Local Joint–Limits using Distance Field Cones in Euler Angle Space.

Morten Engell-Nørregård
Sarah Niebe
Kenny Erleben
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100, Denmark
Email: mort@diku.dk , niebe@diku.dk , kenny@diku.dk

***Abstract*—Joint–limits are often modeled too simple, causing redundancy and allowing unnatural poses. We model the boundary of the feasible region, using a geometric approach. We show how to generate fast, general joint–limit cones for kinematic figures using signed distance fields. The distance–cone joint–limits can support highly nonconvex joint–limits. The joint–limit geometry is easily edited by an animator, either by adding or moving pose samples or sculpturing the distance–cones**

I. Introduction

For animation or simulation of the motion of articulated figures one would use a model describing the range of motion of the individual joints. Inverse kinematics (IK) skeletons, consisting of a hierarchy of bones where each bone is jointed to a parent bone [1] is such a model. The relative coordinate transforms between connected bones are given by a set of joint parameters. In this work we focus on accurate models handling the legal values of these parameters.

This paper is motivated by the work in [2], where machine learning based methods are used for tracking human poses. In this context, a joint–limit model is needed which is capable of accurate description of the feasible region of a single joint and is computationally fast.

From a mathematical viewpoint, an IK skeleton is a hierarchy of homogeneous coordinate transformations, each bone corresponds to a homogeneous transformation. Thus, the joint parameters form an intrinsic parameter space, a Lagrangian representation, of feasible poses of an articulated figure. The joint parameters are constrained within a continuous connected, closed subset of the parameter space. We call a pose feasible if the joint parameters of the pose are all legal. As such, joint–limits describe the boundaries of the feasible region of poses. The box–limit model is the prevalent model for joint–limits, used in file formats such as Acclaims asf/amc and in animations tools such as Maya or Blender [3], [4]. The box–limit model defines limits for each individual joint parameter independently. Box–limits are computationally cheap and easy to use. They are also easily incorporated into optimization methods. There are two major drawbacks of box–limits, they are rarely tight enough resulting in loosely fitted boundaries and second, they disregard any interdependencies between different joint parameters. The alternatives to box–limits offer more descriptive models at the expense of more complicated models. Our model is local, we model joint parameter dependencies for each joint. Our criteria is that the feasible region is two-manifold, to be able to embed it as the zero level-set of a distance field, allowing us to handle any nonconvex joint–limit regions.

II. Previous Work

Despite the shortcomings,the box–limit model is widely used and as [5] demonstrates, box–limits can be automatically determined from motion capture data. This technique was later used in [2]. The authors of [6] use a sinus–cone model from [7], a human shoulder is modeled by a hierarchical IK skeleton. The sinus–cone model is more general than the box-limit model. A reach–cone model based on an idea from [8] is presented by [9]. This is extended in [10] where a general joint component framework is described. In [11], a spline based implicit joint model is suggested for multi-body dynamic. Due to the implicit nature of the model, the geometry of the boundary of the feasible region can be modeled as box–limits in the spline parameter space and interdependency of parameters is omitted. Shoulder joint–limits are modeled in [12] using quaternion field boundaries. Our approach has similarities with this as it uses an implicit surface and supports general nonconvex joint–limits. Projection operators fall in two categories: constant time operation or iterative search schemes. A major feature of our distance–cone is that the projection operator is a constant time operation. Only the box–limit based methods can compete with this time complexity. On the other hand the memory consumption can be high. In our work each joint needs $I \times J \times K$ cells of a map storing a regular sampling of the distance field, where I, J, K are the resolution of the map along the axes. Adaptive distance maps could be used in place of a regular sampling. Our results show that in most practical cases, coarse maps can be used and since distance–cones are only needed for multiple DOF joints the memory storage is acceptable. Most approaches tend to consider local joint–limit models in that respect our work is no different. Although we do consider the full dependency between joint

parameters of a joint. box–limits, sinus–cones and reach–cones do not offer this full generality. One final aspect of joint–limits is the ease with which an animator or modeler can setup the model. This is termed model selection in machine learning and could be approached as a nonlinear regression problem [13]. Sometimes the model selection can be done more easily by working directly with sampled feasible poses. Distance–cone is one such model. Quaternion boundary fields share similarities with our approach in this regard although their coordinate basis (imaginary part of a quaternion) is nontrivial to work in.

The distance–cone offers full modeling generality of local joint limits with constant time operations and easy model selection. Thus the method is a novel approach for more accurate joint–limits, suitable for interactive applications.

III. The Distance–Cone Model

We have adopted a local model of joint–limits [14], [9], our representation could be extended to cover inter-joint dependencies. Our base assumption is that the feasible configurations of a joint form a single connected component, this implies that feasible joint motion is a connected subset. Thus, the test for feasibility is reduced to a simple inside outside test, in case of an infeasible configuration the point is projected unto the closest feasible point.

We use Z-Y-Z Euler angles as basis, where the orientation is specified by the angles $\vec{p} = \begin{bmatrix}\phi & \psi & \theta\end{bmatrix}^T$. This allows us to work in a 3D space rather than 4D or 9D as would be the case for quaternions and homogeneous coordinates. Furthermore, Euler angles form a good foundation for future work on extending the model with directional statistics [15]. By modeling the motion range geometrically, we have a broader basis of well-known geometrical representations to choose from, e.g. polygonal meshes, tetrahedral meshes etc.

Performance is of great importance, so the geometric representation must support two constant time operations: verification of a feasible joint pose and back projection of infeasible joint poses onto the boundary of the feasible region. Distance fields are known to offer both these qualities, this comes at the cost of storage. A distance field is an implicit representation of the geometry, the field is given by the function $\Phi : \mathbb{R}^3 \mapsto Re$ where $\parallel \nabla\Phi(\vec{p}) \parallel = 1$ everywhere and $\Phi(\vec{p}) = 0$ for all $\vec{p}$ corresponding to the geometry.

A. Building the Distance Map

In principle, any distance computation algorithm could be used. For this paper, we used a simple brute-force approach. Because the distance–cone rigging is done as a preprocessing step, the cost of generating the distance map is of minor importance. Experience shows that a fairly coarse resolution is sufficient, the set of motion we study needs only a low number of temporal samples. Running this process off-line means that generating the distance map is not a bottleneck as might have been suspected, Table I lists runtime statistics.

Grid Resolution	Number of Motion Samples	Storage Requirement	Computing Time (secs.)
$8 \times 8 \times 8$	200	4 kB	0.21
$16 \times 16 \times 16$	200	32,7 kB	1.34
$32 \times 32 \times 32$	200	262 kB	8.30

TABLE I
COMPUTATIONS TIMES OF OUR BRUTE-FORCE APPROACH TO DISTANCE MAP GENERATION FOR 30 JOINTS IN A JUMPING MOTION SAMPLE. OBSERVE THAT COMPUTATIONS TIMES ARE NOT TOO LARGE. IN FACT WE DONT EVEN BOTHER TO STORE THE PRE-PROCESSING FOR OUR TESTING.

B. Adaptive Sampling of Motion

We use the chronological ordering of our samples to interpolate between neighboring samples, sparsely sampled regions are thus subsampled using spherical linear interpolation(slerp [16]) in angle space. We can subsample the region between two neighboring samples as densely as necessary, we ensure that the distance between the new samples are never greater than half the grid resolution. The resulting quaternions are then converted into Euler angles and added to the list between the existing samples. If using disjoint motions, some interpolation scheme must be established to connect the manifolds of the two motions. For testing purposes, we simply interpolate between the last frame of one animation with the first frame of the second animation. For the test sets used, this simple approach works well.

C. The signed distance property

All motion samples are feasible poses, not all feasible poses are necessarily represented. Some poses lie in the interior of the feasible region, some lie on the boundary. When we use a specific motion to build our distance–cone, we wish to treat all samples as if they lie on the boundary of the feasible region. To add an interior void to the representation, the distance map is converted into a signed distance map. A run through all cells, marking those cell not crossing a zero level-set as positive. In the end, all unmarked cells must be interior cells and are thus marked as negative. This requires that the feasible region is a closed manifold, if not the region will be hollowed out. This can be helped by either ensuring that the motion is sufficiently densely sampled or by using an alternative method for determining boundary cells. Ensuring the density of the samples has shown to be difficult [14], and for this application the extra work is not needed. The single motion sampling represents a very limited area where all samples can be assumed to be exterior and only points sufficiently close to a sample is considered feasible.

D. Applying Distance–Cones for Inverse Kinematics

We use a inverse kinematic modeling approach similar to [1], the IK problem is formulated as an optimization problem which is solved using an iterative line search method. The feasibility of a given IK iterate, $\vec{p}$, is determined by testing the corresponding Euler angle distance value in the distance–cones:

$$\Phi(\vec{p}) \leq \varepsilon \Rightarrow \vec{p}\ is \text{ feasible} \quad (1)$$

ε is a threshold value to counter round-off and impression errors, the impact of this parameter is analyzed in our results section. The lookup in the distance–cone is performed in constant time by indexing the surrounding grid points and performing a trillinear interpolation at the lookup position. Infeasible poses are projected back onto the feasible region, by moving in the opposite direction of the distance map gradient. The gradient is computed as a central finite difference approximation:

$$\nabla\Phi_{i,j,k} = \begin{bmatrix} \frac{\Phi_{i+1,j,k}-\Phi_{i-1,j,k}}{2\Delta i} \\ \frac{\Phi_{i,j+1,k}-\Phi_{i,j-1,k}}{2\Delta j} \\ \frac{\Phi_{i,j,k+1}-\Phi_{i,j,k-1}}{2\Delta k} \end{bmatrix} \quad (2)$$

where Δi,Δj,Δk, denotes the grid scaling along each coordinate axis. The gradients at the grid nodes surrounding $\vec{p}$ are interpolated using a trillinear interpolation on each component of the gradient. The projection of the infeasible $\vec{p}$ is then:

$$\vec{p} \leftarrow \vec{p} - \Phi(\vec{p})\nabla\Phi(\vec{p}) \quad (3)$$

The central difference approximation of the gradient may cause numerical dissipation in the computation of $\Phi(\vec{p})$ and $\nabla\Phi(\vec{p})$, to alleviate this the projection (3) can be applied more than once. Due to the distance maps properties more than two iterations is not needed. This gives a constant time operation.

IV. Results and Experiments

Due to our application perspective we have chosen to verify and validate the distance–cones in the context of IK. We use the box–limit model as a base of reference, mostly due to its wide-spread use in interactive application and because it is the current model used in human motion tracking. We have excluded the alternative methods from the testing since none of them live up to both the time and modeling demands of our application. We focus on the quality of animation, as well as performance and accuracy.

A. Parameterizing the Distance–Cones

The distance–cone model is dependent on the user specified grid resolution $I \times J \times K$ and the ε parameter. The parameters are orthogonal in the sense that grid resolution mostly influences the distance–cone generation while the feasibility threshold parameter is a run-time only parameter. We tested different grid resolutions and it turned out that for our single motion sampled distance–cones there seemed to be a sharp line around a resolution of $16 \times 16 \times 16$, above which the animations ran smoothly and with acceptable visual quality. Under this resolution threshold, the animations where jagged and tended to get stuck.

The ε parameter needs to be large enough to ensure the existence of a solution for all poses, yet small enough to counter loose limits. We observed that, for values above 0.2 the limits where too loose, and for values below 0.2 the animation is jittery and the IK solver tends to stall.

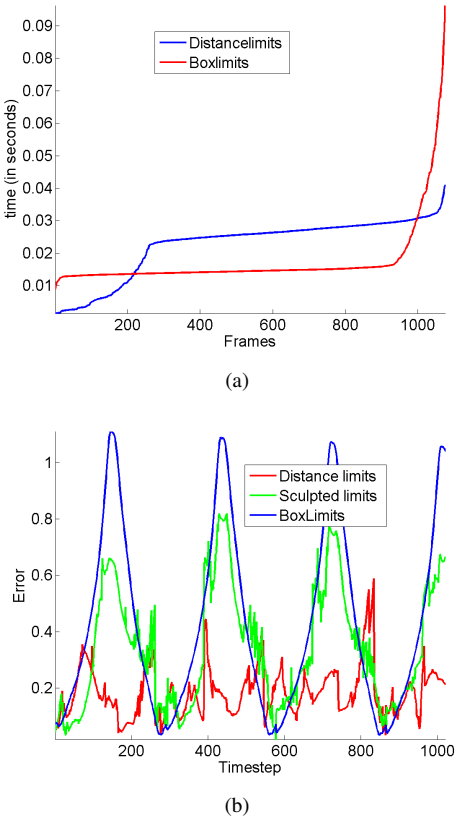

Fig. 1. (a) Computation times using box–limits versus distance–cones. The measurements are ordered by value to make comparison easier. (b) Error measure of difference from motion capture using three different joint–limit methods.

B. Constant Time Performance

We have measured the computational time in an application where end-effectors of an IK solver is driven by the corresponding end-effector positions in a motion capture example. The measurements are performed on the IK solution. Figure 1(a) shows a plot of our measurements. As it can be seen our experiment show that the distance–cones are slower, than the box–limits. this was to be expected, however we can also see that the worst case performance of the distance–cones are much better than the box–limits. Thus, in the worst case the distance–cones still has approximately 20 fps, while the box–limits has approximately 10 fps.

C. Increased Modeling Accuracy

Using the more constraining distance–cone model, we expect a gain in accuracy as this should reduce the redundancy of the simulation. The test uses an elbow, this joint is a child joint of the shoulder which has a wide motion range. We measured the deviance of the joint using both sampled, sculpted, and box–limits, Figure 1(b). As expected, the results show that the box–limits does not constrain the position very much, the error compared to the motion capture is as large as 1.0 unit, corresponding to 20cm. The simulated motions are shown in Figure 2.

Fig. 2. Comparison study of resulting animation quality of box–limits versus distance–cones. The red figure on the left shows distance–cones, the blue figure on the right shows box limits. The green figure in the center is the motion capture reference.

V. Discussion

We have presented a novel joint–limits model using distance–cones. The model supports general nonconvex joint–limits of 3 DOF and works well with sampled motion data. The memory usage is cubic in the resolution of the mesh, however, in most cases it is possible to get by with a comparably low resolution in which case the memory usage is acceptable. The assumption of locality has been shown to be insufficient, e.g. the orientation of the hip joint indeed has an effect on the joint–limits of the knee. The fitting of our model, although loose on account of the threshold, is still the tightest fitting model.

Acknowledgment

This research presented is part of the HUMIM research project

References

[1] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Trans. Graph.*, 13(4):313–336, 1994.

[2] Søren Hauberg, Jerome Lapuyade, Morten Engell-Nørregård, Kenny Erleben, and Kim Steenstrup Pedersen. Three dimensional monocular human motion analysis in end-effector space. Lecture Notes in Computer Science, pages 235–248. Springer, August 2009.

[3] Blender. The blender foundation. web page. http://www.blender.org.

[4] Autodesk. Autodesk. web page. http://usa.autodesk.com.

[5] Morten Engell-Nørregård and Kenny Erleben. Estimation of Joint types and Joint Limits from Motion capture data. In *WSCG 2009: 17-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, ., 2009. .

[6] Walter Maurel and Daniel Thalmann. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics*, 24(2):203–218, 2000.

[7] A. E. Engin and S. T. Tümer. Three-dimensional kinematic modelling of the human shoulder complex—part i: Physical model and determination of joint sinus cones. *Journal of Biomechanical Engineering*, 111(2):107–112, 1989.

[8] James U. Korein. *A Geometric Investigation of reach*. MIT Press Cambridge, 1985.

[9] Jane Wilhelms and Allen Van Gelder. Fast and easy reach-cone joint limits. *J. Graph. Tools*, 6(2):27–41, 2001.

[10] Wei Shao and Victor Ng-Thow-Hing. A general joint component framework for realistic articulation in human characters. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 11–18, New York, NY, USA, 2003. ACM.

[11] Sung-Hee Lee and Demetri Terzopoulos. Spline joints for multibody dynamics. *ACM Trans. Graph.*, 27(3):1–8, 2008.

[12] Lorna Herda, Raquel Urtasun, Andrew Hanson, and Pascal Fua. Automatic determination of shoulder joint limits using quaternion field boundaries. *International Journal of Robotics Research*, 22(6):419–434, June 2003.

[13] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[14] Lorna Herda, Raquel Urtasun, Pascal Fua, and Andrew J. Hanson. Automatic determination of shoulder joint limits using quaternion field boundaries. *I. J. Robotic Res.*, 22(6):419–438, 2003.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.

[16] Erik Dam, Martin Koch, and Martin Lillholm. Quaternions, interpolation and animation. Technical report, University of Copenhagen, 1998.