Computing Inverse Kinematics with Linear Programming

Edmond S.L. Ho Taku Komura Rynson W.H. Lau

Department of Computer Science
City University of Hong, Hong Kong

ABSTRACT

Inverse Kinematics (IK) is a popular technique for synthesizing motions of virtual characters. In this paper, we propose a Linear Programming based IK solver (LPIK) for interactive control of arbitrary multibody structures. There are several advantages of using LPIK. First, inequality constraints can be handled, and therefore the ranges of the DOFs and collisions of the body with other obstacles can be handled easily. Second, the performance of LPIK is comparable or sometimes better than the IK method based on Lagrange multipliers, which is known as the best IK solver today. The computation time by LPIK increases only linearly proportional to the number of constraints or DOFs. Hence, LPIK is a suitable approach for controlling articulated systems with large DOFs and constraints for real-time applications.

1. INTRODUCTION

Inverse kinematics (IK) has been widely used for synthesizing motions of linked bodies in robotics and computer animation. Among the numerical methods of IK, the method based on Lagrange multipliers is known to be the best, as its computational cost increases only linearly with the number of DOFs, and the motions generated are natural. The problem with this method is that it cannot handle inequality constraints. Inequality constraints are necessary when solving problems for multibody structures that has joints with limitations in the ranges of motion, or collisions between different segments. The other problem is that its computation time grows cube proportional to the number of constraints. Hence, when controlling multiple characters under multiple constraints, the performance drops significantly.

In this paper, we propose a Linear Programming based Inverse Kinematics solver (LPIK) for interactive control of arbitrary multibody structures. Instead of calculating the least squares solution, a criterion based on minimizing the sum of absolute values using linear programming is proposed. We have evaluated the performance of LPIK. Comparing to the least squares method, which uses pseudo-inverse matrix or quadratic programming to minimize the criterion, computation time of LPIK is significantly lower. Comparing to the Lagrange multiplier's method, the cost is comparable, or even better, when the number of constraints is large.

The contributions of this paper can be summarized as follows. First, we propose an efficient IK solver based on linear programming. We also propose an objective function to reduce jittering. Second, through conducting a number of experiments and analyzing the results, we show that the computational cost of LPIK is $O(mn)$, where m is the number of constraints and n is the DOFs. This is comparable to the Lagrange multiplier's method, which cost is $O(n + m^3)$.

2. RELATED WORK

IK is an old problem in robotics for controlling robot manipulators. Researchers in computer graphics have been using it to generate animation of multi-segment characters such as animals and human figures. Its applications include editing keyframe postures, generating interactive animation, and generating motion such as reaching and walking.

Due to the popularity of motion capture systems, many researchers have started to work on topics such as editing and synthesizing motions using Mocap data. Several techniques have been proposed to combine such concepts with IK so that the animators or users can edit or control human characters interactively based on some captured motion data. Grochow *et al.* [3] propose an IK system that utilizes captured motion data to solve the redundancy problem. Kovar *et al.* [4] and Rose *et al.* [9] propose example-based IK solvers, which may solve the IK problems efficiently by searching and interpolating the appropriate motion. These new methodologies utilizing the Mocap data still need to use the old fashion IK solvers when adjusting the positions of the segments, especially when the data space is not dense enough. Therefore, all newly developed methods will benefit if the computational cost of the IK solvers can be reduced.

The IK solvers can be roughly divided into two main categories: *analytical solvers* and *numerical solvers*. Analytical solvers provide explicit solutions to calculate the generalized coordinates from the position information. Their advantage is that a solution can be obtained in a very short time. For example, Lee *et al.* [5] propose an analytical solution to determine the posture based on the positions of the hands and feet relative to the positions of the shoulders and hips. The method shows good performance in motion editing. However, analytical solvers must be pre-tuned for individual systems, and there is no general method to create such solvers for arbitrary chain structures. Hence, their applications are limited to objects with simple structure.

On the other hand, numerical solvers linearize the relationship of the generalized coordinates and the 3D coordinates of the end effectors around the current posture to obtain the IK solutions for new 3D coordinates of the end effectors close to the current position/orientation. Numerical solvers have three main advantages. First, they can be applied to arbitrary chain structures. Second, various types of constraints, such as positional or planar constraints, can be handled in the same platform. Third, the constraints can be easily switched on and off. Hence, more numerical solvers have been developed. The most practical and commonly used numerical solver is based on the least squares methods [11]. Since the original least squares method becomes unstable near singular points, various methodologies such as SR inverse [7] have been developed to stabilize the system near such singular postures. The bottleneck of these methods are the cost of computing the pseudo-inverse matrix, which grows cube-proportional to the number of constraints. Baraff [1] proposes a method of forward dynamics for articulated body structures, which can be used for solving the IK problems. Instead of calculating the pseudo inverse matrix, an equation of Lagrange multipliers is solved. Since the matrix used in his method is sparse, efficient solvers for sparse matrix can be used. However, the method can only handle equality constraints, and the cost still increases cubic proportional to the number of auxiliary constraints.

In order to analyze the physiological role of every human muscle, Nakamura *et al.* [8] propose a method to estimate muscle forces from the torque made at the joints using linear programming. Stimulated by their approach, we propose in this paper an IK solver based on linear programming.

3. METHODOLOGY

IK is a technique to calculate the motion of the whole body from the trajectories of the body segments. Since the relationship of the joint angles and the positions of the segments are nonlinear, finite differences are often used as parameters in numerical approaches. If the finite differences of the generalized coordinates are represented by $\boldsymbol{\Delta q}$, and the translational and rotational motions of a segment are represented by $\boldsymbol{r}$, the relationship of $\boldsymbol{\Delta q}$ and $\boldsymbol{r}$ can be written as:

$$\boldsymbol{r} = \boldsymbol{J}\boldsymbol{\Delta q} \tag{1}$$

where $\boldsymbol{J}$ is called the Jacobian matrix. Usually, given $\boldsymbol{r}$, there are infinite sets of $\boldsymbol{\Delta q}$ that satisfy Eq. (1), as the DOFs of the system are often larger than the number of constraint equations.

3.1 The Traditional Pseudo-Inverse Method

In order to cope with redundancy, Whitney [11] proposes a method to solve the IK problems by optimizing a quadratic function as:

$$Q(\boldsymbol{\Delta q}) = \boldsymbol{\Delta q}^T \boldsymbol{W} \boldsymbol{\Delta q} \tag{2}$$

where $\boldsymbol{W}$ is a positive definite matrix. The $\boldsymbol{\Delta q}$ that minimizes Eq. (2) can be obtained by

$$\boldsymbol{\Delta q} \;=\; \boldsymbol{W}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T)^{-1}\boldsymbol{r} \tag{3}$$

In case $\boldsymbol{W}$ is an identity matrix, $\boldsymbol{\Delta q}$ can be calculated by

$$\boldsymbol{\Delta q} \;=\; \boldsymbol{J}^{\boldsymbol{T}}(\boldsymbol{J}\boldsymbol{J}^{\boldsymbol{T}})^{-\boldsymbol{1}}\boldsymbol{r} \;=\; \boldsymbol{J}^{+}\boldsymbol{r} \tag{4}$$

where matrix $\boldsymbol{J}^{+}$ is called the pseudo-inverse matrix. This gives the least squares solution for $\boldsymbol{\Delta q}$. The set of $\boldsymbol{\Delta q}$ satisfying Eq. (1) can be written in the following form [6]:

$$\boldsymbol{\Delta q} = \boldsymbol{J}^{+}\boldsymbol{r} + (\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})k \tag{5}$$

where k is an arbitrary vector. The problem with the pseudo-inverse method is that the computational cost increases cubically to the number of constraints, as $\boldsymbol{J}\boldsymbol{J}^{\boldsymbol{T}}$ is dense.

3.2 The Lagrangian Multiplier's Method

In order to handle large scale systems, an efficient method is needed. Baraff [1] proposes to use Lagrange multipliers to solve forward dynamics problem. This method can be applied to solve the IK problems. The following equation is used to calculate the motion of the generalized coordinates:

$$\begin{matrix} \boldsymbol{W} & \boldsymbol{J}^{\boldsymbol{T}} \\ \boldsymbol{J} & \boldsymbol{0} \end{matrix} \quad \begin{matrix} \boldsymbol{\Delta q} \\ \boldsymbol{\lambda} \end{matrix} \quad = \quad \begin{matrix} \boldsymbol{0} \\ \boldsymbol{r} \end{matrix} \tag{6}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers. Since the connectivity of the multibody is usually low, the matrix on the left will be sparse. As a result, $\boldsymbol{\Delta q}$ can be obtained efficiently by a LU decomposition library that takes advantage of the sparsity. The problems that remain here are that this method cannot handle inequality constraints, which are usually needed to limit the range of the generalized coordinates, and the cost still increases cubic proportional to the number of constraints [1].

3.3 LPIK: The New Approach

As the pseudo-inverse method is equivalent to finding the least squares solution of $\boldsymbol{\Delta q}$ that satisfies Eq. (1), we propose to calculate the absolute sum solution of $\boldsymbol{\Delta q}$, which is the sum of absolute values of the elements in $\boldsymbol{\Delta q}$, instead of the least squares solution. The IK solution is by minimizing the absolute sum using LP as follows:

$$\min_{\boldsymbol{\Delta q},\,\boldsymbol{\delta}} \boldsymbol{a\delta} \quad \text{where} \begin{cases} \boldsymbol{r} = \boldsymbol{J}\boldsymbol{\Delta q} \\ -\boldsymbol{\delta} \leq \boldsymbol{\Delta q} \leq \boldsymbol{\delta} \\ \boldsymbol{\delta} \geq \boldsymbol{0} \end{cases} \tag{7}$$

where $\boldsymbol{a}$ is a weight vector to determine the stiffness of the joint. Since absolute operators cannot be used in linear programming, a new variable δ is introduced. The results calculated by the above solution can generate natural and smooth motion. However, the trajectories of the generalized coordinates suffer from jittering. To obtain results similar to those by the pseudo-inverse method, it is necessary to add an additional term to Eq. (7) as:

$$\min_{\boldsymbol{\Delta q},\,\boldsymbol{\delta},\,\gamma} \boldsymbol{a\delta} + \alpha \boldsymbol{b}\gamma \quad \text{where} \begin{cases} \boldsymbol{r} = \boldsymbol{J}\boldsymbol{\Delta q} \\ -\boldsymbol{\delta} \leq \boldsymbol{\Delta q} \leq \boldsymbol{\delta} \\ -\gamma \leq \boldsymbol{\Delta q} - \boldsymbol{\Delta q'} \leq \gamma \\ \gamma \geq \boldsymbol{0}, \boldsymbol{\delta} \geq \boldsymbol{0}, \boldsymbol{a} \geq \boldsymbol{0}, \alpha > 0 \end{cases} \tag{8}$$

where $\boldsymbol{\Delta q'}$ is the solution of $\boldsymbol{\Delta q}$ calculated in the previous step. α is a constant scaler. $\boldsymbol{a\delta}$ and $\boldsymbol{b}\gamma$ represent the absolute sums of $\boldsymbol{\Delta q}$ and of $\boldsymbol{\Delta q} - \boldsymbol{\Delta q'}$. Term $\alpha\boldsymbol{b}\gamma$ has an effect to remove the jittering from the trajectories of the generalized coordinates. Term $-\gamma \leq \boldsymbol{\Delta q} - \boldsymbol{\Delta q'} \leq \gamma$ compares the difference of the generalized coordinates of the current and the previous frames. It adds a damping effect to the motion to remove the jittering. The resulting trajectories look closer to those by the pseudo-inverse method.

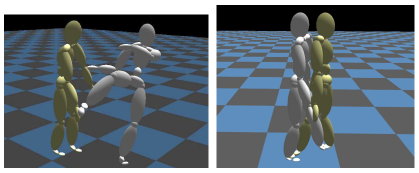

Figure 1: Two characters pushing each other.

No. of constr.	LPIK (ms)	LM (ms)	PI (ms)
8	6.60	3.67	21.93
13	9.95	4.04	24.92
18	13.14	4.53	32.92

Table 1: Performance of LPIK with two characters pushing each other.

Among the linear programming methods available, the simplex method shows the best performance for solving this problem. Since we are solving a problem for real-time computer animation, the computation time must be short while the quality of the animation only needs to be good enough. By using Eq. (8), while the computation time can be significantly reduced, the motion generated is similar to that obtained by the pseudo-inverse method.

4. EXPERIMENTAL RESULTS

To evaluate the performance of LPIK, we have conducted experiments with multiple human characters. We use human models composing of 20 segments with 63 DOFs. When solving the problem for multiple characters, the DOFs of all the characters are handled together and all the constraints are put into the Jacobian matrix to solve for a single IK problem. Only the positional constraints are included in the Jacobian and the rotations of the segments are not specified. Therefore, when there are k characters appearing in the scene and the total number of constraints is m, the size of the Jacobian becomes $3m \times 63k$. The experiments here are conducted on a PC with a 2.6GHz P4 CPU and 1GB RAM. Linear programming is conducted by the mathematical library *CPLEX* [2]. The Lagrange multiplier's method was conducted by *SuperLU* [10], which handles sparsity of the matrix very efficiently.

In our first experiment, two characters interfering each other at numerous contact points are generated (Fig. 1). When the characters are standing, the feet are constrained onto the ground so that no sliding occurs. The pelvis of each character is controlled sinusoidally so that the characters may appear to be pushing each other. We gradually increase the number of contact points between the two characters to study the computational cost. Table 1 shows the performance comparison of LPIK, the Lagrange multiplier's method (LM) and the pseudo-inverse method (PI). We may observe that the computation time of LPIK increases roughly linearly with the number of constraints, and is much lower than the pseudo-inverse method, although it is high than the Lagrange multiplier's method.

In our second experiment, multiple characters holding each other's hands in a circle are generated (Fig. 2). Again, the feet are constrained onto the ground and the number of characters increases to forty. The performance is again compared

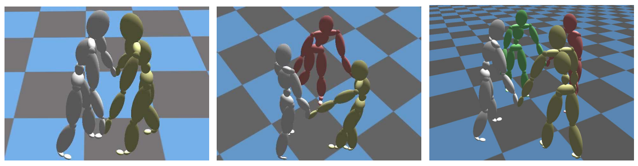

Figure 2: Multiple characters holding each other's hands and moving.

No. of chars.	No. of constr.	DOFs	LPIK (ms)	LM (ms)	PI (ms)
2	8	126	6.17	4.76	36.87
4	16	252	15.48	7.35	379.32
20	80	1260	157.86	168.36	58623
40	160	2520	407.11	757.52	571500

Table 2: Performance of LPIK with multiple characters pushing each other.

with PI and LM. Here, both the DOFs and the number of constraints increase linearly when a new character is added to the scene. All the characters' torsos are controlled in a sinusoidal manner. The motions of the remaining joints are calculated so that the constraints imposed at the hands and the feet are satisfied. Table 2 shows the our experimental results. The computation times of our method and LM are plotted in Fig. 3. We can see that the performance of LPIK is better than LM when the number of characters is above twenty. This means that LPIK is more efficient than LM when the number of DOFs and constraints are large.

In our third experiment, the trajectories of the generalized coordinates are examined when using LPIK. Usually, the appearance of the motions generated by LPIK looks natural and smooth. However, due to the use of a linear function as an objective function, there is a possibility that some jittering may occasionally appear in the motion. To examine such effect, we drag the hand of a character to the front and then analyze the finite difference of the flexion at the chest as shown in Fig. 4. When using LPIK without the second term of the objective function, i.e., using Eq. (7), the trajectory of the flexion is sometimes jaggy (the dashed line in Fig. 4) and the chest tends to move a lot at the beginning. By using LPIK with the second term of the objective function, i.e., using Eq. (8), the jaggyness is removed and the trajectory (the solid line) is now similar to that of PI (the dotted line).

5. COMPLEXITY ANALYSIS

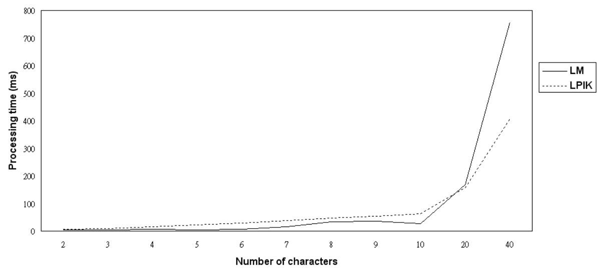

Figure 3: Performance comparison of multiple characters holding each other's hands and moving.

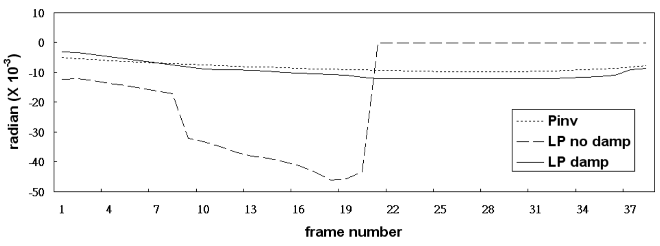

Figure 4: The flexion trajectory of the chest when pulling the hand to the front.

To study the computational complexity of the three different approaches for IK, we denote the DOFs as n and the number of constraints as m. The most costly part of the pseudo-inverse method in Eq. (4) is the inversion of $\boldsymbol{JJ^T}$, which costs $O(m^3)$. Hence, its performance will drop significantly when the number of constraints increases. In [1], Baraff also points out that the cost for the Lagrange multiplier's method increases cubically with the number of constraints. Our experiments have shown that LPIK becomes more efficient when the number of constraints is large.

In the previous experiments, we have found that the computational cost of LPIK increases linearly either as the number of constraints increases while keeping the DOFs constant or as the DOFs increase while keeping the number of constraints constant. When the DOFs and the number of constraints increase linearly at the same time, as in the second experiment above, the computational cost increases in a non-linear manner. Based on statistical analysis, we have found that the cost here increases square proportional to the number of characters in the scene. Taking into account these results, it is possible to say that the computational cost of LPIK can be approximated by $\sim O(mn)$. It must be noted that this is not a general formula for LPIK, as the interaction of a character is limited to only two other characters in this example. However, in most animations, the number of characters each one interferes with is no more than two. Therefore, the computational cost must be close to this figure in most cases. Our experiments also show that the change in the DOFs affects the computation time more significantly than the change in the number of constraints.

6. DISCUSSIONS AND CONCLUSIONS

In this paper, we have proposed a new method to solve the IK problems using linear programming optimization. Its advantage is that it shows good performance for general articulated structures with multiple constraints. We have demonstrated this through comparison with other numerical methodologies. Comparing with the pseudo-inverse method, our method has a much lower computation time. Comparing with the Lagrange multiplier's methods, our method has a slight higher computation time when the number of characters is small. However, as the number of characters increases, our method begins to show a much lower computation time, while being able to maintain the quality of the generated motion as good as other methods.

Another advantage of LPIK is that inequality constraints can be used, which are useful for setting the range of the generalized coordinates and handling collisions between segments. In order to handle inequality constraints with a pseudo-inverse method or the Lagrange multiplier's method, it is necessary to check whether the constraints are satisfied or not first. If they are violated, the problem must be solved again by adding another equality constraint to keep the solution on the boundary of the inequality constraint.

Generally speaking, the computational cost of the simplex method changes according to the problem. It depends on various factors such as the sparsity of the constraint matrix and the starting point of the computation. Theoretically, in the worst case it becomes exponentially proportional to the number of constraints or variables. However, statistically, the simplex method gives the solutions in a very short time. Hence, we analyze the performance of LPIK by changing the DOFs and the number of constraints, and the results are convincing. This means that LPIK can be a very powerful tool for generating scenes in which many characters are densely interacting with each other such as rugby or american football. It is also suitable for real-time applications such as 3D games and virtual environments.

7. ACKNOWLEDGEMENTS

The work described in this paper was partially supported by a CERG grant from the Research Grants Council of Hong Kong (RGC Reference No.: 9040930) and a SRG grant from City University of Hong Kong (Project No.: 7001759).

8. REFERENCES

[1] D. Baraff. Linear-time Dynamics Using Lagrange Multipliers. *Proc. of ACM SIGGRAPH'96*, pages 137–146, 1996.

[2] CPLEX. *ILOG Inc., http://www.ilog.com.*

[3] K. Grochow, S. Martin, A. Hertzmann, and Z. Popovic. Style-based Inverse Kinematics. *ACM Trans. on Graphics*, 23(3):522–531, Aug. 2004.

[4] L. Kovar and M. Gleicher. Automated Extraction and Parameterization of Motions in Large Data Sets. *ACM Trans. on Graphics*, 23(3):559–568, 2004.

[5] J. Lee and S. Shin. A Hierarhical Approach to Interactive Motion Editing for Human-like Figures. *Proc. of ACM SIGGRAPH'99*, pages 39–48, 1999.

[6] A. Liegeois. Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. *IEEE Trans. on Systems, Man, and Cybernetics*, 7(12):868–871, 1977.

[7] Y. Nakamura and H. Hanafusa. Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986.

[8] Y. Nakamura, K. Yamane, Y. Fujita, and I. Suzuki. Somatosensory Computation for Man-Machine Interface from Motion Capture Data and Musculoskeletal Human Model. *IEEE Trans. on Robotics*, 21(1), Feb. 2005.

[9] C. Rose, P. Sloan, and M. Cohen. Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation. *Computer Graphics Forum*, 20(3):239–250, 2001.

[10] SuperLU. *http://crd.lbl.gov/∼xiaoye/SuperLU/.*

[11] D. Whitney. Resolved Motion Rate Control of Manipulators and Human Prostheses. *IEEE Trans. on Man-Machine Systems*, 10:47–53, 1969.