# Motion Retargeting for Crowd Simulation

Yann Pinczon Du Sel* Golaem
Nicolas Chaverou[θ] Golaem
Michaël Rouillé[#] Golaem

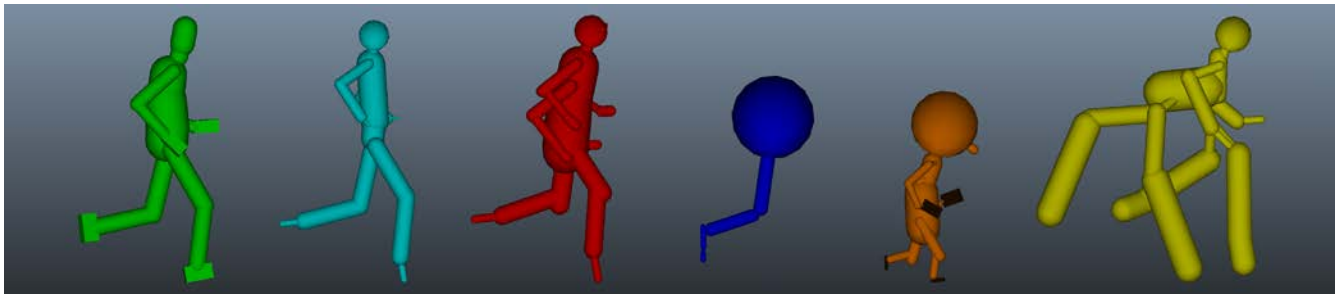

**Figure 1:** *Retargeting of the same run animation on different character morphologies.*

## Abstract

CG Crowds have become increasingly popular this last decade in the VFX industry: formerly reserved to only a few number of high end studios and blockbusters, crowd simulation is now widely used in TV shows or commercials. In a context where budget and time constraints increase at each project, quality requirements stay the same: generate believable character animations at the microscopic level while ensuring that the overall crowd simulation stays coherent at the macroscopic level.

Golaem Crowd is an artist-oriented crowd simulation plugin for Autodesk Maya. Amongst the various tools embedded in the software, it includes a dedicated character animation engine. Golaem Crowd animation engine allows real-time motion retargeting as well as interactive motion synchronization and blending. In this paper, the motion retargeting workflow is detailed, as well as its features and how it helps solving production constraints by reducing the number of assets to create and by facilitating the reuse of assets.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

**Keywords:** character animation, motion retargeting

## 1 Introduction

Motion retargeting is the process of adapting and adjusting a keyframed or captured motion to a character with a different skeleton topology. The target skeleton topology can include a different number of limbs or bones, different hierarchies or scaled segments.

*e-mail:yann@golaem.com
θe-mail:nicolas@golaem.com
#e-mail:michael@golaem.com


http://dx.doi.org/10.1145/2791261.2791264

In a crowd simulation context, motion retargeting has several benefits. A same motion can be used for several character morphologies helping to reduce the number of motions to capture, process or create. It also allows reusing motions created for a specific character, in a previous show, on a new character.

To benefit from motion retargeting, the Golaem Crowd animation engine relies on an intermediate skeleton representation, hereafter named *Golaem Skeleton*. This new skeleton representation, detailed in the next section, needs to be created for both the source motion data and the target character skeleton. Each converted motion data is saved as a .gmo file (**G**olaem **Mo**tion) and the skeleton as a .gch file (**G**olaem **Ch**aracter). The following figure provides an overview of the conversion workflow:

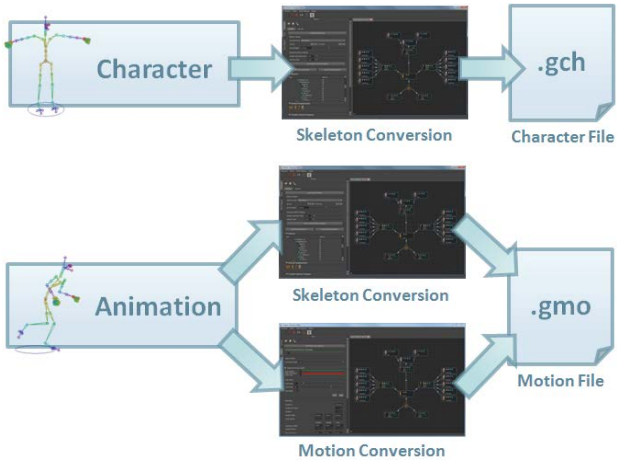


**Figure 2:** *Character and motion conversion workflow.*

Once the skeleton and motions are converted, a *Golaem Skeleton-Motion Mapping* must be defined to specify how to replay the source animation Golaem Skeleton on the target character Golaem Skeleton. This mapping can be automatically computed at simulation time or manually created and saved as a .gmm file (**G**olaem Skeleton-**M**otion **M**apping), as shown in Figure 3.

The Golaem motion retargeting process is inspired by the Maxis animation engine retargeting process developed for the Spores games [Hecker et al. 2008].

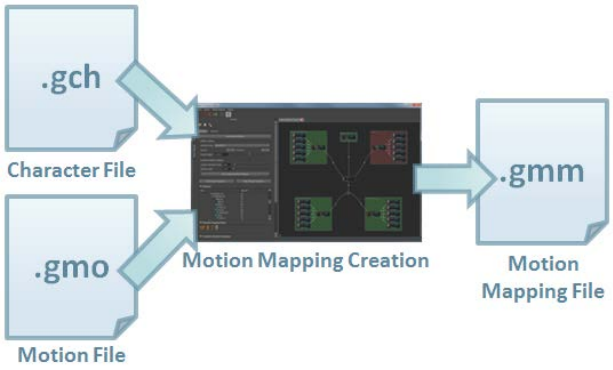


**Figure 3:** *Skeleton-motion mapping creation workflow.*

## 2 Golaem Skeleton

To retarget a motion on a character with a different morphology than the motion morphology, it is required to create a Golaem Skeleton of both the source motion skeleton and the target character skeleton. The Golaem Skeleton is based on a hierarchy of *Bone Chains* which defines the beginning and the end of important kinematics parts of the input skeleton (e.g. an arm will be described as a chain between the shoulder and the wrist joints).

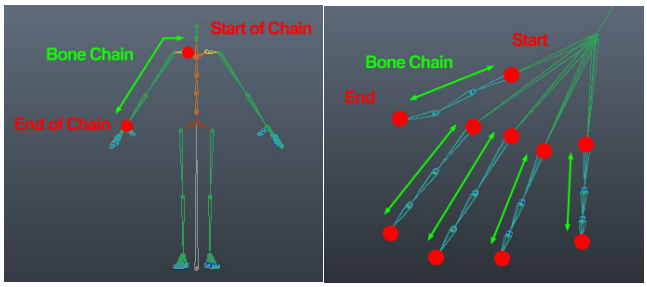


**Figure 4:** *Example of bone chains.*

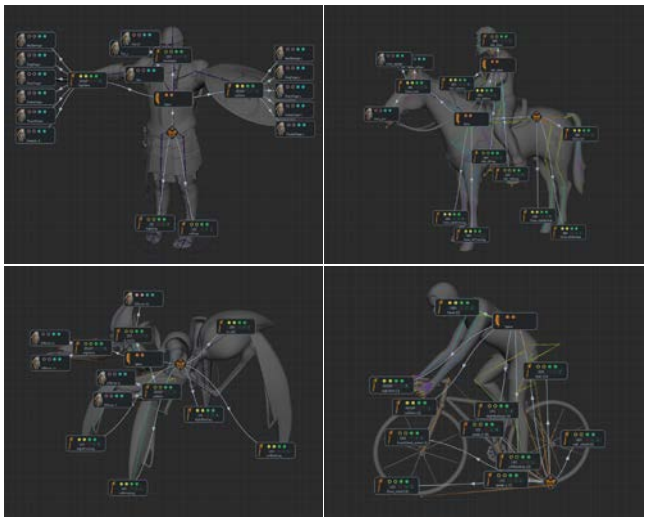

**Figure 5:** *Examples of characters converted to Golaem Skeleton.*

Bone chains can be defined with four different types of nodes:

- pelvis: root bone of the character (only one per Skeleton);
- spine: base structure on which limbs are connected;
- limb: main part of a body limbs (arms, legs, neck…);
- effector: extremity of a body limb (fingers, jaw…).

As shown in Figure 5, these node types are sufficient to represent any kind of character morphology: e.g. human biped, horse and rider, six-legged arachnid and even a completely-geared bike.

Since a Golaem Skeleton usually defines several limb nodes, those nodes need to be tagged to differentiate them from each other and help detect which limb animation from the source should be replayed on the target. Two tags are available:

- limb body position: front / center / back, bottom / center / up, left / center / right;
- limb type: grasper / leg / generic.

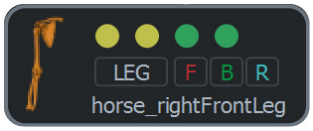


**Figure 6:** *Example of a tagged horse right front leg. (F) stands for Front, (B) for Bottom and (R) for Right.*

Finally, each limb and effector node stores extra information for the animation retargeting:

- inverse kinematics half-plane of the bone chain: half-plane in which the intermediate bones of the chain moves;
- mirror node: when retargeting a mirrored motion on a non-symmetrical skeleton, relevant limb / effector nodes have to be mapped to their mirror nodes in the Golaem Skeleton.

Due to the amount of information required to create a Golaem Skeleton for each character and motion, there was concerns that it could slow down the process of assets creation. Thus we developed an automatic creation process. Once a character or a motion skeleton is imported in the character authoring tool, it basically takes care of every step described previously:

1. limb nodes are detected first, by searching the isomorphic branches in the skeleton hierarchy tree. Limb position tags are determined based on the world position of each limb relatively to each other. Limb type is guessed depending on the limb world position (e.g. legs are close to the ground);
2. effector nodes are extracted from the limbs;
3. the pelvis node is the lowest joint in the hierarchy that has every limb joint as a direct or indirect child;
4. spine nodes are added and mapped to the joints that are between the pelvis and the beginning of each limb joint.

The automatic creation process has been tested on many different morphologies coming from various customers, and in most cases, the result was directly usable in production. When not usable, the result can be combined with the Conversion Quality tool, to provide a helpful working base for the Golaem Skeleton creation and point out which parts must be improved. By narrowing down which nodes need to be corrected, fixing the Golaem Skeleton usually takes less than an hour.


**Figure 7:** *Example of Conversion Quality tool feedback (here the Knee and the HipPartL require some position corrections).*

## 3 Golaem Skeleton-Motion Mapping

Once a motion and a character have been converted to a Golaem Skeleton, it is required to create a Golaem Skeleton-Motion Mapping for this tuple. The Skeleton-Motion Mapping defines the association between the source motion Golaem Skeleton nodes and the target character Golaem Skeleton nodes (e.g. Limb node *A* of the source motion will be replayed on the Limb node *A'* of the target character) and how to apply the motion data on the target skeleton, hereafter named *Movement Mode*.

### 3.1 Mapping

Combined with the Golaem Skeleton, the Golaem Skeleton-Motion Mapping is the cornerstone of Golaem Crowd animation modularity. The Skeleton-Motion Mapping allows a wide range of combinations of animations from different morphologies, as shown in Figure 8.

For example, a centaur can be animated using a horse animation on the lower part of the body combined with a human animation on the upper part; or a four-armed biped animated by using twice the arms animations of a regular biped.

Even if the Golaem Skeleton-Motion Mapping is required for each skeleton-motion combination, in most situations the source and the target skeletons are alike. Thus we developed an automatic Skeleton-Motion Mapping process based on the limb body position and type tags. The assignment is based on a score computed for each possible combination of limbs.

### 3.2 Movement Mode

The movement mode represents the meaning of the motion, i.e. what is important about it for expressing the desired intent [Hecker et al. 2008]. The movement mode is defined for each limb / effector node and is used to determine how the motion should be replayed when the *rest pose* (or neutral pose) of the source and the target differs. Two modes are available:

- Absolute mode: for each frame, the world orientation of the source animation is applied to the target skeleton, whatever the rest pose of each morphology (e.g. grasping an object with the arm);
- Rest Relative mode: for each frame, an offset between the animation and the rest pose of the source skeleton is computed and added to the rest pose of the target skeleton (e.g. mostly used for clavicles and heads nodes, especially when the source and the target rest poses have a significant different orientation).

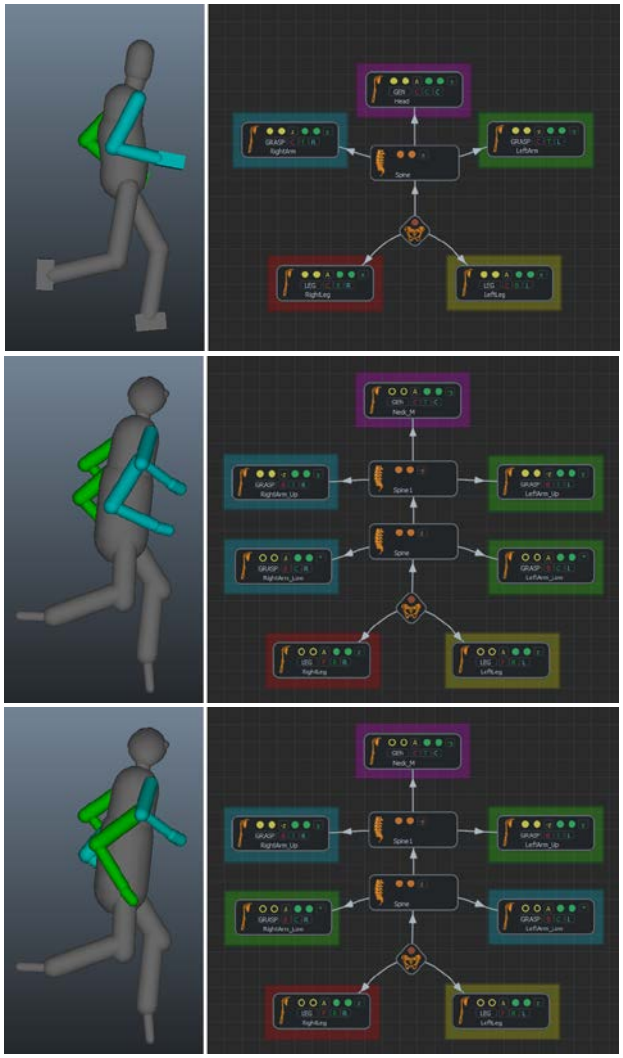

**Figure 8:** *Examples of different mappings of a regular biped animation applied on a four-armed biped.*

## 4 Animation Playback

Golaem Crowd animation engine use a blend tree system containing nodes for animation playback, blending with or without synchronization, ground adaptation [Edsall 2003]... As only the animation playback node is involved in the motion retargeting process, other nodes will not be detailed here.

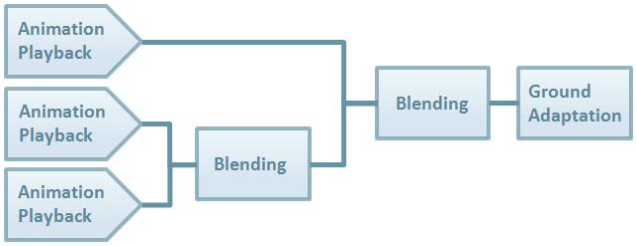


**Figure 9:** *Example of a Golaem Crowd blend tree.*

In a crowd simulation context, performances are critical. To be able to retarget thousands of characters in real-time, the playback system of the animation engine relies on a simplified representation of the Golaem Skeleton [Kulpa et al. 2005]. In this representation, every bone chain of the source and the target Golaem Skeletons is replaced by:

- an orientation representing the global direction of the bone chain, expressed as the direction between the last and first bones in the chain;
- a floating value representing the length of the bone chain.

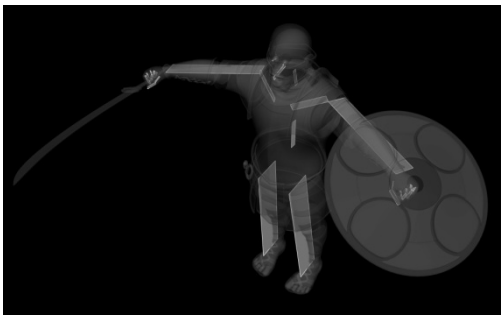

**Figure 10:** *Simplified representation of a Golaem Skeleton.*

The animation playback process is divided in four main steps:


**Figure 11:** *Animation playback process.*

Once the animation playback has been computed, the resulting posture is processed by other nodes of the animation engine.

## 4.1 Apply Skeleton-Motion Mapping

For each limb node of the target character, the engine checks the skeleton-motion mapping to determine which source animation limb should be played. The orientation and length of the limb bone chain is determined according to the Movement Mode.

- If the mode is set to absolute, then the source delta-from-rest-pose orientation is first added to the rest orientation of the target skeleton (to ensure the uniqueness of the final orientation), and then the difference of orientation with the source animation is computed, and added to the previous result.
- If the mode is rest relative, then the source delta-from-rest-pose orientation of the node is simply added to the rest orientation of the target skeleton.

## 4.2 Interpolation

The previous process is done twice, for two different frames of the animation, and the results are then interpolated together. This step allows the animation engine to run seamlessly with any source animations and simulation frame rates (e.g. a 12 fps animation can be used in a 100 fps shot with no animation glitch).

## 4.3 Mirror

Animation mirroring is optional. For each limb, the engine checks the target Golaem Skeleton to determine the mirror limb node. As the source animation has already been applied to the target skeleton at this step, it is not required to define the mirror nodes of the source animation Golaem Skeleton.

## 4.4 Inverse Kinematics

In the previous steps, the animation engine only processes a simplified representation of the Golaem Skeleton (i.e. lengths and orientations of each bone chain). Thus, an inverse kinematics (IK) computation step is required to convert this representation to a representation which can be used for skinning and render (e.g retrieve the position of the elbow in an arm chain). As this computation is CPU intensive, it is only done when needed, i.e. when characters are visible in the camera frustum or when the simulation results are cached.

Two different IK algorithms are used by the Golaem animation engine for two different use cases.

For the specific case of bone chains containing exactly three bones (e.g. an arm chain composed of a shoulder, an elbow and a wrist) an *optimized analytic solution* is used [Kulpa et al. 2005]. In this case, the positions of the two extremities are known from the simplified representation and only the position of the intermediate bone need to be computed. It is done by:

1. computing the intersection of the two spheres centered on each bone chain extremity and of radius determined by the length of each bone (e.g. shoulder-elbow and elbow-wrist for an arm chain). This intersection results in a circle;
2. the intermediate bone position is the intersection between this circle and the IK half-plane specified for this node in the Golaem Skeleton.

For bone chains with more than three bones (e.g. a finger chain) an implementation of the *FABRIK algorithm* [Aristidou and Lasenby, 2011] is used. This algorithm has been selected because it is simple to implement and it distributes orientations evenly among the bone chain (contrary to the Cyclic Coordinate Descent algorithm [Canutescu and Dunbrack, 2003]).

FABRIK is a two phases algorithm that starts by a forward reaching phase and is then followed by a backward reaching part which is repeated until the target is reached or the maximum iteration count is done.

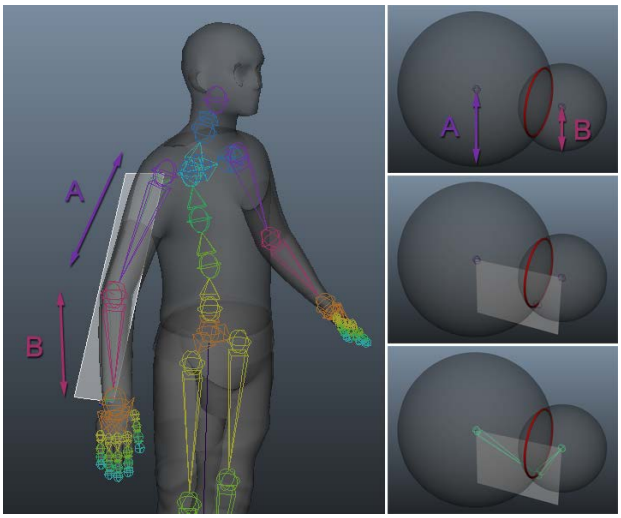


**Figure 12:** *Example of an elbow position computation using the optimized analytic solution.*

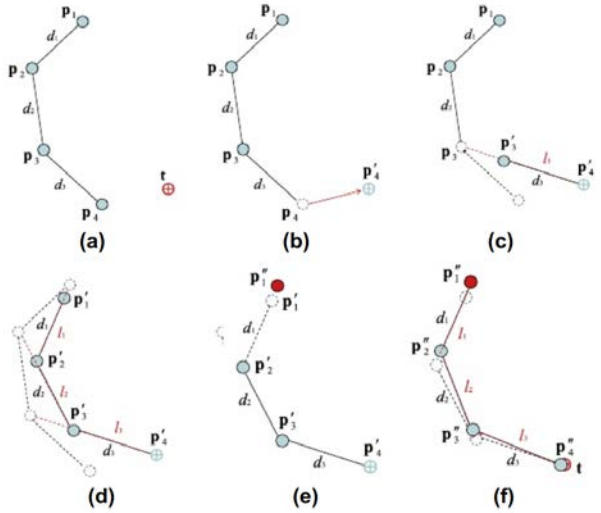


**Figure 13:** *Example of a full FABRIK iteration for a four bones chain. Image courtesy of [Aristidou and Lasenby, 2011].*

As this step is one of the highest CPU consuming tasks of the animation workflow, an IK cache has been developed to optimize simulation times. The IK Cache is computed for every FABRIK-eligible bone chain of every target Golaem Skeleton and allows using the best resolution result for a negligible cost at simulation time (the analytic solution has proven to be fast enough).

| | With IK Cache | Without IK Cache |
|---|---|---|
| FABRIK | 11% | 53% |
| Opt. Analytic | 2% | 1% |
| Blend Tree Update | 58% | 27% |
| Others | 29% | 19% |
| Sim. Update Time | 117ms | 212ms |

**Figure 14:** *Performances for a 5 000 characters simulation with and without IK Cache, on an Intel i7-3632QM @ 2.2Ghz CPU.*

The result of an IK resolution being the same for every orientation of a bone chain, it only caches the results of the IK algorithm for different sample lengths of the chain. To properly handle bone chain scale, the IK cache stores the ratio between the sample length and the maximum length of the bone chain. Thus, sample values range from 0 to 1.

During the simulation, the intermediate bones positions are determined by interpolating the value of the two nearest samples. Even if the quality of the IK resolution is closely linked to the number of samples, internal tests have shown that 51 samples, i.e. a sample every 2% of the bone chain length, is perceptually good enough.

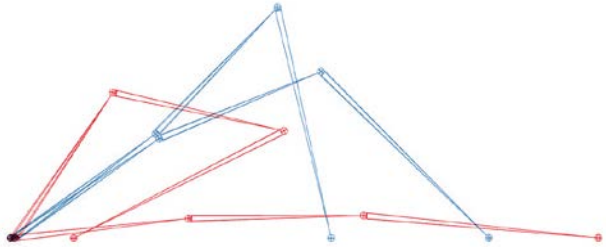

**Figure 15:** *Simplified example with two cached samples (in red), and the interpolated results (in blue)*

## 5 Limitations

Our motion retargeting system aims at re-using assets (and especially animations) from one project to another with the maximum fidelity, and therefore is not the best approach to produce different styles of animations depending on the characters. Playing a walk female animation on a male character will not change the feeling that is conveyed by the source animation.

Furthermore using an inverse kinematics algorithm to build the final bone posture may lead to some differences with the source animation in the case of complex bone chains (more than 4 bones in the chain, or a chain bending in several directions at the same time).

Most of the time, the loss of animation fidelity can be avoided by using a controllable IK Cache or a better suited configuration of the Golaem Skeleton (e.g. splitting a bone chain in two separate bone chains, each bending in a different direction).

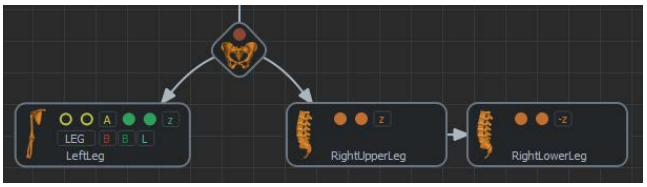


**Figure 16:** *Example of a leg splitted in two separate bone chains.*

## 6 Conclusion and Future Work

The Golaem Crowd Motion Retargeting workflow has been presented, from the Golaem Skeleton and Golaem Skeleton-Motion Mapping conversions to the Animation Playback. By introducing a simplified internal skeleton representation, the two different types of IK algorithms that are used, and in particular the

IK cache optimization, we purposefully focused on performances, a staple of crowd simulation.

However it is important to note that even if the IK cache was originally developed to improve performances, one of its main advantages is also that it is totally independent from the IK algorithm that is used to fill it up. So, a complementary way of using it would be to feed it with artist made animations, and thus create an *artistic-driven IK cache* that would offer better controllable results.

## References


ARISTIDOU, A., AND LASENBY, J. 2011. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. *Graphical Models* 73, 5, 243-260.

CANUTESCU, A. A., AND DUNBRACK, R. L. 2006. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Science : A Publication of the Protein Society* 12(5), 963–972.

EDSALL, J. 2003. Animation blending: Achieving inverse kinematics and more. *Gamasutra*, Jul, 4.

HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND PROOIJEN, K. 2008. Real-time Motion Retargeting to Highly Varied User-Created Morphologies. *ACM Trans. Graph.* 27, 3, (August), 27:1--27:11.

KULPA, R., MULTON, F., AND ARNALDI, B. 2005. Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum* 24, 3, (September), 343-351.