An Efficient Energy Transfer Inverse Kinematics Solution

Jing Huang and Catherine Pelachaud

Telecom ParisTech - CNRS,
37, rue Dareau, 75014 Paris, France
{jing.huang,catherine.pelachaud}@telecom-paristech.fr
http://www.springer.com/lncs

Abstract. In this paper, we present a fast and easy-to-implement locally physics-based Inverse Kinematics(IK) method. Our method builds upon a mass-spring model and relies on force interactions between masses. Joint rotations are computed using the closed-form method with predefined local axis coordinates. Combining these two approaches offers convincing visual quality results obtained with high time performance. Our IK solver is suitable for multiple constraints application and constrained 3D humanoid models.

1 Introduction

Kinematics is a general method for manipulating interactively articulated figures and generating postures. It is widely used in computer animation, robotics, bionics simulation studies, mechanical engineering, etc. In computer graphics, articulated skeleton models are used to control virtual vertebral living creatures, such as human beings or animals, which frequently appear in films and video games.

Inverse Kinematics (IK) is a method for computing joint rotation values of individual degrees of freedom via predefined rotation and position constraints. It is often used to animate autonomous agents as neither predefined movements nor keyframe precomputations can be done offline.

We propose a low computational cost, physically-based IK solver which combines a mass-spring system with a traditional heuristics factor. Our method is fast, and can generate good results for IK chain with multiple constraints in a efficient way without increasing too much computations. It can be used in a sequential IK manner for creating full body animations of virtual agents. The energy transfer can be easily combined with the expressivity parameters in virtual character systems which allows us to modulate the expressive quality of the movements. In the remainder of the paper, we first present existing approaches, then we describe our IK solution. We illustrate our method with several examples.

2 Previous Works

In robotics and animation, the production of realistic and plausible animation in a fast and efficient way is still a challenging task. The key frame method is a single manner to generate animation sequences. IK methods can be used to create key frame postures. Different reaching models have been proposed by using different IK solutions. They can be clustered into 3 main methods: analytical, numerical and hybrid methods.

Analytical Methods Analytical methods or closed form solutions, e. g. as proposed by Wu et al. [36] and Gan et al. [13], specify the figure with constraints and solve the IK problem by studying possible relative posture properties. These methods are fast and accurate, but they can be used only for specific constrained models.

Numerical Methods Numerical methods achieve satisfactory solutions for more general cases, but require more computational iterations. Zhao and Badler [38] introduced the idea of local minimization of a set of non-linear equations and proposed their Cartesian space constraints solution.

Tang et al. [27] proposed to use the Verlet integration [29] to solve unconstrained equations, then based on a set of Lagrange multipliers, they applied some constraint forces to correct their motions using Shake model in molecular simulations.

Most recent methods [25, 33] [35] [30] [37] [8] are using the Jacobian matrix to get a linear approximation. This family of methods models the end effector's movement relative to the changes of the whole system which includes translation and rotation transformations between joints.

The Jacobian matrix is a matrix of the first order partial derivatives of vector functions. In an IK system, it keeps the first order linear behavior of the whole system. The linear equation is given by $\Delta p = J(\theta) \times \Delta\theta$, where Δp is the end effectors' movement, $J(\theta)$ is the Jacobian matrix that represents the end effectors' movements relative to the changes of local rotation, $\Delta\theta$ is the local rotation. We need to compute the rotation angle by inverting the function: $\Delta\theta = J(\theta)^{-1} \times \Delta p$. So the remaining problem is to solve the inverse of the Jacobian matrix or it can be seen as a error minimization. Several methods have been proposed to compute or approximate the Jacobian inverse, such as Pseudo-Inverse Jacobian [25, 33], Jacobian Transpose [35], Damped Least Squares (DLS) [30], Singular Value Decomposition Jacobian (SVD-DLS) [37], Selectively Damped Least Squares(DLS) [8].

The second family of numerical IK solvers is based on Newton's method or its approximations, such as those suggested in [10, 12]. The solution is to minimize the error functions. Newton's method is based on a Taylor series expansion to compute the highly complex Hessian matrix. It has a high computational footprint for iterations, but it offers smooth motion results. And when compared to the Jacobian inverse methods, it does not have the singularity problem.

The third family is based on heuristics. One of the most popular methods is Cyclic Coordinate Descent (CCD) [31]. The main idea is to align one joint with the end effector and the target at a time, and to bring the end effector closer to the target iteratively. Several extensions have been made, e. g. by Welman [32] and Canutescu [9]. As a heuristic method, thus without any matrix computation, CCD is extremely efficient. But it suffers from unrealistic looking results and is difficult to handle when multiple end effectors are used.

Another heuristic method was presented by Muller [24], called Triangulation Inverse Kinematics. It is based on the cosine rule to compute each joint's rotation by starting at the root joint and traversing up to the end effector. It is guaranteed to find a solution without constraints. This method requires less computations than CCD does. However, it can only be used for single chain problems. It cannot be applied for complex constraint models. The corresponding constraint version does not provide good results due to independent computations for each joint.

Several methods also propose to solve the IK problem in the position space instead of the orientation space. Aristidou et al. [2] propose a method based on forward and backward iterative movements by finding a joint's new position along a line. Our method follows a similar idea where joint rotation converges by adjusting each joint's position using mass-spring model.

Hybrid Methods Hybrid methods, such as the Morphology-independent representation of motions [22] and Sequential Inverse Kinematics (SIK) [6,28], combine analytical and numerical approaches to reconstruct 3D human body movement. Here, the IK problem is solved sequentially by using different analytical iterative algorithms for various parts of the body in a specified order.

Alternative methods exist as well, such as Sequential Monte Carlo [11] which uses Importance sampling combined with forward kinematics to solve the IK problem. Such a method avoids the computation of the inverse matrix, using the hidden Markov model [1] to define a possible hidden state, and combines a filtering framework to reformulate the IK chain.

In physics-based computer animation research, many works have addressed the control of articulated figures using robotics-inspired proportional-derivative (PD) controllers which can handle different types of motor tasks, such as walking, running, swimming, boxing [39] [7].

There exist also some other methods using particle system, such as Hercker et al. [16], Zordan et al. [40]. They are designed for some complex models with highly varying skeleton morphologies. Our method follows the same direction, but is a lightweight solver that is easier to implement and can generate postures with good qualities.

IK methods are useful for defining gestures without precomputing key frames. In the next section, we will describe our method which is efficient, has a good visual quality and does not suffer from the singularity problem. It can be a good choice for users. We now demonstrate how we apply our method efficiently.

3 Mass-Spring Based Inverse Kinematics

We build a mass-spring system to deal with target reaching tasks. Then, a general angular constraint solver has been integrated in our system for solving the rotations. If the skeleton rotation information is not needed such as the snake game [19], we can simplify the process by only performing the reaching task.

Mass-Spring Systems(MSS) are defined as harmonic oscillator systems in classical mechanics. In computer graphics, they are often used to model particle systems. Positions are adjusted by minimizing the force energy which is conserved in springs. Hooke's Law [23, 26] defines the mass spring system as:

$$\vec{F}\,(t) = -k\ \vec{l}\ (t)$$

where F is a restoring force, proportional to the displacement l in a spring with a positive spring constant k.

The composition of forces received for a mass is defined as: $\vec{F_{sum}} = \sum_{i=0}^{n} \vec{F_i}\,(t)$

We need to find the new position $x(t)$ for each mass to achieve a stable state in the system.

The formula needs to be extended to calculate the new displacement of a mass by using Newton's second law of motion:

$$F(t) = m\frac{d^2}{dt^2}x(t) + \frac{\beta}{m}\frac{dx}{dt} = ma$$

where F is the force, β is the damping factor and a is the acceleration of the mass m. $x(t)$ is the displaced position of the mass at time t, velocity $v = \frac{dx}{dt}$, and acceleration $a = \frac{d^2}{dt^2}x(t)$.

To solve the particle path, we use Newton's second law. This is suitable since the IK chain can be regarded as one particle chain. The path formula can be defined as: $y(t_1) = f(t_1, y(t_0))$.

The basic method for solving the particle motion is based on Euler's equation. It is an approximation of Newton's method [10, 12].

In our case, we propose to use Störmer-Verlet integration [15, 29] instead of Euler's equation. Because it can be seen as an approximation of the second derivative by using central differences and it is more stable than Euler's method.

The Störmer-Verlet formula is defined as

$$\begin{aligned}\frac{\Delta^2 \boldsymbol{x}_n}{\Delta t^2} &= \frac{\frac{\boldsymbol{x}_{n+1}-\boldsymbol{x}_n}{\Delta t} - \frac{\boldsymbol{x}_n-\boldsymbol{x}_{n-1}}{\Delta t}}{\Delta t} \\ &= \frac{\boldsymbol{x}_{n+1} - 2\boldsymbol{x}_n + \boldsymbol{x}_{n-1}}{\Delta t^2} \\ &= \boldsymbol{a}_n \\ \boldsymbol{x}_{n+1} &= 2\boldsymbol{x}_n - \boldsymbol{x}_{n-1} + \boldsymbol{a}_n\,\Delta t^2, \end{aligned} \tag{1}$$

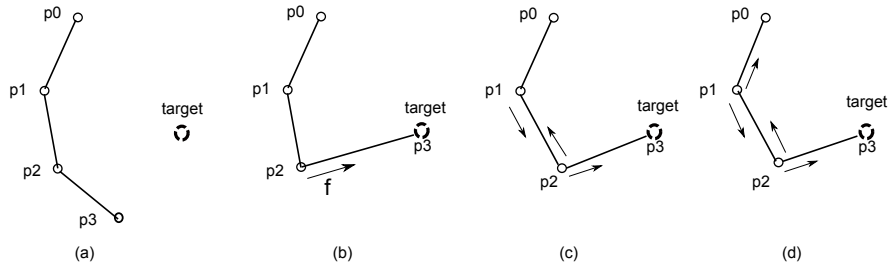

Fig. 1. illustrates the procedure of our solver: (a) Initial configuration, (b) End-effector is moved to target position, (c) and (d) mass-spring system is simulated until equilibrium. p_i represents the joints on the chain, and each joint has a serial position x_n generated by our solver through time.

Let us consider an IK chain with a set of joints $J_i, i = 1, .., n$. A set of mass points M_i and springs S_i is created by matching the joints and the distances. The end effector is given as E and the target as T. For this computation, we ignore the rotation information, and set the general mass value equal to 1, i. e. the system only conserves the positional information of mass points. Two implementions can be used to solve the chain.

Method 1: Force based method We start each iteration by adding a user defined external force to the end effector: $f = -k(T-E)$. Then, we compute the internal force using the mass spring model. The acceleration can iteratively drive the mass points to the new positions by using Verlet's formula (Eq.1). During each iteration, we check the sum of internal forces of the entire system and make sure that $\| \sum \vec{F}_{sum} \| < \epsilon$. We defined the ϵ as a threshold which influences the iterations. Finally, the end effector will always converge, i. e. get closer to the target. We keep the normalized directional vector of each pair of mass points (one mass point and its parent mass point) to build local axis for each joint.

Method 2: Position based method This is an optimized version of method 1. It is illustrated in Figure 1. We fix the end effector p_3 to the target position, using spring energy to correct the positions of mass points.

To achieve faster convergence, we use the clamp function [8] which retargets the chain to avoid too much oscillations when the target is too far away.

After all iterations, we convert the mass system into a joint system. We have the directional vector of each joints pair (a joint and its parent joint). We then use the closed-form method [17] to compute the local rotation of each joint using its bone vector and its up vector which is perpendicular to the bone vector, and we update the whole skeleton system. The up vector is automatically updated when we perform a rotation on its bone vector. This is important when we need to solve general constraints.

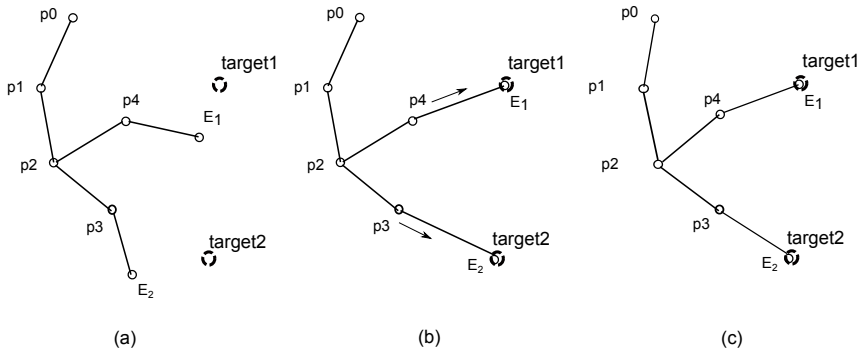

Fig. 2. illustrates the multiple end effectors procedure: (a) Initial configuration, (b) End-effectors are moved to targets positions, (c) mass-spring system is simulated until equilibrium.

3.1 Multi-Targeting

Models can be defined by one or more kinematic chains. For example, a human body can be defined by several chain constraints by multiple targets. Even a complex chain model can have multiple positional constraints, i. e. multiple end effectors. Therefore, it is useful to adapt the IK solver to support multiple targeting tasks. As mass-spring systems are used to cope with large data sets of particles, our mass-spring system based IK method can easily be extended to be suitable for multi-targeting tasks. The mass-spring system can also be applied in a parallel fashion [14]. The algorithm is as follows: All the end effectors are set to their target positions, then we start the iterations, as illustrated in Figure 2. When using multiple end effectors, there is at least one joint that is connected to multiple joints. We call such a joint a "sub base joint". In Aristidou's model FABRIK [2], the authors propose to calculate the centroid of the virtual sub base joint of each sub chain when doing the forward stage. The centroid of the virtual sub base joints corresponds to the new position of the sub base joint. In contrast, with our method, we do not need such a centroid supplementary stage; it is directly obtained. The last step of our algorithm updates the skeleton structure. In skeletal models, one joint can only have one rotational value. We use either the priority or weight method [4] to convert rotations to the skeleton structure. The first uses the preferential sub-chain rotation, while the latter the average by weights between sub-chain rotations.

3.2 General Joint Constraints

In most analytical skeleton models, joints can be modeled as hinges, pivots, ellipsoids, etc. Several anatomical and biomechanical methods exist that formalize specific motion intervals for articulated characters, with the main purpose of having multiple parameters to describe the hierarchical structure within a motion space.

We chose to use a ball and socket model for joints. The ball and socket joint is defined by 3 DOFs, i. e. with 3 euler angles, different rotation types are combined together. The euler angles can be converted from a quaternion [21].

Several methods have been presented for solving constraint problems. Zhao and Badler [38] introduced their Cartesian space constraints solution. Wilhelms and Van Gelder [34] define spherical joint limits with reach cones, that have been predefined by the user. The final joint value is obtained by projecting the joint segment onto the reach cone to ensure that they are all within certain limits. Blow et al. [5] present a similar idea using a reach window, but they aim to find the closest point on the polygon instead of the direct intersection point. Korein et al. [20] decompose the ball and socket joint rotation into arbitrary orientation components, and directly control the rotation of the joint. Baerlocher and Boulic [3] also propose to define a two components constraint to stabilize the Jacobian method. Since our method has a similar position adjustment process as FABRIK [2], our IK solver is compatible with their constraint solver for solving positional constraints.

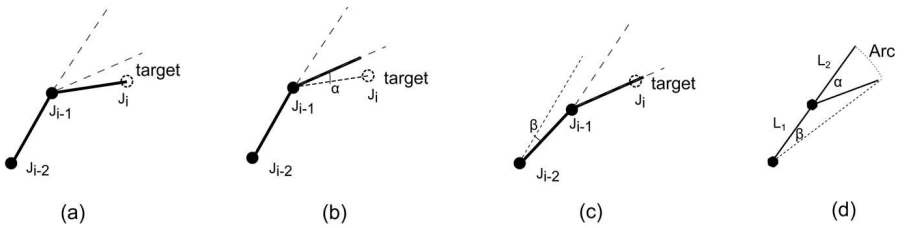

Fig. 3. Example of mass-spring joint system constraints: **(a)** The end effector J_i reaches the target, but the rotation of J_{i-1} exceeds the constraints, shown by the dotted line. **(b)** Rotation of J_{i-1} to the nearest limit boundary with angle α. **(c)** Rotation of J_{i-2} to move the end effector closer to the target with angle β. **(d)** Approximation of β.

Our skeleton system is based on the conversion from global space position into local space rotation. We propose another angular constraint(boxed intervals) solver based on a parametrization of rotations in local Euler space with X, Y, Z axes. Nevertheless, all the rotations are computed and applied with quaternions before the constraint stage. We need to do the conversion between quaternion and euler angles [21]. Each DOF constraint is defined within range values. The minimum and maximum values are used to limit the rotation of one joint in each DOF, resulting in six boundaries. The main idea of our method is to correct the local orientation, propagate exceeding values to the parent and iteratively converge to a good result within limits.

We apply our constraint solver by checking all the joints' rotation values after the mass spring process. The order is from the end effector to the root joint. The rotation needs to be decomposed along 3 axes. As illustrated in Figure 3, we check the local rotation of J_{i-1}. If the rotation in local space is larger than the max limit angle or smaller than the min limit angle, it means the rotation value is exceeding the constraint value. We call this event a *constraint event*. In

that case, we cut off the exceeded value of J_{i-1} and perform a second rotation β for J_{i-2} to reposition the end effector. We can easily compute β by the following approximation: $Arc = \alpha L_2 \approx \beta(L_1 + L_2)$.

Afterwards, we check the local rotation of J_{i-2}. We proceed in a bottom-up fashion, by pushing exceeding values of a joint to its parent in case the constraint is not satisfied until the base joint is reached. If the root also triggers the constraint event, we only cut off the exceeded value and restart the full loop until a solution is found or the max loops number is reached. This iterative process allows us to solve general constraints and to keep the end effector as close as possible to the target.

4 Results

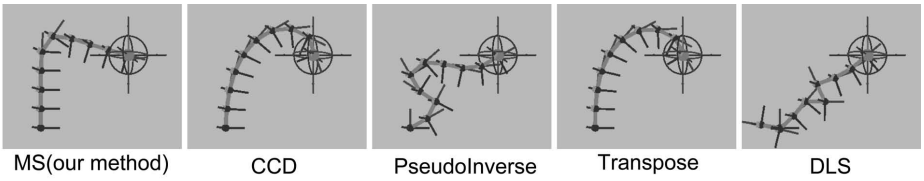

Fig. 4. Results of using different IK solvers without constraints.

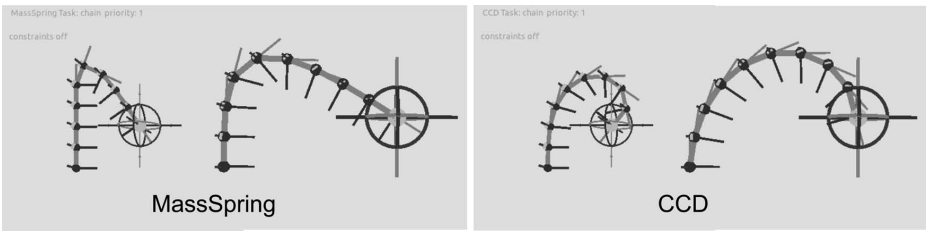

Fig. 5. One comparison result between our method and CCD method: the CCD resulting chain rolls around itself.

Our method has been implemented in Java and C++. We have implemented several other existing methods (CCD, PseudoInverse, Tranpose, DLS, etc) in C++ for comparison and experimental purposes. For each IK solver we have implemented, we measure the duration of the whole process including the computation of positions and rotations for the whole chain until the end effector reaches the target.

The result is illustrated in Figure 4. When compared to other methods, our approach gives good results with less changes on the joints from initial states. More particularly, if we compare our method with CCD, our method will not have the rolling artifacts as CCD does(see Figure 5) as mentionned in FABRIK [2]. Our method supports multiple positional constraints as shown in Figure 6. It can find the best solution. For other IK methods, it is difficult to solve

this situation. Our method can solve this case in a much more efficient and easier way than other methods.

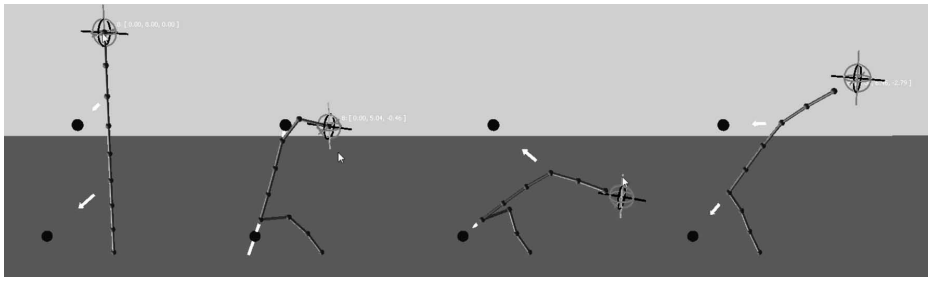

Fig. 6. Our method supports multiple targeting constraints. The 4 images illustrate examples of a chain with two positional constraints and one target(using the manipulator). Our method can find the best solution for getting closer to both the target and the positional constraints. User-defined weights can be assigned to specify how both can be complied.

Method	Joints	Our Method 1 Force-based	Our Method 2 Position-based	CCD [31]	PseudoInverse [25] [33]	Transpose [35]	DLS [8]
Iterations	6	49.6	31.5	33.2	51.8	6.8	55.6
	10	60.5	40.1	39.7	55.9	17.1	68.7
Time(ms)	6	0.195	0.132	0.131	1.095	0.124	1.237
	10	0.253	0.169	0.148	1.310	0.235	1.450

Table 1. Performance comparison between our method and other methods without constraints for a chain of 6 and of 10 joints. The results are averaged over 15 runs.

We also compared the performance of these methods with ours, as shown in Table 1. Our method is as time-efficient as CCD but much faster than the other methods. When an animation needs to be computed, we transform the position information into angular values. This allows us to interpolate these angular values for each frame. When such transformation is not required [19], our method can be even faster. We split the IK reaching task into two parts: reaching a target and updating the rotation while checking constraints.

Limitation Our IK system suffers from the oscillation that derives from using Mass Spring system. Even though small oscillations will not influence the final posture after iterations, we do need to setup appropriate parameters for mass spring system to reduce the oscillation.

Virtual Agent We have integrated our IK solver into our virtual character system. The skeleton is decomposed into several chains(eg. the arm chain is defined from clavicle to wrist). Different constraints can be applied to these IK chains(see Figure 7). The animation of the virtual character is obtained through 2 steps:

compute the body posture to reach the defined constraints and apply the movement expressivity parameters. These latter are used to specify qualitatively the movement of the agent (such as the speed and acceleration of a movement, its tension and amplitude). We have merged our IK solver with our model of expressivity parameters [18]. In particular, we use the arm chain's force energy and the value of the expressivity parameters to influence shoulders and torso movements.

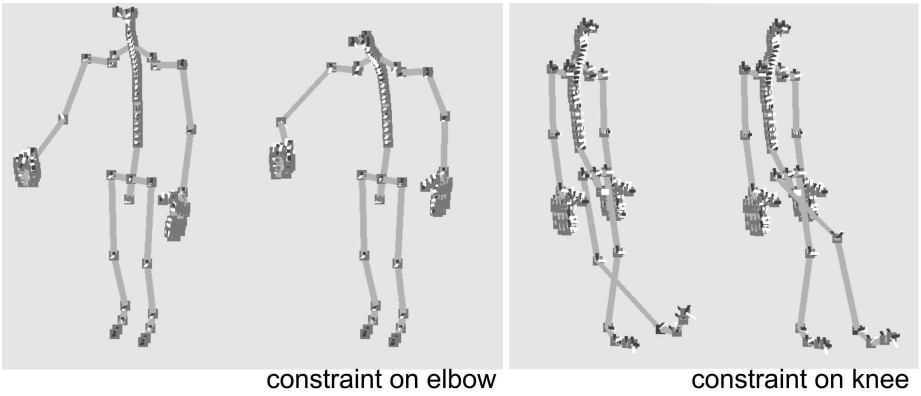

Fig. 7. The IK chains of body parts are constrained: **(Left)**: The arm swivel angle(elbow) is constrained by the expressivity parameters [18]. **(Right)**: Our constraint corrects the leg's rotation.

5 Conclusion

We have presented a novel algorithm to approximate physically-based methods for solving inverse kinematics by using mass-spring systems. Our IK method can be applied to any chain type, with multiple constraints. It provides high quality visual results with computation times that are comparable and, at times, even lower than existing IK methods. Moreover, using mass-spring system provides more control parameters, such as mass quality, spring stiffness and damping factor to fine tune the movement. So our model offers more flexibility to configure IK chains of any complexity. We have merged our model with a model of expressivity parameters to animate human characters. Our method can also be a good option for gaming usage.

Acknowledgements

This work has been funded by the French National Research Agency (ANR, for the project: CeCil). Many thanks to the anonymous reviewers that provided excellent feedbacks.

References

1. Hidden Markov models: applications in computer vision. World Scientific Publishing Co., Inc., River Edge, NJ, USA (2002)
2. Aristidou, A., Lasenby, J.: FABRIK: A fast, iterative solver for the inverse kinematics problem. Graphical Models 73(5), 243–260 (2011)
3. Baerlocher, P., Boulic, R.: Parametrization and range of motion of the ball-and-socket joint (2000)
4. Baerlocher, P., Boulic, R.: An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. The Visual Computer 20, 402–417 (2004)
5. Blow, J.: Inverse kinematics with quaternion joint limits. Game Developer (2002)
6. Boulic, R., Varona, J., Unzueta, L., Peinado, M., Suescun, A., Perales, F.: Evaluation of on-line analytic and numeric inverse kinematics approaches driven by partial vision input. Virtual Reality 10(1), 48–61 (2006)
7. Bounard, A., Gibet, S., Wanderley, M.M.: Hybrid inverse motion control for virtual characters interacting with sound synthesis: Application to percussion motion. Vis. Comput. 28(4), 357–370 (Apr 2012), http://dx.doi.org/10.1007/s00371-011-0620-9
8. Buss, S.R., Kim, J.s.: Selectively damped least squares for inverse kinematics. Methods 10(3), 1–13 (2004)
9. Canutescu, A.A., Dunbrack, R.L.: Cyclic coordinate descent: A robotics algorithm for protein loop closure. Protein Science (5), 963–972 (2003)
10. Chin, K.W., Konsky, B.R.V., Marriott, A.: Closed-Form and Generalized Inverse Kinematics Solutions for the Analysis of Human Motion, vol. 19 (1997)
11. Courty, N., Arnaud, E.: Inverse kinematics using sequential Monte Carlo methods. Lecture Notes in Computer Science
12. D.F.Shanno, P.C.Kettle: Optimal conditioning of quasi-newton methods. Mathematics of computation 24(111) (1970)
13. Gan, J.Q., Oyama, E., Rosales, E.M., Hu, H.: A complete analytical solution to the inverse kinematics of the pioneer 2 robotic arm. Robotica 23, 123–129 (January 2005)
14. Georgii, J., Westermann, R.: Mass-spring systems on the gpu. Simulation Modelling Practice and Theory 13, 693–702 (2005)
15. Hairer, E., Lubich, C., Wanner, G.: Geometric numerical integration illustrated by the stormer/verlet method. Acta Numerica 12, 399–450 (2003)
16. Hecker, C., Raabe, B., Enslow, R.W., DeWeese, J., Maynard, J., van Prooijen, K.: Real-time motion retargeting to highly varied user-created morphologies. ACM Trans. Graph. 27(3), 27:1–27:11 (Aug 2008)
17. Horn, B.K.P., Hilden, H.M., Negahdaripourt, S.: Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America A 4, 629–642 (1987)
18. Huang, J., Pelachaud, C.: Expressive body animation pipeline for virtual agent. Proceedings of 12th International Conference on Intelligent Virtual Agents (2012)
19. Jeff, L.: Making kine more flexible. Game Developer 5(3), 15–22 (1998)
20. Korein, J.U.: A geometric investigation of reach. MIT Press, Cambridge, MA, USA (1985)
21. Kuipers, J.B.: Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality. Princeton University Press (1999)
22. Kulpa, R., Multon, F., Arnaldi, B.: Morphology-independent representation of motions for interactive human-like animation. Computer Graphics Forum 24(3), 343–351 (2005)

23. Lewin, W.: Hook's Law, Simple Harmonic Oscillator. MIT Course: Classical Mechanics, Lecture (1999)
24. Muller-Cajar, R.Mukundan: Triangualation - a new algorithm for inverse kinematics. Proceedings of Image and Vision Computing New Zealand 2007 (5), 181–186 (2007)
25. Roberts, R.G., Maciejewski, A.A.: Singularities, stable surfaces, and the repeatable behavior of kinematically redundant manipulators. International Journal of Robotics Research 13, 70–81 (1994)
26. Simo, J.C., Hughes, T.J.R.: Computational Inelasticity. Springer (1998)
27. Tang, W., Cavazza, M., Mountain, D., Earnshaw, R.A.: Real-time inverse kinematics through constrained dynamics. Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments pp. 159–170 (1998)
28. Unzueta, L., Peinado, M., Boulic, R., Suescun, A.: Full-body performance animation with sequential inverse kinematics. Graph. Models 70, 87–104 (September 2008)
29. Verlet, L.: Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. Phys. Rev. 159, 98–103 (Jul 1967)
30. Wampler, II, C.W.: Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. IEEE Trans. Syst. Man Cybern. 16, 93–101 (January 1986)
31. Wang, L.C.T., Chen, C.C.: A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. IEEE Transactions on Robotics and Automation 7, 489–499 (1991)
32. Welman, C.: Inverse kinematics and geometric constraints for articulated figure manipulation. Science C(April), 86 (1993)
33. Whitney, D.E.: Resolved motion rate control of manipulators and human prostheses. Science MM(2), 47–53 (1969)
34. Wilhelms, J., Gelder, A.V.: Fast and easy reach-cone joint limits. journal of graphics, gpu, and game tools 6(2), 27–41 (2001)
35. Wolovich, W.A., Elliott, H.: A computational technique for inverse kinematics, vol. 23, pp. 1359–1363. IEEE (1984)
36. Wu, X., Ma, L., Chen, Z., Gao, Y.: A 12-dof analytic inverse kinematics solver for human motion control. Journal of Information Computational Science 1(1), 137–141 (2004)
37. Yuan, J.: Local svd inverse of robot jacobians. Robotica 19, 79–86 (January 2001)
38. Zhao, J., Badler, N.I.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. ACM Trans. Graph. 13, 313–336 (October 1994)
39. Zordan, V.B., Majkowska, A., Chiu, B., Fast, M.: Dynamic response for motion capture animation. ACM Trans. Graph. 24(3), 697–701 (Jul 2005), http://doi.acm.org/10.1145/1073204.1073249
40. Zordan, V.B., Van Der Horst, N.C.: Mapping optical motion capture data to skeletal motion using a physical model. Proceedings of the 2003 ACM SIGGRAPH symposium on Computer animation pp. 245–250 (2003)