frac{a^2 + c^2 - b^2}{2ac}\right)$$

$$\theta_1 = \alpha_1 - \delta_b$$

$$\theta_k = \alpha_k = \pi - \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2ab}\right) \quad (3)$$

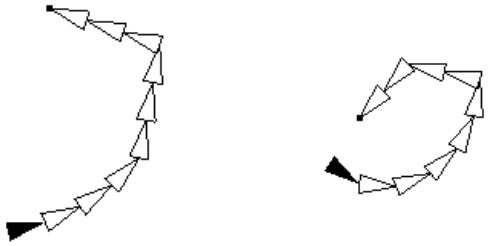
where α_i is defined as in Section II. With the above modification of the triangulation algorithm, the results previously shown in Figs 3(b), 3(c), 3(d) change to that given in Figs 6(a), 6(b), 6(c) respectively.



(a) (b) (c)

Fig. 6. The triangulation algorithm can be modified to split the chain near the midpoint.

As seen in Fig. 6, the method only produces a 2-link equivalent of the joint chain to produce a solution that is devoid of any twisting motion. Large angle rotations are still present, even though some of the unwanted "backward" movement of the chain could be eliminated. The values of θ_k in Figs 6 (a), (b), (c) are respectively 55.4 degrees, 107.21 degrees, and 162.71 degrees, and the last two configurations are generated by rotations greater than 90 degrees. Therefore we now consider joint angle constraints and try to avoid rotations that violate such constraints. This can be achieved by orienting the current link at an angle that is nearly orthogonal to the target vector, finding the middle link of the remaining chain, computing θ_k, and repeating the whole process with the next link if the value of θ_k is beyond acceptable limits. This process of "going round" a target is illustrated in Fig. 5(b). The actual angle by which we rotate each link should depend on how close or far away the target is with respect to the current link. If the target is too close to the link, we will have to start moving away from the target, and later move towards the target. In Fig. 7(a), where the target position is same as what is shown in Fig. 6(b), the link moves incrementally towards the target, till a triangulation with θ_k less than 90 degrees becomes possible. In Fig. 7(b) (which corresponds to Fig. 6(c)), the link is rotated away from the target. The joint angle constraints are met in both cases, with the maximum rotation in the first figure being 81 degrees, and in the second figure 75 degrees.



(a) (b)

Fig. 7. The proposed algorithm tries to move a link closer or away from the target, depending on its distance from the target.

The following figure (Fig. 8) explains the important parameters that need to be taken into account while forcing a joint chain to go around a target.

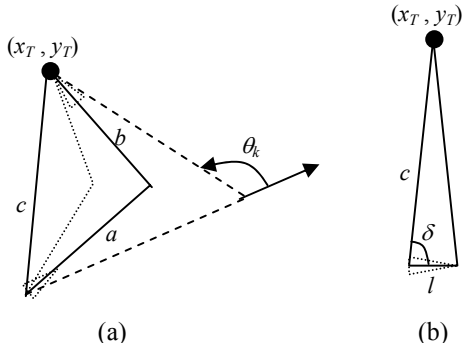


Fig. 8. Angle parameters that control joint rotations in the modified algorithm.

With reference to Fig. 8(a), the value of θ_k is greater than 90 degs for the outer dotted triangle. Obviously, the necessary condition for this to happen is

$$a^2 + b^2 - c^2 > 0 \quad (4)$$

If the above condition is satisfied, we decide to either move away or towards the target based on the target distance c. We calculate θ_k using (3), and if this angle is greater than 135 degrees in magnitude, we move away from the target, otherwise we move closer. This direction of movement is determined as follows. Referring to Fig. 8(b), the distance to the target will not change if

$$\delta = \cos^{-1}\left(\frac{l}{2c}\right) \tag{5}$$

In order to move closer to the target, we rotate the current link such that it makes an angle δ–20 degs to the target vector. To move away from the target, this angle is set to δ+20 degs. The overall algorithm for the proposed method is given below in pseudo-code form:

1: Set $i = 1$; $k = n/2$
2: $a = k*l$; $b = n*l - a$
3: Compute distance to target c from the current link i.
4: Compute α_i
5: If $(c > a + b)$ then rotate base by angle α_i; End.
6: If $a^2 + b^2 - c^2 > 0$ then
  6.1: Compute θ_k using (3)
  6.2: Compute δ using (5)
  6.3: If $\theta_k > 130°$, $\delta = \delta + 20$; else $\delta = \delta - 20$;
  6.4: $\theta_i = \alpha_i - \delta$
  6.5: Rotate i^{th} link by angle θ_i
  6.6 $n = n - 1$; $k = n/2$; $a = k*l$; $b = n*l - a$; $i = i+1$
  6.7: Compute a, b, c for the new link; Goto 6
7: Compute α_i
8: Compute δ_b using (3)
9: $\theta_i = \alpha_i - \delta_b$
10: Rotate i^{th} link by angle θ_i
11: Rotate k^{th} link by angle θ_k

V. COMPARATIVE ANALYSIS

The proposed algorithm is designed to avoid large angle rotations and twisted/self-intersecting configurations that can be produced by CCD and triangulation algorithms. By comparing the pseudo codes of the CCD algorithm in Section II and the proposed method in the previous section, three fundamental differences become obvious: (i) The CCD algorithm processes links from the end-effector towards the base, while both the triangulation and the proposed algorithms process links from the base and move towards the end-effector. (ii) The CCD algorithm uses several passes through the joint chain to converge to a solution, while the proposed method visits each node at most once to find a solution (iii) The CCD algorithm computes joint angles for every link, while the proposed algorithm rotates only those joints that are needed to move the end-effector to the target. The "single-pass" nature of the proposed method makes it a fast algorithm suitable for real-time graphics applications.

We give below a comparative analysis of the three methods using different types of cost functions:
(i) Total number of joint angle rotations performed, where only rotations greater than 5 degs in magnitude are counted.
(ii) Sum of magnitudes of all joint angle rotations performed.
(iii) Total distance traveled by the end effector.

Ten target positions were randomly generated, and the results for the above three cost functions are tabulated in Table 1, 2, 3 respectively. As shown in the various examples of the paper, the joint chain had 10 links, each of length 2 units. The initial configuration of the joint chain for all experiments was parallel to the x-axis, with the base located at the origin (0, 0), and the end effector at (20, 0).

Table I. Comparison of total number of rotations.

		Total Number of Rotations		
	Target Position	CCD	Triangulation	Proposed
1	(15.17, 4.58)	12	3	2
2	(8.33, 2.83)	9	6	10
3	(–10.58, 2.58)	49	6	4
4	(–4.08, 16.0)	35	2	2
5	(6.41, –10.08)	25	6	6
6	(–17.75, 15.75)	77	1	1
7	(3.33, –13.0)	37	4	3
8	(–1.08, 6.91)	17	8	9
9	(–8.33, 0.33)	34	2	5
10	(0.667, 11.33)	28	6	7

Table II. Comparison of sum of joint angle rotations.

		Sum of Joint Angle Rotations		
	Target Position	CCD	Triangulation	Proposed
1	(15.17, 4.58)	512.68	355.17	96.09
2	(8.33, 2.83)	668.19	879.56	299.94
3	(–10.58, 2.58)	2156.80	228.25	225.05
4	(–4.08, 16.0)	1899.72	171.66	139.43
5	(6.41, –10.08)	914.28	549.08	153.99
6	(–17.75, 15.75)	2753.46	130.41	138.41
7	(3.33, –13.0)	1238.17	346.76	119.74
8	(–1.08, 6.91)	1281.12	593.44	242.92
9	(–8.33, 0.33)	2067.28	191.73	249.20
10	(0.667, 11.33)	1350.44	465.85	174.50

Table III. Comparison of distance travelled by end-effector.

		Total Distance Travelled		
	Target Position	CCD	Triangulation	Proposed
1	(15.17, 4.58)	14.05	83.34	19.42
2	(8.33, 2.83)	14.07	192.76	46.35
3	(–10.58, 2.58)	72.81	58.27	52.98
4	(–4.08, 16.0)	100.64	44.69	34.21
5	(6.41, –10.08)	36.80	141.94	26.17
6	(–17.75, 15.75)	207.21	37.39	37.39
7	(3.33, –13.0)	57.72	94.92	25.69
8	(–1.08, 6.91)	31.22	155.56	34.33
9	(–8.33, 0.33)	53.37	39.21	56.95
10	(0.667, 11.33)	40.29	128.15	31.40

From the results presented above in Table II, it can be seen that the proposed method produces significantly less amount of joint rotations compared to other methods. This is an important cost factor to be considered for both hardware and software implementations as it directly translates to the total effort expended by joint motors. Table I shows that both triangulation method and the proposed method generate considerably less number of rotations than CCD algorithm. On an average, the number of rotations for the proposed method is slightly larger than the triangulation method, because of the additional transformations used to move around the target for certain configurations. Table III shows that the proposed method gives a shorter path for the end effector in most of the cases, wben compared with the other two methods. Figs. 9, 10 compare the shape and lengths of paths traced of the end effector for the three methods, for two different target positions.

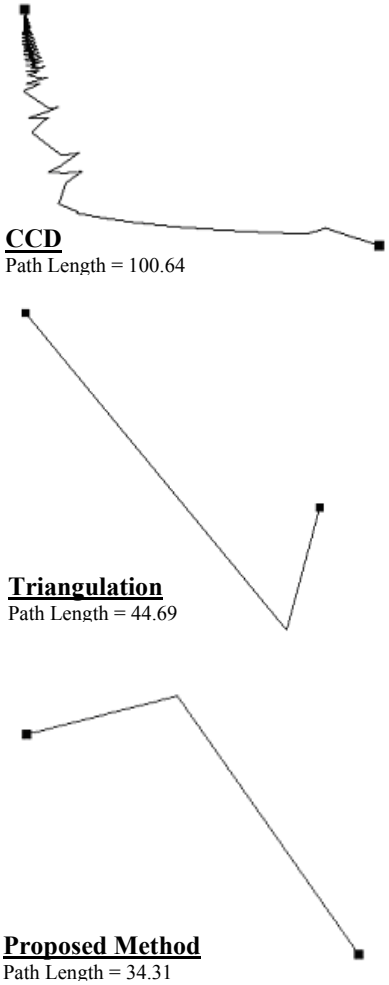


Fig. 9. Comparison of end effector traces for target position (–4.08, 16.0)

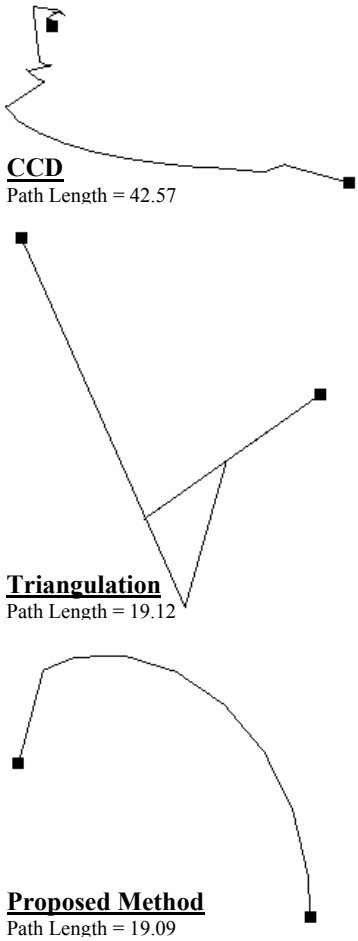


Fig. 10. Comparison of end effector traces for target position (3.06, 8.91)

VI. CONCLUDING REMARKS

This paper has discussed the inverse kinematics solution for an *n*-link joint chain and the methods used by the Cyclic Coordinate Descent algorithm and the triangulation algorithm. The main limitations of the two algorithms have been outlined. The paper then proposed an improved method similar to the triangulation algorithm, but providing a solution without large angle rotations. The proposed method can be easily implemented in real-time rendering applications, as it processes each link at most once to obtain a solution. A detailed comparative analysis has also been presented to show the benefits of the proposed algorithm over CCD and triangulation algorithm in terms of a set of cost functions.

A possible future extension of the method presented is a more general IK solution in 3D space, with quaternion rotations [8]. The solution provided by the proposed algorithm could be further optimized in terms of the cost functions, such as minimum path distance, or minimum sum of joint angles.

REFERENCES

[1] R.W. Sumner, M. Zwicker, C.Gotsman, J.Popovic, "Mesh based inverse kinematics," *ACM Trans. on Graphics*, vol. 24 (3), pp. 488-495, 2005.

[2] C.Y.Chen, M.G. Her, Y.C. Hung, M. Karkoub, "Approximating a robot inverse kinematics solution using fuzzy logic tuned by genetic algorithms," *Intnl. Jnl. of Advanced Manufacturing Technology,* vol. 20 (5), pp. 375-380, 2002.

[3] J. Zhao, N.I.Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures", *ACM Trans. on Graphics*, vol. 13, pp. 313–336, 1994.

[4] Bruderlin and Calvert, "Goal-Directed, Dynamic Animation of Human Walking", *Computer Graphics (Siggraph)*, vol. 23 (3), pp. 233-242, 1989.

[5] M. Meredith, S. Maddock, "Adapting motion capture data using weighted real-time inverse kinematics", *Computers in Entertainment*, vol. 3, pp. 5-20, 2005.

[6] J. Lander, "Making Kine more flexible", *Game Developer Magazine*, vol. 11, pp. 15-22, 1998.

[7] R. Muller-Cajar, R. Mukundan, "Triangualation – A new algorithm for inverse kinematics", *Proc. Image and Vision Computing – IVCNZ 07*, 5-7 Dec 2007, Waikato, New Zealand, pp. 181-186. 2007.

[8] Y. Aydin, S. Kucuk, "Quaternion based inverse kinematics for industrial robot manipulators with Euler wrist", *Proc. IEEE Conf. on Mechatronics*, pp. 581-586, 2006.