Visualization and Analysis of Inverse Kinematics Algorithms Using Performance Metric Maps

Oliver Cardwell, Ramakrishnan Mukundan
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch
New Zealand
orc13@student.canterbury.ac.nz, mukundan@canterbury.ac.nz

ABSTRACT

Iterative inverse kinematics (IK) algorithms are commonly used in graphics animations involving goal-directed motion of joint chains and articulated character models. A well-known algorithm is the Cyclic Coordinate Descent. For certain joint chain configurations and target positions, iterative methods can generate undesirable joint rotations. Similarly, certain target positions may require large number of iterations, or may not even be reachable. This paper presents a novel concept called performance metric maps as a tool for visualizing and analysing the performance characteristics of an iterative IK algorithm under parametric variations. The proposed method is particularly useful in determining how well an algorithm converges within a given region of the workspace. The paper presents the visualization aspects of the metric maps, and the results of comparative performance analysis of two IK algorithms.

Keywords

Inverse kinematics algorithms, Cyclic coordinate descent, Goal-directed motion, Articulated character animation, Performance metric maps

1 INTRODUCTION

Animation of articulated character models and the goal-directed motion of serial joint chains often require inverse kinematics (IK) algorithms that provide a converging solution for both joint angles and the target position [1],[6]. Cyclic Coordinate Descent (CCD) is a well-known iterative algorithm used in computer graphics and animation [3]. Even though the algorithm is conceptually simple and easy to implement, certain target positions may require a large number of iterations before an acceptable solution is obtained. Similar algorithms have been recently proposed either to improve the convergence of the solution, or to eliminate problems associated with large angle rotations [4],[5].

The performance analysis of such methods will have to take into account several factors that affect the parameterization of the joint chain in terms of angles, such as number of joints, link lengths, and joint angle constraints.

Several types of metrics can be defined to evaluate the performance of an iterative IK algorithm. Some of these are outlined in [4]. However, the pattern of variation of these metrics changes with the configuration of the joint chain. Given two target positions in the work space, it is often difficult to predict the value of the metric at an intermediate point. Metrics such as the minimum number of iterations, distance traveled by the end-effector, etc., do not have a linear relationship to changes in target positions.

This paper proposes a novel method for representing the values of performance metrics on a discretized pixel coordinate space that is mapped to the joint chain's workspace. The map not only provides an exhaustive set of values of a performance metric at all reachable points, but also gives an image-based visualization of its variation within the two-dimensional workspace. Here we assume that every joint other than the root has a

single degree-of-freedom given by a relative angle of rotation about a fixed axis. We also assume that motion to an arbitrary point in three-dimensional space can be considered as a combination of (i) a rotation of the chain about the root so that base, end-effector and the target lie on a plane, followed by (ii) a solution of the 2D IK problem for the chain configuration and target position on that plane. Thus a two-dimensional performance metric map is adequate for the analysis of most of the iterative IK algorithms used in computer animation. This paper gives a comparative analysis of iterative IK algorithms using performance metric maps, and shows the effectiveness of the method in analyzing the workspace characteristics of a given algorithm in terms of configuration dependent parameters.

The paper is organized as follows. The next section gives a general overview of IK structures that we will be dealing with in this paper. Section 3 looks at the CCD algorithm and also introduces a new IK algorithm that finds a solution where all joints are placed along a circular arc. Section 4 introduces the concept of performance metric maps. Section 5 gives a comparative analysis of the CCD and the circular algorithms and presents experimental results. Section 6 concludes the paper and outlines future research directions.

2 IK STRUCTURES

Common IK structures used in robotics and animation are articulated bodies. An articulated body is simply a list of joints linked end to end; forming a joint chain. Each joint in the chain has a length and an angle offset from its parent joint. One end of the joint chain is the base, and is fixed in some frame of reference, and the other end is the end-effector. In the case of a robotic arm the end-effector would be the manipulator or hand (Fig. 1). A joint chain can be described as a list of

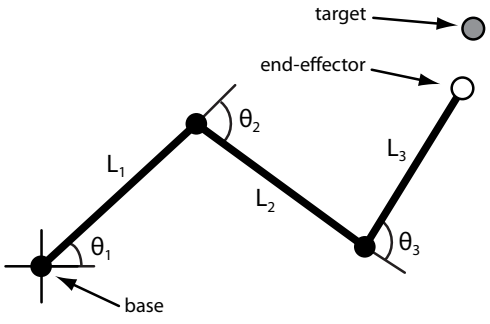

Figure 1: Joint Chain with length $n = 3$

lengths $\mathbf{l}$ and a corresponding list of angles $\mathbf{q}$ such that

$$\begin{aligned} \mathbf{l} &= [L_1, L_2, \ldots, L_n] \\ \mathbf{q} &= [\theta_1, \theta_2, \ldots, \theta_n] \end{aligned} \tag{1}$$

The state of a joint chain is given by $\mathbf{q}$. To perform forward kinematics on a joint chain, that is given $\mathbf{l}$, $\mathbf{q}$ and the base coordinates $\mathbf{b}$, find the location of the end-effector $\mathbf{e}$ then it follows that

$$\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} + \sum_i^n \begin{bmatrix} L_i cos\theta_i \\ L_i sin\theta_i \end{bmatrix} \tag{2}$$

However, to perform inverse kinematics on a joint chain, a function is needed that takes the desired location of the end-effector $\mathbf{e}$ and computes a valid state $\mathbf{q}$. Finding a valid state for a given joint chain configuration can be a difficult process and there are numerous methods of computing valid states.

3 ITERATIVE IK ALGORITHMS

There are a number of common parameters that are given to an inverse IK algorithm. These parameters are

n The number of joints in the chain

i The maximum number of iterations

ε The threshold distance to which the end-effector can be considered at the target location

Therefore, a typical parameter iterative IK algorithm will be passed values for n, i and ε along with the joint chain.

Cyclic Coordinate Descent

The CCD algorithm uses a heuristic approach to find a solution by iteratively rotating the links so that the end-effector moves closer to the target. Each iteration performs a sequence of rotations of links i, starting from the end-effector towards the root, trying to minimize the angle θ_i between the vector from the link joint towards the end-effector and the vector towards the target [2] (Fig. 2).

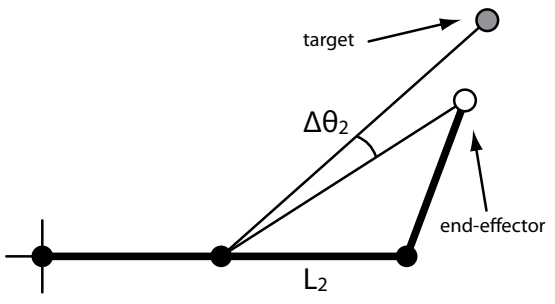

Figure 2: Joint angle rotations in a CCD algorithm

Joint angle rotations for a CCD algorithm can be easily computed and implemented in graphics applications. However, the method suffers from primarily

three types of problems: (i) certain target positions within the workspace require a large number of iterations, (ii) a solution may involve large angle rotations, and (iii) target positions near the base of the chain may result in self-intersecting configurations. Examples of these cases are shown in Fig. 3.

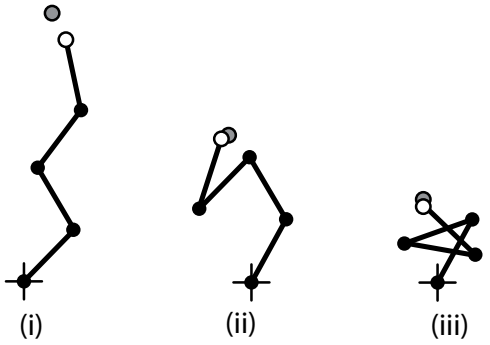

Figure 3: Limitations of the CCD algorithm

Because the CCD algorithm must visit each joint in the chain for each iteration it can be shown that the CCD algorithm has a computational complexity of $O(n)$ for each iteration.

Circular Alignment Algorithm

A few methods have been recently proposed ([4],[5]) to circumvent the limitations of the CCD algorithm. In this section, we propose a new algorithm that can be considered as a further improvement of the method proposed in [5].

If d denotes the distance of the target $\mathbf{t}$ from the base of a joint chain, then there exists a unique circumscribing circle with radius r along which all nodes can be positioned (Fig. 4).

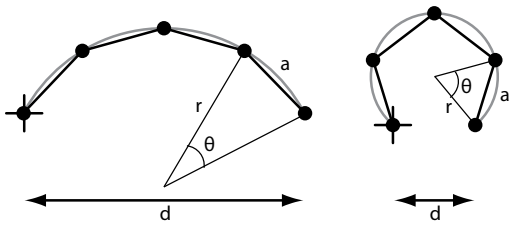

Figure 4: Circular alignment of joints

In the following, we make the assumption that all joints have the same length. If there are n joints in the chain then the chain contains $n+1$ nodes. If θ is the segment angle of a joint of length a in the circumscribing circle with radius r then it follows that

$$a = 2rsin\left(\frac{\theta}{2}\right) \tag{3}$$

And therefore we can define the relationship between lengths a and d as

$$\frac{sin\left(\frac{\theta}{2}\right)}{sin\left(\frac{n\theta}{2}\right)} = \frac{a}{d} \tag{4}$$

We seek the solution of the above equation for θ, by defining the function

$$f(\theta) = dsin\left(\frac{\theta}{2}\right) - asin\left(\frac{n\theta}{2}\right) \tag{5}$$

with the derivative

$$f'(\theta) = \frac{d}{2}cos\left(\frac{\theta}{2}\right) - \frac{na}{2}cos\left(\frac{n\theta}{2}\right) \tag{6}$$

The solution for θ is obtained using Newton-Raphson iteration

$$\theta_{i+1} = \theta_i - \frac{f(\theta_i)}{f'(\theta_i)} \tag{7}$$

with initial condition

$$\theta_0 = \frac{2\pi}{n} \tag{8}$$

This givens the initial configuration that the joint chain is curled around so the end-effector is at the base, such that the length $d = 0$. As the angle θ approaches 0 the chain opens up until d is within the threshold distance ε of the target t.

When the joints are aligned along a circular path, the joint angles will automatically assume values in an acceptable range, and there is no possibility of the chain intersecting itself. The Newton-Raphson method yields fast convergence for the parameter θ, from which the joint angles that are all equal, can be computed.

Because the Circular Alignment Algorithm (CAA) method does not need to visit each joint during an iteration it can be shown that the CAA method has a computational complexity of $O(1)$ for each iteration.

4 PERFORMANCE METRIC MAPS

Metric maps allow easy visualization and analysis of otherwise complex or dense data. A common example of metric maps are terrain or elevation maps. In these metric maps the terrain height at any point on the map is represented as a color shade. They also often include contour lines at designated elevation intervals to help us visualize the layout of the terrain represented by the map.

This basic principal can be generalized to display many other types of data. We will show how using metric maps can greatly simplify the visualization and analysis of the behavior of various iterative inverse kinematic algorithms.

In order to generate a performance metric map of the workspace of a given joint chain we need to encapsulate the whole workspace in an image. Because the workspace is a circle we need a square image and if we set the base of the chain to be the center of the image then the length of the joint chain becomes the radius. An example of this can be seen in Fig. 5.

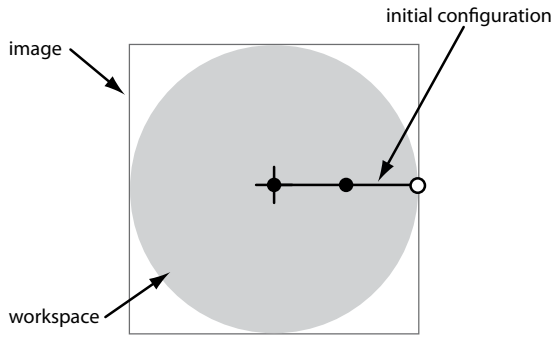

Figure 5: Joint Chain Workspace

From this initial configuration we can now iterate over each pixel in the image using the pixel's coordinates relative to the image center as the target point for the joint chain. For each of these targets we simply run an iterative inverse kinematic algorithm over the joint chain and record the desired metric, for example, the number of iterations taken to move the end-effector to within ε of each pixel can be seen in Fig. 6. Where black indicates that the algorithm failed to reach that pixel, red indicates that algorithm successfully reached the pixel on the last attempt and in hue scale down to blue which indicates that the pixel was reached in a single iteration.

It is interesting to note in Fig. 6 the dark blue (single iteration) region separates the remaining iso-surfaces into two disjoint regions. These properties and patterns cannot be analytically derived but by using metric maps these properties can be observed.

Other metrics can also be measured using the same method. An example of plotting the distance traveled by the end-effector can be seen in Fig. 7.

5 COMPARATIVE ANALYSIS

Using metric maps to help visualize the behavior of a iterative IK algorithm on a joint chain gives a much clearer picture of problem areas. In Fig. 6 it can been seen that the CCD method fails to place the end-effector at target locations in and around the initial end-effector location. The existence of this large void is somewhat unintuitive but can be clearly seen using a metric map. It is also can been seen that CCD algorithm in general requires more iterations to reach targets further from

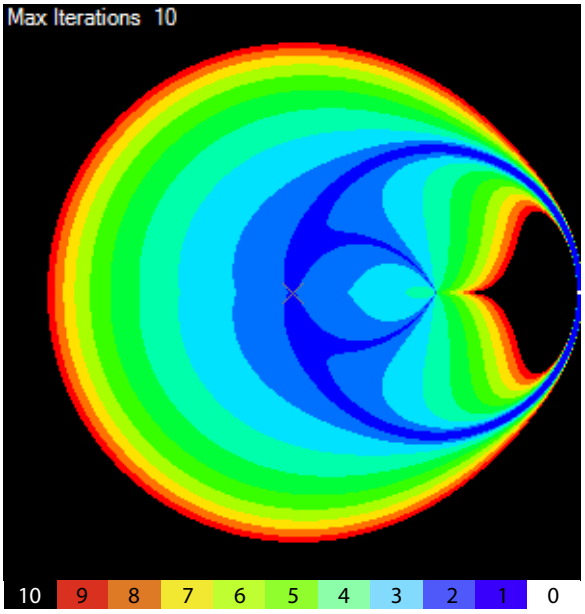

Figure 6: Metric Map of Iterations ($n = 2, i = 10, \varepsilon = 0.01$)

the base of the chain but covers most of the possible workspace.

However, as we increase the number of joints in the chain and plot their corresponding metric maps a number of interesting properties can be observed (Fig. 8). The observed structure of the metric map becomes more complex and the inclusion of more voids is evident. A interesting observation that can be made using metric maps is that the workspace coverage of the joint chain diminishes as the number of joints increases. Intuitively it could be thought that increasing the number of joints in the chain would allow the chain more freedom to move the end-effector to the target location and thus increase the workspace coverage. However it appears that adding more than three joints to a chain decreases the chain's coverage. This can been seen in Fig. 9.

An interesting optimization can be done to the workspace coverage by altering the lengths of the joints in the chain. If the lengths of the joints are set so that, starting from the end-effector, each joint is the equal to the sum of lengths before it, then the coverage is largely unaffected by the increase in the number of joints (Fig. 10). For example, if we have a joint chain where $n = 6$ then the the length ratios are $\mathbf{l} = [16, 8, 4, 2, 1, 1]$.

Using metric maps we can compare the two algorithms, CCD and CAA for iterations and workspace coverage (Fig. 12).

We can clearly see through the use of metric maps the

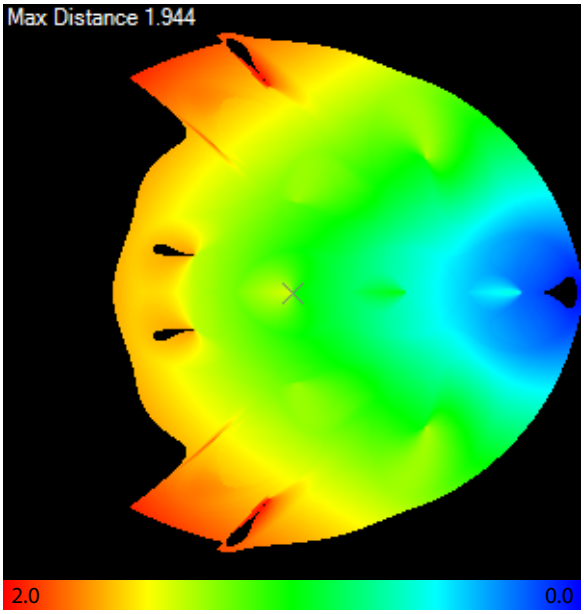

Figure 7: Metric Map of Distance Traveled ($n = 5, i = 10, \varepsilon = 0.01$)

differences in behavior between the two algorithms. In this example the CCD method has 67% workspace coverage and clearly uses up to the maximum number of iterations, colored red, to reach various target positions.

However, with exactly the same configuration we can see that the CAA method has 100% coverage and at no position needs to use up to the maximum number of iterations. In fact, the CAA method only uses a maximum of 5 iterations to reach every target point in the workspace. On closer evaluation of the CAA method we can see that the number of iterations required to completely cover the workspace is solely dependent on the size of the threshold ε. A comparison of the computational efficiency also shows that the CCD has an $O(n)$ computational complexity while the CAA operates in $O(1)$. An example of this can be seen in Figure 11.

6 CONCLUSION AND FUTURE WORK

We have shown how the use of metric maps can aid greatly in visualizing and analyzing the behavior of inverse kinematic problems. By generating metric maps for a popular iterative IK method, CCD, we were able to easily discover and identify the algorithm's various properties, including helping to formulate a new method, CAA, to address some of the negative performance aspects of the CCD method.

Although generating these metric maps can be somewhat computationally expensive, as essentially they are an exhaustive search of workspace, the generation lends itself well to parallel computational techniques. Even without parallel techniques our implementation generated 300 by 300 sample images in only a couple of seconds (see figure 11).

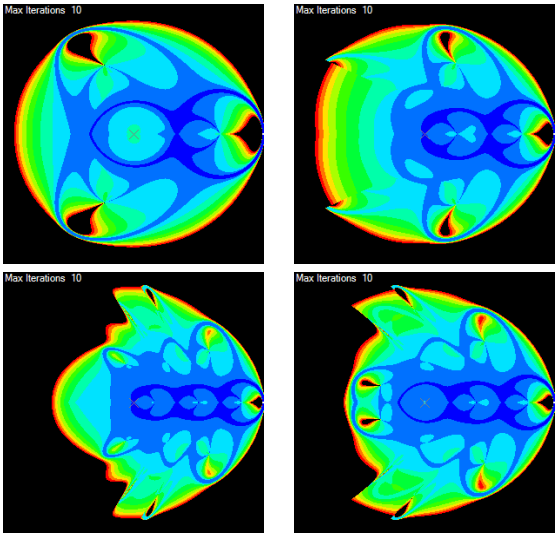

Figure 8: Clockwise from top left, $n = 3, n = 4, n = 5$ and $n = 6$

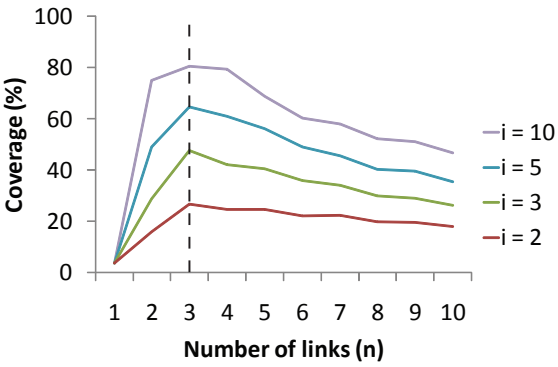

Figure 9: CCD coverage with joints of equal length ($\varepsilon = 0.01$)

The CCA method shows promise as a high performance iterative but relies on some important assumptions that may not be practical in some situations. We would like to expand the CAA method to be able to incorporate chains with joints of different lengths.

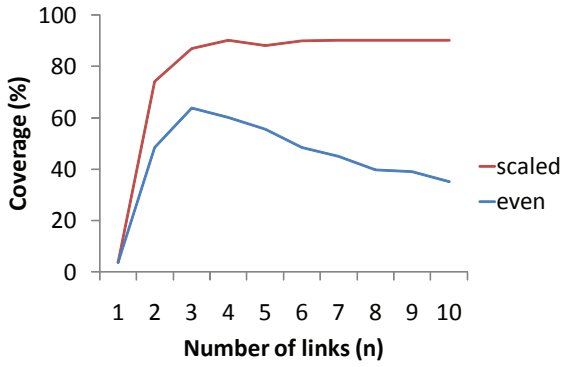

Figure 10: CCD coverage ($i = 5, \varepsilon = 0.01$)

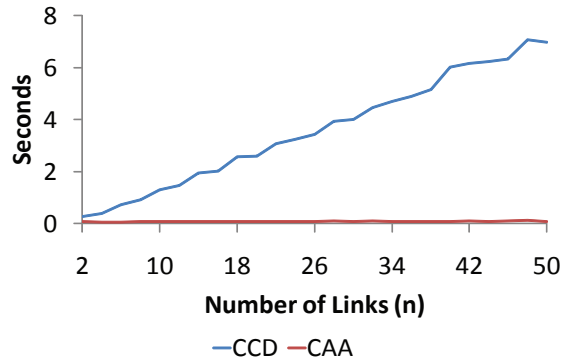

Figure 11: Performance (4x Multi-threaded) for a 300x300 metric map ($i = 20, \varepsilon = 0.01$)

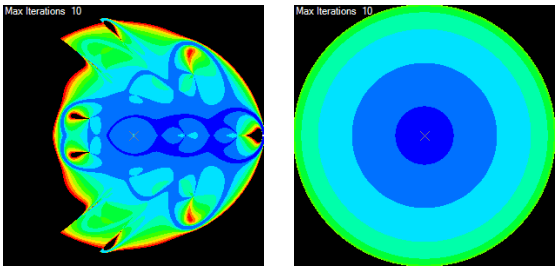

Figure 12: Comparison of CCD (left) and CAA (right) with $n = 5, i = 10, \varepsilon = 0.01$

References

[1] M. Engell-Nørregard. Inverse Kinematics The state of the art. 2007.

[2] G.Z. Grudic and P.D. Lawrence. Iterative inverse kinematics with manipulator configuration control. *IEEE Transactions on Robotics and Automation*, 9(4):476–483, 1993.

[3] J. Lander. Making kine more flexible. *Game Developer Magazine*, 11:15–22, 1998.

[4] R. Mukundan. A robust inverse kinematics algorithm for animating a joint chain. *International Journal of Computer Applications in Technology*, 34(4):303–308, 2009.

[5] R. Muller-Cajar and R. Mukundan. Triangualation-A New Algorithm for Inverse Kinematics. 2007.

[6] C. Welinan. *Inverse kinematics and geometric constraints for articulated figure manipulation*. PhD thesis, Simon Fraser University, 1993.