Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 24 (2011) 350 – 354

Procedia Engineering

www.elsevier.com/locate/procedia

2011 International Conference on Advances in Engineering

A Fast Inverse Kinematics Algorithm for Joint Animation

Wei Song [a,b], Guang Hu[c*]

[a] *Department of Computer science and technology, Wuhan University of technology,Wuhan,430070, China*
[b] *Hubei Radio & TV University, Wuhan,430074, China*
[c] *School of Resource and Environmental Science, Wuhan University,Wuhan,430072 China*
weisong@cssci.info

Abstract

The cyclic coordinate descent(CCD) is a well-known algorithm used for inverse kinematics solutions in multi-joint chains. CCD algorithm can be easily implemented, but it can take a series of iterations before converging to a solution and also generate improper joint rotations. This paper presents a novel Target Triangle algorithm that can fast decides orientation and angle of joint rotation, 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 methods.

Keywords: joint animation; cyclic coordinate descent; CCD; inverse kinematics;

1. Introduction

Currently many algorithms that solve the inverse kinematics problem use numerical and iterative procedures [1, 2, 3, 4]. The use of CCD to solve the inverse kinematics problem was first proposed by Wang and Chen [6]. They found it to be numerically stable as well as efficient. Fedor in his comparison of different algorithms found it to be a good compromise between speed and accuracy [5]. Although it can take many iterations to reach the target, each iteration is computationally cheap, which makes it possible to use this method in real time applications.

This paper presents a novel target 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 method with the proposed algorithm using different types of cost functions.

* *Corresponding author. Tel.: ; fax:.*
E-mail address:.

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2011.11.2655

The paper is organized as follows: the next section gives an overview of the CCD algorithm and outlines its drawbacks. Section 3 presents the proposed algorithm. Experimental results are presented in Section 4. Concluding remarks and possible future extensions are discussed in Section 5.

2. Cyclic coordinate descent

The cyclic coordinate descent approach has a lot of advantages over the previously mentioned IK solving approaches. The main advantage is, that CCD is easier to implement. It starts at the end-effector which is usually the last shape of a model. The algorithm measures the difference between a joint's position and the end-effectors position. It then calculates either a rotation or quaternion to reduce this difference to zero. It does this for each joint, iterating from the end-effector to the immobile joint at the root of the kinematic chain.

CCD's drawbacks are known to the graphics community. There are two typical problems associated with CCD algorithm. In the first place, CCD is an iterative method that moves joints in opposite order to their importance. Outer joints are turned firstly, which makes the movement seem unnatural, especially when this method is used for the animation of humanoid models. Secondly as the joints close to the end-effector rotate more than the joints close to the immobile joint, the kinematic chain will appear to roll in on itself.

3. Target triangle algorithm

We present an n-link chain as shown in Fig. 2, with the following notations: V_t is the target vector for the ith link , V_e is end-effector vector for the ith link , θ is the angle between vectors V_t and V_e , V_r is the rotation axis.

$$\theta = \cos^{-1}\left(V_e \bullet V_t\right) \tag{1}$$

$$V_r = \frac{V_e \times V_t}{\left|V_e \times V_t\right|} \tag{2}$$

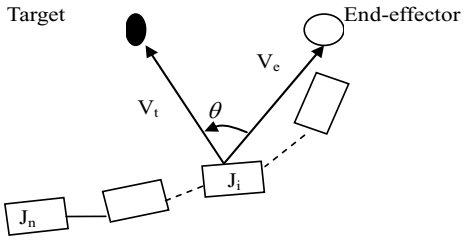

Fig. 2.An n-link joint chain

For each joint J_i we find the angle so that we can form Δabc , where a is the length of the joint we are currently moving, b is the length of the remaining chain and c is the distance from the target to the current joint.We use the properties of triangles accurately calculate the angles required to move the chain towards the target. The mathematical formula of Low of Cosines is: $c^2 = a^2 + b^2 - 2ab\cos\varphi_c$. By using the dot product of two normalized vectors to find the angle between them, we can adapt this formula to find the angles required to complete a triangle in 3-D space[7].

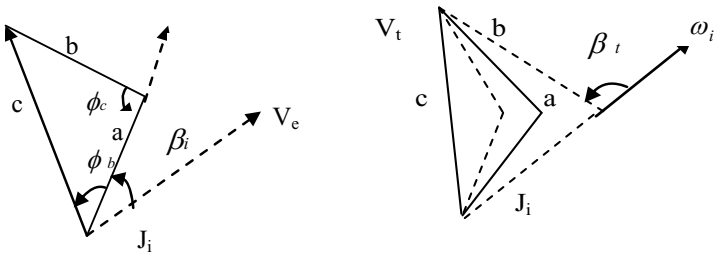

Fig. 3.(a)Target Triangle algorithm;(b) Angle parameters that control joint rotations

$$\varphi_b = \cos^{-1}\left(\frac{a^2+c^2-b^2}{2ac}\right) \tag{3}$$

$$\varphi_c = \cos^{-1}\left(\frac{a^2+b^2-c^2}{2ab}\right) \tag{4}$$

$$\beta_i = \theta - \varphi_b \tag{5}$$

We rotate each joint by β_i calculated in equation (5), about the axis of rotation V_r .In the ideal case this method will rotate only the first two joints, leaving the remainder straight. But in some complicated configurations, Large angle rotations are still present, which is generated by rotations greater than 90 degrees. Therefore we now consider joint angle constraints and try to avoid rotations that violate such constraints. With reference to 4, the value of β_t is

$$\beta_t = \pi - \varphi_c \tag{6}$$

When β_t is larger than 90° for the outer dotted triangle. Obviously, the necessary condition for this to happen is $a^2 + b^2 - c^2 > 0$ .If the above condition is satisfied, we decide to either move away or towards the target based on the target distance .

Go a step further, the Target Triangle algorithm given in takes into account the rotation limits of joint. We take a simple approximation of a human arm for example. The latter chain has the joints restricted so that joint two (the elbow)cannot rotate by more than 120° relative to joint one, and joint three (the wrist) cannot rotate by more than 90° relative to joint two. We use ω_i to notate the rotation limits of joint. We calculate β_t using (6) and if this angle is greater ω_i , we move away from the target, otherwise we move closer. In order to move closer to the target, we rotate the current link such that it makes an angle $\varphi_b = \varphi_b - 10°$ to the target vector. To move away from the target, this angle is set to $\varphi_b = \varphi_b + 10°$.

When a triangle Δabc cannot be formed, there two situations. If $c > a + b$, the target is too far away from the current joint then we cannot reach it. So the base joint should rotates from V_e to V_t , this ensures that the end-effector comes as close to the target as is possible. If $c < |a - b|$, then we cannot form a triangle with the sides a, b and c. In this case, 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.

Different from CCD, Our method Target Triangle starts from the base of the kinematic chain, which is most important for moving of the chain, process links from the base and move towards the end-effectors. The overall algorithm for the proposed method is given below in pseudo-code form:

For each Joint i do

compute V_e, V_t, V_r ,a,b,c, θ using(1)(2);

if $c > a + b$ then

J_i rotate θ ,rotation axis is V_r

end

if $c < |a - b|$ then

J_i rotate - θ ,rotation axis is V_r

End

if $a^2 + b^2 - c^2 > 0$ then

compute φ_b , β_t using(3)(5)

if $\beta_t > \omega_i$,then $\varphi_b = \varphi_b - 10°$ else $\varphi_b = \varphi_b + 10°$

compute $\beta_i = \theta - \varphi_b$

J_i rotate β_i ,rotation axis is V_r

4. Comparing CCD and target triangle

In this section we run a small experiment to compare two inverse kinematics algorithms with each other. The compared algorithms are CCD and our new method called Target Triangle. We created a simple OpenGL application in C++, which animates a simple kinematic chain with one end-effector. The chain has a length of 40 units, split into four joints of length nine, and one joint of length four. ω_2 =120°, ω_3 =90°.It means that he latter chain has the joints restricted so that joint two cannot rotate by more than 120° relative to joint one, and joint three cannot rotate by more than 90° relative to joint two. This is a simple approximation of a human arm. Both chains as shown by the application can be seen in figure 5.

As can be seen in figure 5, Target Triangle achieved more natural poses when reaching for the same target as CCD did. This is caused by the major joints rotating before the joints close to the end-effector. It is also worth noticing that most joints in figure4 simply remain straight, instead of twisting in several different directions . This had an effect on the cost function, which was lower for the Target Triangle method than for CCD using both kinematic chains.

We can compare the complexity of the two algorithms using the number of calculations required for each joint. The dot product of two vectors in three-dimensional space requires three multiplications to compute. The cross product needs six multiplications. Solving the cosine equation as shown in equation (3) (4) needs 4 multiplications and one division. This is approximately as complicated as a cross product. We define the number of calculations required for the rotation or transformation as r. We define the number of iteration as IterNum. We define the number of joints as JNum. We can represent these values by the following equations:

$$N_{ccd} = IterNum * JNum * (n + r)$$

$$N_{T\arg etTriangle} = JNum * (n + 2n + r)$$

The CCD algorithm uses several passes through the joint chain to converge to a solution, while the Target Triangle method visits each node at most once to find a solution. Clearly $N_{ccd} > N_{T\arg etTriangle}$ when CCD requires more than one iteration.

Fig.4. (a)5-Jointed kinematic chains using Target Triangle;(b) 5-Jointed kinematic chains using CCD

5. 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. The main limitations of the CCD algorithms have been outlined. The paper then proposed an Target Triangle algorithm, 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 algorithm in a small experiment.

A possible future extension of the method presented is a more general IK solution in 3D space, with quaternion rotations. 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] K. C. Gupta and K. Kazerounian, “Improved numerical solutions of inverse kinematics of robots,” in Proceedings of the 1985 IEEE International Conference on Robotics and Automation, vol. 2, March 1985,p. 743–748.

[2] A. Goldenberg, B. Benhabib, and R. Fenton, “A complete generalized solution to the inverse kinematics of robots,” IEEE Journal of Robotics and Automation, vol. 1, March 1985, p. 14–20.

[3] V. J. Lumelsky, “Iterative coordinate transformationprocedure for one class of robots,” IEEE Transactions on Systems Man and Cybernetics, vol. 14, Jun. 1984,p. 500–505.

[4] V. D. Tourassis and J. Ang, M.H., “A modular architecture for inverse robot kinematics,” IEEE Transactions on Robotics and Automation, vol. 5, October 1989,p. 555–568.

[5] M. Fedor, “Application of inverse kinematics for skeleton manipulation in real-time,” in SCCG ’03: Proceedings of the 19th spring conference on Computer graphics. New York, NY, USA: ACM Press, 2003 ,p. 203–212.

[6] D. L. Pieper, “The kinematics of manipulators under computer control,” Ph.D. dissertation, Stanford University Department of Computer Science, October 1968.

[7] R. Mukundan .A robust inverse kinematics algorithm for animating a joint chain. Computer Applications in Technology,Vol. 34, No. 4, 2009,p. 303-308 .