Keyframe interpolation with self-collision avoidance

Jean-Christophe Nebel
University of Glasgow
Computer Science Department
Glasgow, G12 8QQ, UK

Abstract. 3D-keyframe animation is a popular method for animating articulated figures. It allows artistic expressiveness by providing control to the animator. The drawback of this process is that it requires significant effort from the animator. Recently, work has focused on high level techniques such as adapting reference movements. However, whatever the way the animation is produced, the final process is an interpolation between keyframes. Our problem is that these interpolations do not deal with the avoidance of collisions between the limbs of an articulated figure, either an animator has to add new keyframes or the motion produced contains unrealistic positions. In this paper we present a new interpolation method producing self-collision free paths based on geometrical properties. Our method is a high level interpolation in which any classical interpolation method can be used. Experimental results using a human model show that the animator can reduce the level of detail needed for describing a movement and still get realistic results at interactive speeds.

1 Introduction

3D-keyframe animation derives from the principles of traditional animation that were mainly developed in the 1930's at the Walt Disney studios. The animator defines a movement by providing a set of poses, then the motion is obtained by simply interpolating between these keyframes. Despite the fact that it is quite an old technique and many other animation methods exist now such as physics-based or behavioral methods [17], it is still a popular method for animating articulated figures. It allows artistic expressiveness by providing control to the animator and only geometrical properties are needed. It is used through a very wide range of applications from computer aided choreographic design [7] to animation of volumetric objects ("Visible Human Walk" [22]).

The drawback of this process is that it requires significant effort from the animator. Defining a movement often requires a high level of detail to ensure that the interpolation curves induce the desired motion. In order to ease this task, a few tools have been offered for building poses such as forward kinematics, inverse kinematics, emotional posturing [8] and genetic algorithms [9]. Recently, work has focused on high level techniques such as adapting reference movements obtained either by keyframing or motion capture. The higher level of control provided reduces the user's load of direct specifications for the desired movement. Two approaches have been followed: interpolation between keyframes of

reference motions [10], [5] or derivation of a motion from a reference motion by adding emotions or behaviours to keyframes [19],[8].

However, whatever the way the animation is produced, the final process is an interpolation between keyframes typically using cubic splines. In order to have more realistic motions, interpolations controlled using kinematics [16], [13] or dynamic constraints [15], [2] have been proposed. Our problem is that these interpolations do not deal with the avoidance of collisions between the limbs of an articulated figure. If low level keyframe animation is created interactively, the animator can avoid self-collisions at the cost of adding new keyframes which appear to be necessary for a realistic interpolation. In the case of the adaptation of a reference motion, either the animator can correct the frames produced or if the animation is created automatically for virtual reality applications [19], the produced motion may contain unrealistic positions. The purpose of this research work is to offer a new interpolation method producing self-collision free paths which frees animators from interpolation considerations and leads to realistic motions when animations are generated automatically.

Section 2 relates our efforts to previous work. Section 3 gives the principle of our interpolation methods. Section 4 describes the details of our algorithm. Finally, Section 5 presents and discusses some experimental results.

2 Related work

When multiple body parts or several objects are animated it is possible for them to collide and interpenetrate. This is often an undesired situation. This problem has been addressed in many different ways.

In the case of animations that are produced using physical simulations, once collisions are detected, reactions to these collisions are computed. This has been efficiently implemented for the dynamical control of clothing animation [18] and the simulation of rigid object collision [3]. Despite the fact they deal with responding to collisions whereas we want to avoid them, these works may be useful to us because they provide very efficient collision detection algorithms.

Combining spatial and joint constraints into collision free motion is a fundamental goal of robotics - motion planning - and a variety of exact and heuristic solutions exist. The problem is that the complexity grows exponentially with the number of degrees of freedom, making exact solutions effectively impossible on a figure with the articulation of a simulated human [6]. Much progress has been made, however, in producing fast planners by considering schemes that are not *complete*, i.e., may fail to find a path when one exists, or probabilistically complete, i.e., find a solution in bounded time with high probability [4]. Koga [20] has offered an incomplete planner to deal with finding a collision free path for grasping an object with articulated arms. However they assume that arms are in such a configuration that they cannot obstruct each other's path and results are not provided at interactive speed.

In the field of inverse kinematics, some work is more similar to ours. They deal with achieving collision avoidance while articulated figures - virtual humans - are reaching a goal [21], [11]. Because of the nature of the inverse kinemat-

ics algorithm, sensors are used for the collision detection. Response vectors are calculated and integrated into the inverse kinematics equation system. There are fundamental differences between this approach and ours. At first, inverse kinematics has as a goal to bring a particular end-effector to a particular place, whereas in keyframing all limbs must match the keyframe provided. Secondly, the algorithms proposed are progressive: each time a collision is detected, an intermediate goal is created. The motion goes from intermediate goals to intermediate goals, which does not ensure a coherent or realistic motion. Finally the way collisions are detected cannot assure a self-collision free path in any case.

3 Principle

In this research an articulated body is defined as a hierarchy of rotational joints each of them having up to three degrees of freedom and being limited to produce only believable postures. Keyframes can be either created by an animator or selected from previous motions. Once keyframes have been specified, interpolations are achieved to produce the animation. The task we deal with is to offer an interpolation algorithm which generates self-collision free motions.

We want the interpolated motions to be produced at interactive speed. In this way it could be used either as an interactive tool allowing an animator modifying some frames, if the suggested interpolation does not fit exactly their expectations, or as an automatic generator of self-collision free motion in virtual reality applications. For these purposes, we propose an *incomplete* algorithm based on geometrical properties. It generates automatically some of the keyframes that the animator previously had to add manually, this with a speed which would globally reduce the time the animator has to spend producing a self-collision free animation.

Input: K_0,K_1, keyframes specified at the time steps T_0, T_1
Algorithm:

CheckAndCorrect($(K_0, T_0), (K_1, T_1)$ *)*
{
  Interpolate between K_0 *and* K_1
  Get for each time step between T_0 *and* T_1 *the list of self-collisions*
  If(self-collision)
    Select the first collision to be corrected $(T_{Collision}, ColType)$
    Move the limbs at $T_{Collision}$ *in order to correct the collision*
    Create a sub-keyframe $(K_{Collision}, T_{Collision})$
    CheckAndCorrect($(K_0, T_0), (K_{Collision}, T_{Collision})$ *)*
    CheckAndCorrect($(K_{Collision}, T_{Collision}), (K_1, T_1)$ *)*
}

Fig. 1. Production of a self-collision free motion between two keyframes

The principle of our scheme is to compute a first interpolation using any classical inbetweening method. Then, self-collisions are detected and sorted. At some collision times, frames are modified automatically to generate self-collision free sub-keyframes using geometric properties. Finally, these sub-keyframes are used for a new classical interpolation. This process continues until a self-collision free

interpolation is obtained (see algorithm Fig. 1).

This algorithm does not make any hypothesis about the kind of objects which compose the articulated figure, only a rigid skeleton is needed. So whatever the way the figure is defined, if an associated collision detection algorithm is available our scheme can be used to generate animations without self-collisions.

4 Implementation

In order to apply the principle described, we define the way of ordering the tasks of intersection correction in order to create sub-keyframes at relevant time steps. Then we explain the process by which intersections are corrected at a given time. Finally we give a detailed example of how our algorithm works on a basic interpolation case.

4.1 Production of a sub-keyframe

In order to correct intersections, we use the concept of *member*. A member is defined as a list of consecutive limbs starting from the centre of a figure and finishing at a limb that does not have any children. For example, the limbs of a humanoid are divided in 5 members: from the waist to each foot, from the waist to each hand and from the waist to the head. The trunk is then shared by 3 different members.

Once the first interpolation round has been performed between two keyframes, collisions between limbs are detected. They are grouped: for each pair of intersecting limbs, the members involved and intersection period are computed. The first intersection to correct has to be chosen, it has to be a dominant intersection whose correction suppresses a large number of collisions. Practically, all the intersection periods are scanned and the longest one is selected. The frame whose time index is at the centre of the period is the first frame to be corrected. The member that has to be moved to correct it has to be selected. If two members are involved in the intersection, the member that has moved the most between the previous and the next frame is chosen, since a large movement allows more freedom for correcting an intersection.

To move this member efficiently, we need to know which of its limbs should be moved first for the correction - *starting limb* -, and which part of the member should stay immobile - *fixed limbs* -, because its motion would not have any effect on the correction of the intersection. If the self-collision involves only one member the part between the waist and the first limb intersected should be fixed. Otherwise, the list of fixed limbs is composed of the limbs between the waist and the last common limb. In both cases the limb involved in the collision that is the closest to the extremity of the member should be moved first. In fact the motion of any limb between this chosen limb and the first not fixed limb may correct the collision, moreover this choice has been made in order to try first the motion of the limb which would change the position of the member the least.

Before changing the position of the member we look for any related intersections at the same time in order to correct them during the same step. If

there are other intersections with the selected member, starting limbs and fixed limbs are calculated for each of these intersections. Then the new starting limb and the new list of fixed limbs are computed: the new starting limb is the starting limb which is the closest to the waist and the new list of fixed limbs is the shortest of the computed lists.

Once the member is selected and the starting and fixed limbs are known, the computation of a new pose of the member can start. The process begins with the motion of the starting limb. New positions for the end of the limb are computed (see Section 4.2). For each of these new values, the limb is moved and intersections between the limb and others are detected. As soon as a position is valid - no intersection is detected -, the next step of the process is to search for a valid position for the next child limb of the member. When the last limb of the member has been validated the member is collision free and the associated sub-keyframe is produced. An outline of the algorithm is given Fig. 2.

Input: *start*, index of the starting limb
fixed, highest index in the list of fixed limbs
Output: Boolean (successful or failed)
Initialise: *member*, member to be moved
Algorithm:

```
Move(fixed, start)
{
    limb = member.GetLimb(start)
    limb→GetPositions()
    OrderPositions()
    For each position
        limb→MoveToPosition()
        If(limb→DoesNotIntersect())
            for(i =start+1 to member.GetNbLimbs())
                limb2 = member.GetLimb(i)
                If(limb2→DoesIntersect())
                    If(Move(start, i)==FAILED)
                        Break
            if(i==member.GetNbLimbs())
              return SUCCESSFUL
    start--
    limb→MoveBackToStartingPosition()
    if(start==fixed)
        return FAILED
    else
        return Move(fixed,start)
}
```

Fig. 2. Production of a collision free position for a member

However, if at any step no valid position may be found, new positions are computed for the parent limb, which becomes the starting limb. If the new starting limb belongs to the list of fixed limbs, no move can be performed. In this case the process has finished and has failed. No sub-keyframe is produced.

Once the sub-keyframe has been produced, the whole figure is checked for self-collisions again. If there are remaining intersections - between other members -, they are sorted and the new intersection to be corrected is chosen. Then the sub-keyframe is modified accordingly to the process described above.

4.2 Production of new positions

In order to produce collision free sub-keyframes, limbs are moved to new positions. The realism of the interpolation produced depends on the generation of these positions. At first we assume that the interpolation algorithm or the animation package used by the animator has been carefully chosen to provide motions suitable for their purpose. Hence the collision free motion generated should keep most of the properties of the motion interpolated by the chosen algorithm. In order to do this, any newly generated position of a limb should be closely related to the previously interpolated curve.

Secondly, the keyframe positions cannot be modified. They are absolute constraints provided by the animator to express the motion they expect. When limbs intersect obstacles during their motions, these obstacles have to be passed, the limbs cannot stay on one of their sides.

A limb may be defined by two positions: the beginning of the limb, B, which does not move and the end of the limb, E, which should be moved to correct a detected collision. In order to keep the new suggested positions of the limb connected with the previously computed interpolation curve, these new positions depend on positions of the interpolation curve. Actually, they are obtained using the position of the limb at the time step at which it has to be corrected, (B, E), the position of the limb at the previous time step, (B_0, E_0), and the position of the limb at the next time step, (B_1, E_1), see Fig. 3.

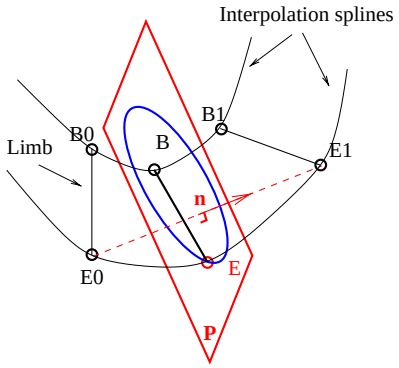


Fig. 3. New positions generated by the plane P (here an ellipse)

Because we want the obstacle to be passed, we assert the constraint that the new position should be in the space between the planes, $P_0(\overrightarrow{E_0E_1}, E_0)$ and $P_1(\overrightarrow{E_0E_1}, E_1)$, perpendicular to $\overrightarrow{E_0E_1}$ and passing through E_0 and E_1 respectively. Practically, we build a first plane perpendicular to $\overrightarrow{E_0E_1}$ and passing through E, $P(\overrightarrow{E_0E_1}, E)$ and then build other parallel planes between E_1 and E_0. The figure has a rigid skeleton and hence the distance between the beginning of the limb, B, and the end, E, is constant, moreover the beginning is fixed, so the positions should belong to the sphere $S(B, \overrightarrow{BE})$.

The space of new positions is the intersection between these planes and

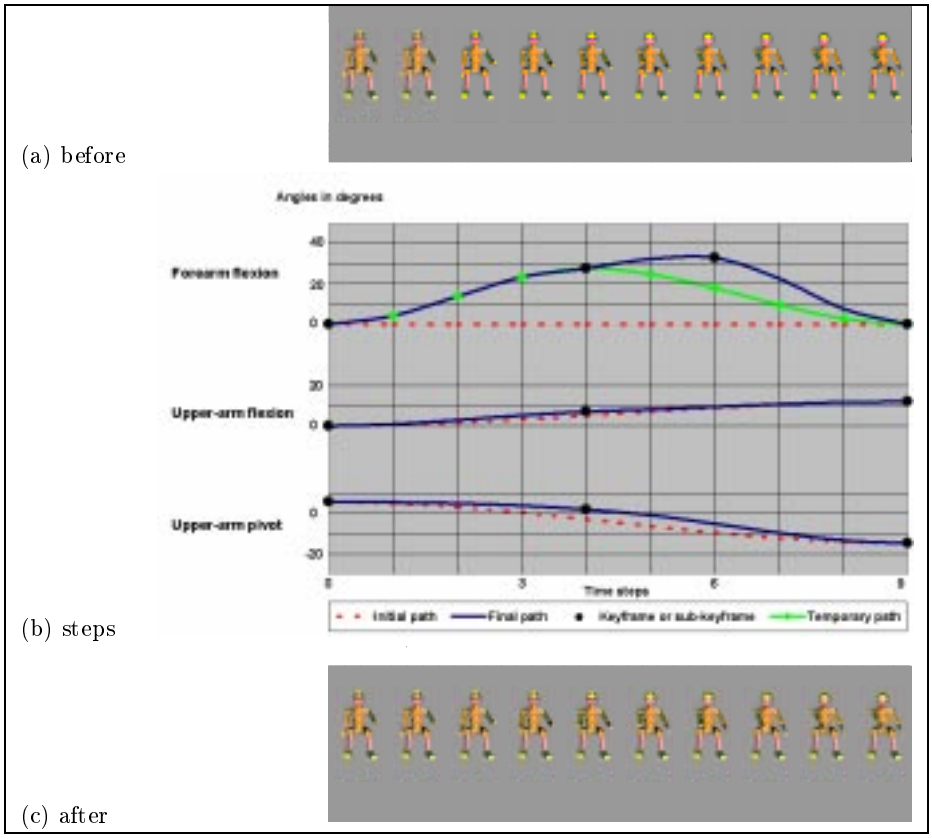


Table 1. Position of limbs during the interpolation process

this sphere: a set of ellipses, circles or points. A sampling of this space is then made and these positions are sorted to offer first positions that will provide the most realistic motion. In order to do this sorting, weights are assigned to each position according to parameters relevant to the expected animation such as degrees of freedom, comfort [12], distance to the interpolated position, E, or probability of belonging to a path defined by physiological models [20].

4.3 Detailed example

Here we study in detail the results provided by our algorithm during the production of a self-collision free path between two keyframes. The articulated figure used to evaluate our system is a humanoid composed of 19 limbs and 15 joints. All interpolations are made using a classical cubic spline scheme. The production of poses of our articulated figure can be achieved in real-time by three different ways: forward kinematics, inverse kinematics and interactive genetic algorithms. The time spent for building a pose has been evaluated for these techniques [9]: a trained animator needs between 30 and 60 seconds to produce a given pose

Interpolation between steps 0 & 9			**Interpolation between steps 4 & 9**		
Time step	Intersections		Time step	Intersections	
3	R. forearm	R. hip	5	R. forearm	R. thigh
	R. forearm	R. thigh		R. hand	R. thigh
4	R. forearm	R. hip	6	R. forearm	R. thigh
	R. forearm	R. thigh		R. hand	R. thigh
	R. hand	R. thigh	7	R. forearm	R. thigh
5	R. forearm	R. hip			
	R. forearm	R. thigh			
	R. hand	R. thigh			
6	R. forearm	R. thigh			

Table 2. Self-collisions detected during interpolations

whatever the method.

We start by interpolating between the two keyframes defined at the time steps 0 and 9. As expected, the result is everything but realistic (see Table 1.a). Actually 9 collisions are detected (see Table 2). Dominant collisions appear to be at time step 4. The starting limb is the right forearm and only the waist is a fixed limb. The motion of the right forearm does not provide any valid positions, so the correction of the collisions starts by the motion of the right upper-arm and then the right forearm is moved (see Table 1.b). The right hand keeps its previous position because its position is collision free. Eventually a new sub-keyframe is created at time step 4.

New interpolations are made between the new sub-keyframe and each keyframe. Between time steps 0 and 4 no collision occurs, but between time steps 4 and 9, 5 collisions are detected (see Table 2). Self-collisions are corrected at time step 6 by the motion of the right forearm (see Table 1.b), which was the starting limb whereas the waist was fixed. A new sub-keyframe is created.

Finally, interpolations between time steps 4 and 6 and between time steps 6 and 9 are made. No collision can be detected. The animation is now collision free (see Table 1.c), only 13 seconds were needed for its generation.

5 Results and discussion

We tested our algorithm using an articulated figure whose skeleton is described in Section 4.3 using a 300 MHz Ultra2 processor of a Sparc 450 workstation.

One of our aims is to free animators from interpolation considerations and to let them to concentrate on what really matters: the creation process. We do not want them to focus on avoiding collisions between the limbs of their articulated figure by adding irrelevant keyframes from the point of view of the creation, which appear to be necessary for realistic interpolation. In order to present a practical example of the use of our method, we used a set of positions provided in a yoga book [1] which is supposed to demonstrate how to reach the lotus posture. These positions have been used as keyframes to produce the animation of a character moving from a position where it is knelt to the lotus posture (see Appendix). This animation of 17 frames is based on 5 keyframes.

Because of the nature of this yoga posture, 25 self-collisions have been detected after using a classical interpolation algorithm. An animator would have a very difficult task adding all the keyframes that would have been needed for a self-collision free motion. On the other hand, our algorithm produced automatically in 43 seconds a collision free motion creating 4 sub-keyframes.

The animator can then judge the result. They may decide to keep the motion generated (see Appendix) or to modify some of the frames produced. In both cases our algorithm eases the task of the animator and is time saving, considering that the time needed by a trained animator for building these 4 poses would have been at least 120 seconds [9].

Previous results [14] have shown the interpolations between two keyframes are produced at interactive speed for the generation of up to 20 frames. In spite of the different qualities of animation which have been obtained, our method has a positive impact globally. The speed-up for getting a collision free motion is between 3 to 20 if no modifications are needed, or between 1.3 and 2.8 if only minor corrections are needed. If the algorithm is unable to build a collision free motion, this result is given almost instantly (speed-up down to 0.8) and information about the self-collisions detected is provided. Knowing this, the animator can then add some new keyframes at specific times.

Actually our algorithm appears to be a useful tool for keyframe animation when an animator is involved. Interactively, the animator can decide if the offered interpolation should be kept, modified or if more keyframes should be provided. This algorithm spares a lot of time for the animator, eases their task and allows them to focus more on the creation process.

Finally another advantage of our method is its generality. Any kind of classical interpolation scheme can be used. It may be used with articulated figures having rigid or deformable objects around a rigid skeleton, and may also be used also to avoid collisions with the environment.

6 Conclusion and future work

We presented a new algorithm for keyframe interpolation whose aim is to generate self-collision free animations. Based on geometrical properties, it is general considering that any kind of low level interpolation scheme may be used and it can deal with articulated figures composed of rigid and deformable objects. Experimental results using a human model show that with our algorithm, the animator can generally reduce the level of detail needed for describing a movement and still get realistic motions at interactive speed. It appears it is time saving and provides a real help assisting animators in the task of producing realistic animations. We think it could also be used for the automatic generation of animations for virtual reality applications, where the guarantee of collision free motions does not need to be absolute. In order to improve the quality of the animation produced, we are currently working on metrics which would express the quality of the interpolated motion and on other ways of selecting the best position among the positions produced. We will also implement some optimisations in order to provide results at near real-time.

Acknowledgements

We gratefully acknowledge the support of the European Union's Training and Mobility of Researchers (TMR) programme in funding this work.

References

[1] *Yoga, the Iyengar way*, pages 54–55. Dorling Kindersley, 1990.

[2] K. Arai. Keyframe animation of articulated figures using partial dynamics. In N. Magnenat-Thalmann and D. Thalmann, editors, *Models and techniques in computer animation*, pages 243–256. Springer-Verlag, 1993.

[3] S. Bandi and D. Thalmann. An adaptive spatial subdivision of the object space for fast collision of animated rigid bodies. In *Eurographics'95*, pages 259–270, 1995.

[4] J. Barraquand, L. Kavraki, J-C. Latombe, T-Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *Journal of robotics research*, 16(6):759–774, 1997.

[5] M. F. Cohen C. Rose and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comp. Graphics and application*, 18(5):32–40, 1998.

[6] J.F. Canny. *The complexity of robot motion planning*. MIT Press, 1988.

[7] J. Lopez D. Rodriguez D. Meziat M. Carbajo A. Casillas and J. L. Bosque. A user interface for the design of human figures multimedia animations. In *HIM97*.

[8] D. J. Densley and P. J. Willis. Emotional posturing: A method towards achieving emotional figure animation. In *Computer Animation 97*, pages 8–14, 1997.

[9] S. Etienne. *Positioning articulated figures*. PhD thesis, University of Glasgow, 1998.

[10] S. Guo and J. Roberge. A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Computer animation and simulation'96*, pages 95–107. Springer Computer Science, 1996.

[11] Z. Huang. *Motion control for human animation*. PhD thesis, EPFL-DI-LIG, 1996.

[12] H. Ko and N. I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and application*, 16:50–59, 1996.

[13] Z. Liu and M. F. Cohen. Keyframe motion imization by relaxing speed and timing. In *1995 Eurographics Workshop on Animation*, Maastrich, Holland, 1995.

[14] J.-C. Nebel. Keyframe animation of articulated figures using autocollision-free interpolation. In *17th Eurographics UK Conference'99*, 1999.

[15] X. Pintado and E. Fiume. Grafield: Field-directed dynamic splines for interactive motion control. In *Eurographics'88*, pages 43–54, 1988.

[16] S. N. Steketee and N. I. Badler. Parametric keyframe interpolation incorporating kinetic adjustement and phrasing control. In *Siggraph'85*, volume 19, pages 255–262, San Francisco, 1985.

[17] N. Magnenat Thalmann and D. Thalmann. Complex models for animating synthetic actors. *IEEE Computer Graphics and Applications*, 11, September 1991.

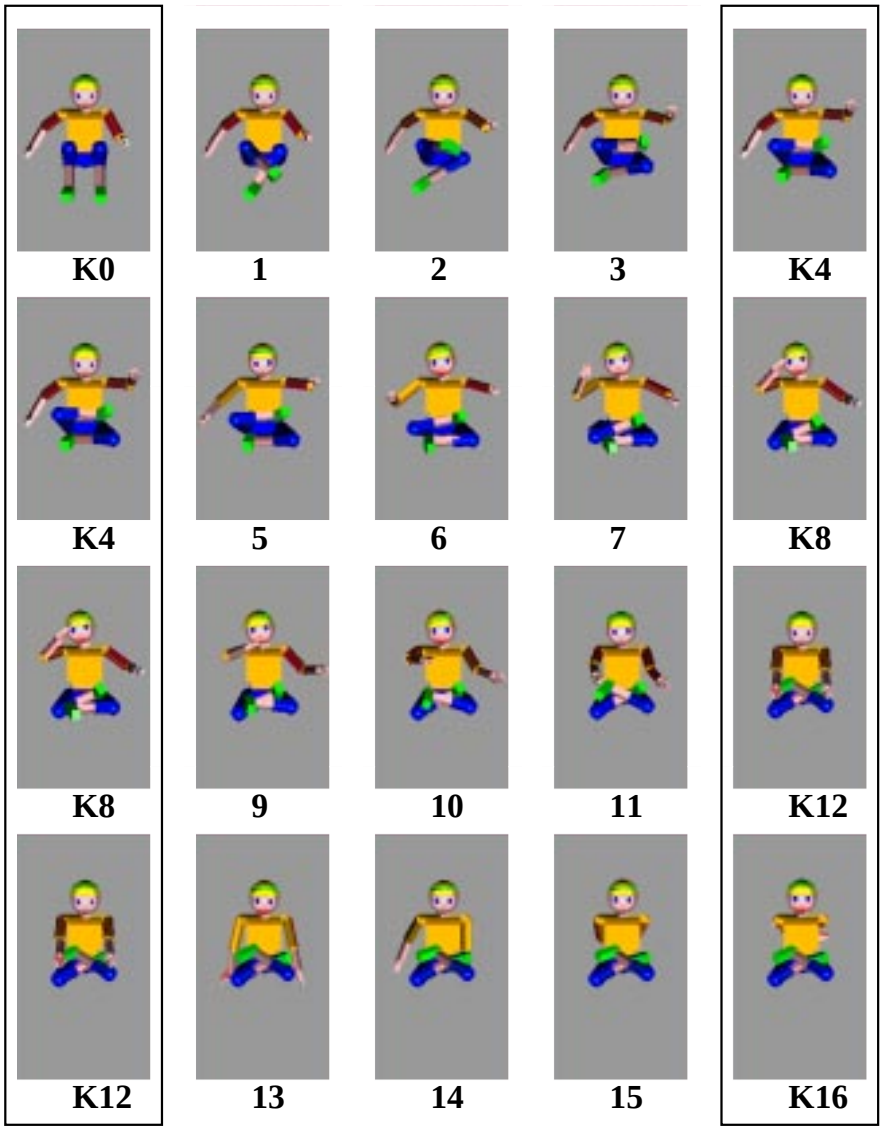
[18] P. Volino, N. Magnenat-Thalmann, S. Jianhua, and D. Thalmann. Collision and self-collision detection: Robust and efficient techniques for highly deformable surfaces. *Eurographics Workshop on Animation and Simulation*, 1995.

[19] D. J. Wiley and J.K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and application*, 17(6):39–45, 1997.

[20] J. Kuffner Y. Koga, K. Kongo and J.-C. Latombe. Planning motions with intensions. In *SIGGRAPH'94*, pages 395–408, 1994.

[21] J. Zhao and N. I. Badler. Interactive body awareness. *Computer-aided design*, 26(12):861–867, 1994.

[22] W. Zhongke and E. C Prakash. Visible human walk: Bringing life back to the dead body. In *International Workshop on Volume Graphics*, pages 247–356, Swansea, UK, 1999.



Frames of the animation of the lotus posture (K: keyframe)