ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/266135706

Application of the Cyclic Coordinate Descent algorithm in the protein loop closure problem

Article · September 2011

DOI: 10.13140/2.1.1898.6249

CITATIONS
0

READS
326

1 author:

Andrzej Wójtowicz
Adam Mickiewicz University
20 PUBLICATIONS **120** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

OvaExpert: An intelligent medical diagnosis support system for ovarian tumor View project

All content following this page was uploaded by Andrzej Wójtowicz on 26 September 2014.

The user has requested enhancement of the downloaded file.

Application of the Cyclic Coordinate Descent algorithm in the protein loop closure problem

Andrzej Wójtowicz
andre@wmi.amu.edu.pl

September 21, 2011

Abstract

In 2003 Adrian Canutescu and Roland Dunbrack Jr. presented an algorithm used in protein structure prediction. In this case one protein element has to be fitted to a fixed fragment. Previous approaches relied on using Jacobian but they have two disadvantages: large computational overhead and need of matrix inversion. Fortunately protein loop closure and inverse kinematics problem are very similar. In robotics and computer animation the Cyclic Coordinate Descent (CCD) method is used to determine, accordingly, th movement of a robotic arm and animation of characters. In contrast to inverse Jacobian, CCD is extremely fast and has not the above-mentioned constraints. An appropriate part of a protein may be considered as a robotic arm which is to e.g. catch a desired object, so we can benefit from CCD. This article presents the idea of the CCD algorithm.

1 Forward and inverse kinematics

Before we focus on the biological aspect, let's have a look at the origin of the problem in robotics. Consider a simple robotic arm in a 2D world presented on the picture below.

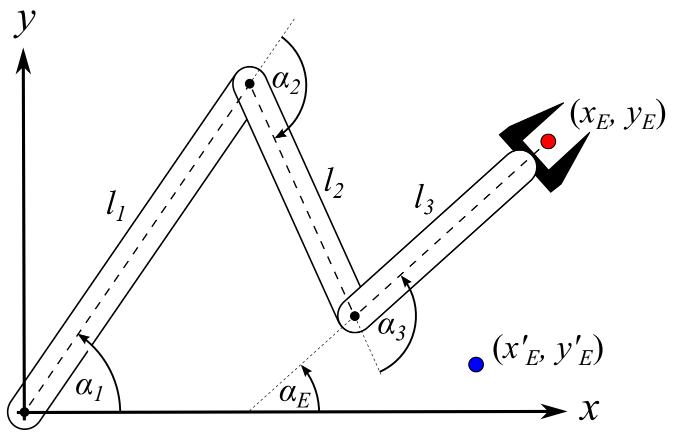

Figure 1: An example configuration of the robotic arm. The red joint indicates, simplifying, the position of the end effector. The blue one is the object which the arm have to grab.

The arm consists of three rotational joints, three links of length l_1, l_2, l_3 and the end effector. At the joints we have angles α_1, α_2 and α_3.

Imagine a human operator who manually changes the initial configuration of the arm. The control unit must know where is the end effector. The **forward kinematics** problem tries to answer the question: *what is the current location of the tip of the arm?* In other words, on the basis of the lengths of the links and the angles at the joints, we try to determine the position of the end effector (x_E, y_E) and its direction α_E.

However, the problem might be turned upside-down: say we have an object with known position near the arm and want to catch it. The question is: *how should the arm move to be able to grab the object?* And this is the **inverse kinematics** problem. We can state it more formal: given the fixed position (x'_E, y'_E), what should be the angles at the joints[1] to set the end effector at that point?

One solution to the inverse kinematics problem are algorithms based on Jacobian but because of matrix inversion in these methods[2] calculations use to fail. Moreover, this approach has a large computational overhead. In some special cases, the solution might be derived analytically but the problem grows with more links in the arm and extension to the third dimension. Notice that in a 3D world a joint may rotate the adjacent link in additional directions (this ability is called degrees of freedom).

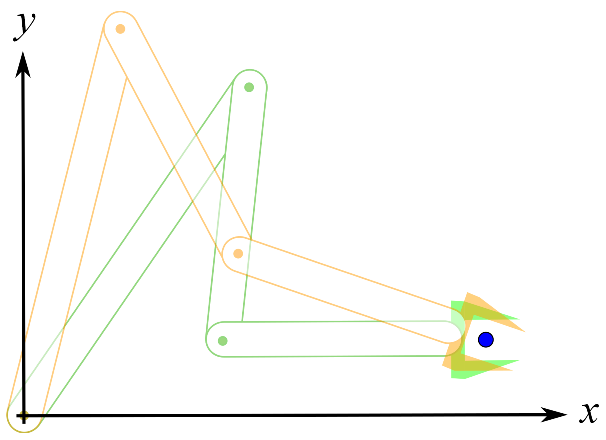

Figure 2: Often there are many solutions to the problem of inverse kinematics.

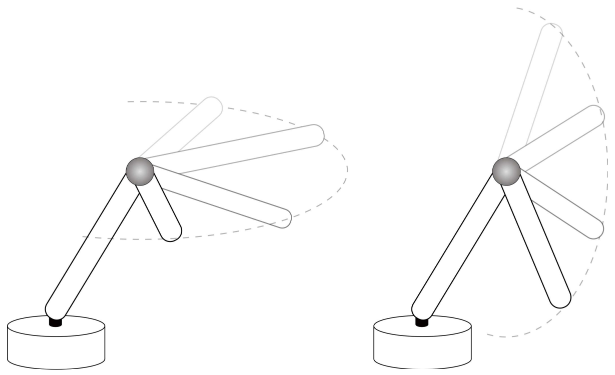

Figure 3: In three-dimensional space the arm has more possibilities to move.

[1]Of course we also know the lengths of the links with initial positions.

[2]Precisely, in case of singular matrix.

2 Cyclic Coordinate Descent

Fortunately there exists a more effective way to deal with the inverse kinematics problem. The CCD algorithm works as follows: iteratively go through the joints, each time find the best rotation angle which brings the end effector the nearest to the goal, and apply the rotation to the next joints. The algorithm stops when the end effector is close enough to the goal or the number of iterations exceeds the limit.

The two examples below show steps of the algorithm in, accordingly, two- and three-dimensional space.

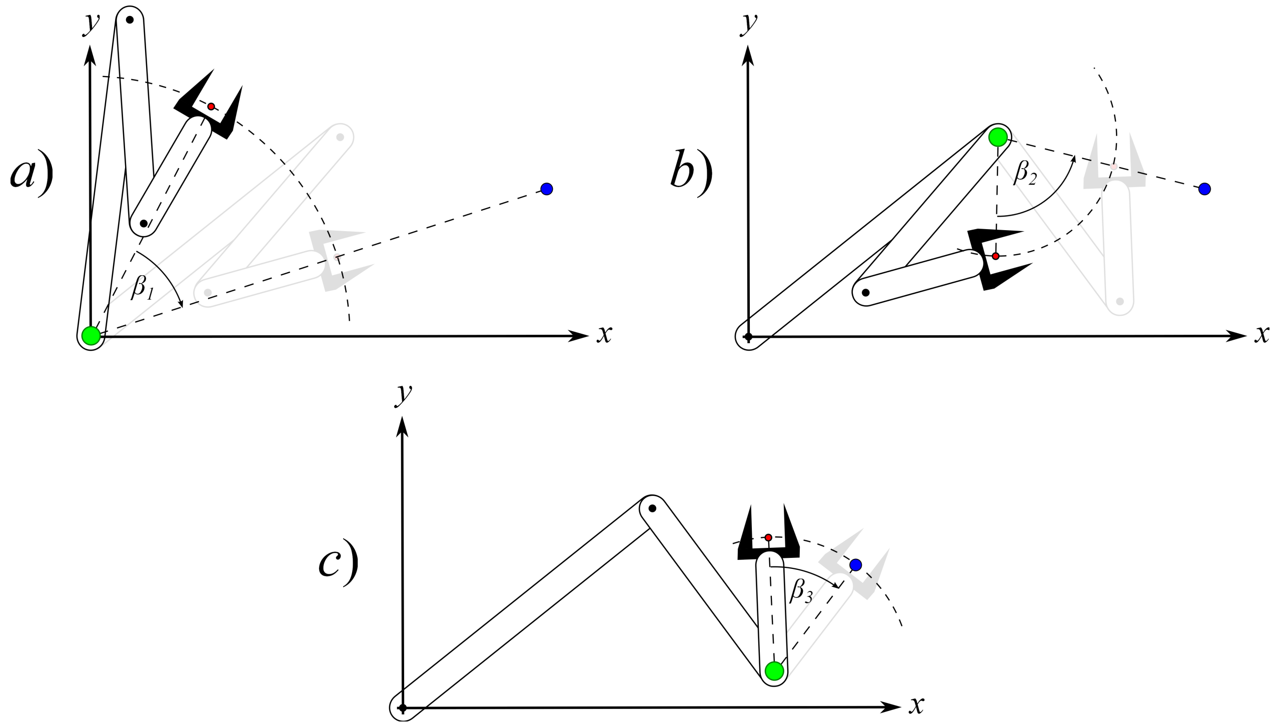

Figure 4: The green joint indicates the rotation in the current step, the blue point is the goal. This time the algorithm succeeds in the first iteration. Application of the β_1, β_2 and β_3 angles to the initial configuration leads the end effector to the goal.

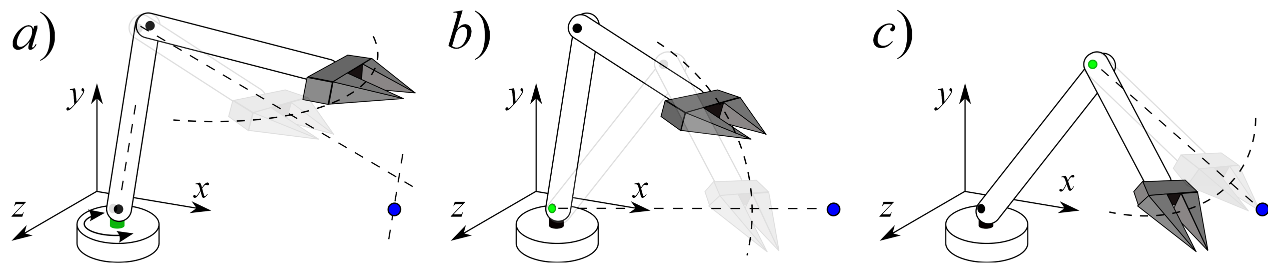

Figure 5: In the step a) the goal must be projected into a plane of rotation. In the step c) the arm almost reached the goal so th next iteration must be performed.

3 Loop closure problem and CCD

The clue is that atoms of the protein between N-terminus and C-terminus may be treated as joints and chemical bonds as links in a robotic arm. However, there are two slight differences in determining the proper rotation angle.

Firstly, the aim is to move three C-terminus atoms: N, C_α and C; near fixed atoms. To enumerate the proper angle in the iteration we minimise the sum of squared distances between the above-mentioned C-terminus moving and fixed atoms. The algorithm stops when the root mean square deviation (RMSD) of these atoms is less than 0.08Å.

Secondly, flat angles must be constant so that only the rotation around the axis between neighbouring atoms is allowed.

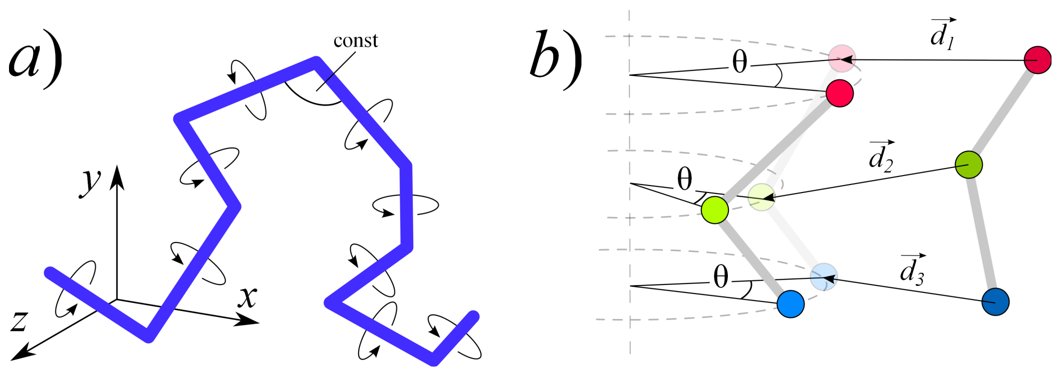

Figure 6: a) shows possible changes in the arrangement. In b) on the left are moving C-terminal atoms, on the right fixed atoms. The goal is to set moving atoms so that the sum of squared distances $\vec{d_i}$ reaches a minimum (in other words, to find the best θ). Detailed images and tricky calculations obtaining θ are (more or less explained) in [1].

4 Further problems and ideas

Consideration should be given to the some special cases in the arrangement of atoms. At the very beginning the algorithm should check whether the goal is within range of the moving arm. This may be done by setting all moving atoms in one plane and then calculating the radius of such an arm.

More complications arise when the algorithm finds a local minimum. In other words, it may happen that in the current phase of an iteration the distance to a goal should be increased. To deal with this problem, the algorithm should restart with random angles.

Fortunately such problems appear rarely in the bioinformatics application. The authors tested the algorithm on real arrangements and it was successful in 99.79% of cases. What's more, use of a Ramachandran map, which eliminates unrealistic rotations, increases the number of correct loop closures to only 0.01%.

It is worth to know that Wouter Boomsma and Thomas Hamelryck presented a variation of the CCD called Full CCD (FCCD). In this case torsion angles may be changed but it solves the loop closure problem for C_α only protein models. The algorithm is quite simple in its implementation but less intuitive because of the singular value decomposition (SVD) and operations on rotation matrices.

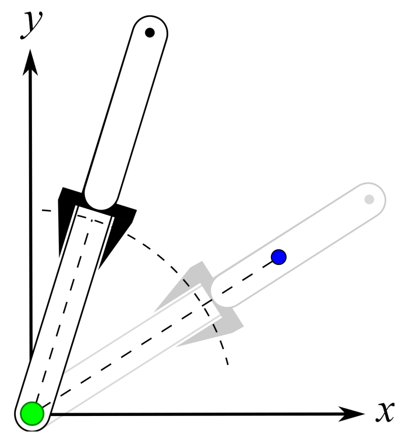

Figure 7: A textbook example of the local minimum problem. The first step causes that the end effector can't near to the goal.

References

[1] A. Canutescu, R. Dunbrack Jr.: **Cyclic coordinate descent: A robotics algorithm for protein loop closure**
available at http://onlinelibrary.wiley.com/doi/10.1110/ps.0242703/pdf

[2] Lydia E.Kavraki: **Protein Inverse Kinematics and the Loop Closure Problem**
available at http://cnx.org/content/m11613/1.12/?format=pdf

[3] C. Hecker: **My Adventures in Inverse Kinematics** at Game Developers Conference in 2002
available at http://chrishecker.com/images/7/76/Gdc2002-ik.ppt

[4] http://www.learnaboutrobots.com/inverseKinematics.htm

View publication stats