ELSA: Euler-Lagrange Skeletal Animations - novel and fast motion model applicable to VR/AR devices

Kamil Wereszczyński[1][0000−0003−1686−472X], Agnieszka Michalczuk[1][0000−0002−8963−1030], PawełFoszner[1][0000−0001−5491−9096], Dominik Golba[2], Michał Cogiel[2], and MichałStaniszewski[1][0000−0001−9659−7451]

[1] Department of Computer Graphics, Vision and Digital Systems, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Akademicka 2A, 44-100 Gliwice, Poland {kamil.wereszczynski, michal.staniszewski}@polsl.pl

[2] KP Labs, Gliwice, Poland

Abstract. Euler Lagrange Skeletal Animation (ELSA) is the novel and fast model for skeletal animation, based on the Euler Lagrange equations of motion and configuration and phase space notion. Single joint's animation is an integral curve in the vector field generated by those PDEs. Considering the point in the phase space belonging to the animation at current time, by adding the vector pinned to this point and multiplied by the elapsed time, one can designate the new point in the phase space. It defines the state, especially the position (or rotation) of the joint after this time elapses. Starting at time 0 and repeating this procedure N times, there is obtained the approximation, and if the $N \to \infty$ the integral curve itself. Applying above, to all joint in the skeletal model constitutes ELSA.

Keywords: Skeletal animation · Euler Lagrange equation · Partial differential equation · Computer Animation · Key-frame animation

1 Introduction

Automatic animation generation in 3D computer graphics is used in many applications, e.g. Min et al. in [32] for generation of human animation, Spanlang et al. in [38] for mapping the motion obtained from aquisition system for avatar and robots or Li et al. in [28] for skinning the body of the skeletal models. Currently, the process of creating an animation is based primarily on remembering key poses for the skeletal model. The key-poses are made by animator, could be obtained from the motion acquisition systems or (in fact in most general cases) are the mixture of this to technique. This method of storing the skeletal animation applies more general technique of *keyframe animation*, described e.g. by Burtnyk and Wein in [8], Catmul in [10] or Parent in [35] (sec 3.5) It is driven from hand-made animation (e.g. Sito and Whitaker in book „Timing for animation" [37]). The remaining poses are joined with different methods of interpolation e.g. Ali Khan and Sarfraz in [4] or Mukundan in [33]. Animations can be generated manually by appropriate software or by external tools such as Motion Capture [5], IMU [25] or markerless systems [43]. In any case, the volume of data that must be generated and saved in order to play the animation is a big limitation. There is also the problem of repeating animations and combining them. It is not always possible to combine animations at any point in time and it is necessary to wait until a previous animation ends. Therefore, the presented work proposes a new approach for animation generation, which is based on the connection of the articulated joints with the partial differential equations (PDE) Euler-Lagrange in the form of the ELSA algorithm: Euler-Lagrange Skeletal Animations. Potential applications in computer graphics, virtual (VR) and augmented reality (AR) systems were also indicated.

1.1 State of the art

One of the possible applications of animation bases on human motion. Lobão et al. in [29] define two types of animations in that context: keyframed animation and skeletal animation (ibidem, pg. 299). The first name can be confusing, because, as they themselves wrote (ibidem, pg. 301), skeletal animation is also based on the key frames. The main difference lays not in the technique of the animation but the object that is animated: in the first case whole mesh is stored in the key frames, while in the second one - only articulated model, called *skeleton*, in which the whole process is based on a skeleton consisting of a combination of rigid bones and joints that connects those ones. Therefore we will name those techniques relatively: *mesh animation* and *skeletal animation*.
Currently the skeletal animation is applied in 3D computer graphics to animate articulated characters and enables to transfer the pose of a virtual skeleton into the surface mesh of a model. The skeletal animation technique can be divided into four parts: rigging, weighting, pose selection and skinning [6, 3]. An extension of the classical skeletal animation pipeline was introduced by Sujar et al. in [41], which relies on dealing also with the internal tissue of a model and allows to adapt a virtual anatomy of a patient to any desired pose by application of the patient's bone and their skins. The animation of complex data can be based on low-dimensional position systems. A popular way of reaching that goal relies on recent marker-based animation methods described by Krayevoy and Sheffer in [23] or Stoll et al. in [40] and on concept of shape deformation methods, in which markers are used as control points in the character's mesh animation e.g. in works of: Zollhöfer et al. [47], Zhao and Liu [45] or Levi and Gotsman [27]. The whole animation pipeline basing on low number of positional constraints was presented by Le Naour et al. in [26] in the course of potential loss of precision and position information. The problem was solved by application of efficient deformation algorithm and an iterative optimization method.
Another approach dealing with skeletal animation implies 4-point subdivision to fulfill the whole animation frames [46], where instead of classical interpolation

methods, an adaptive and high-performance algorithm of additional subdivision was used to create in-between frames in the skeletal animation. In the animation of motion, the cubic Cardinal spline can be used [21] in order to interpolate for each segment piece-wise cubicals with given endpoint tangents. Automatic skinning and animation of skeletal models can be achieved by not only dividing a character models into segments but also by subdividing each segment into several chunks [28] and introducing energy terms and deformation techniques. Correctly generated animation can be also applied for avatars and robots. In some examples, the robot's animation can be generated and modelled in a 3D program and dedicated software is prepared for translating 3D model to the physical robot [17]. For the purpose of the real-time whole-body motion of avatars and robots the issue of remapping the tracked motion was presented [38]. In the following approach, the tracked motion of the avatar or robot was combined with motions visually close to the tracked object.
Alternatively, animations can be generated on the base of mechanical rules. One of possible solution was presented in [22] relying on [19] by application of the diagonalized Lagrangian equations of motion for multi-body open kinematic chains with assumed N degrees of freedom [18]. The given methods focused on the introduced equations of motion, required diagnosing transformations and on the physical interpretation. In this case, diagonalization means that at each fixed point in time, the equation at each joint is out from all other joint equations. The *Partial Differential Equation* PDE (e.g. Evans in [13]) can be used for modelling the physics's phenomena within the motion models as well (e.g. Courant and Hilbert in [11]). This notion was utilized by Castro et al. in [9] for the cyclic animations by joining of mechanical description by PDEs and skeletal system. In order to define the boundary conditions, it is required to create the surface of a model from a set of predefined curves enabling solution of a number of PDEs. The first of presented approaches attached the set of curves into a skeletal model for the purpose of the animation holding along cyclic motion by use of a set of mathematical expressions. The second proposition implies the mathematically manipulated spine connected with the analytic solution of the PDEs treated as a driving mechanism in order to obtain a cyclic animation. The generated animation can be used in two different ways, as a surface structure of a model based on the PDEs or as an original model by means of a point to point map.
One of the widespread standard for exchanging the skeletal animations is *Biovision Hierarchy* - BVH (e.g. Dai et al. in [12]), which contains the bones' offset and the hierarchy of the model in the *header* and the Cardan angles (e.g. Tupling and Pierrynowski in [42]) for each joint in the *data description* section. The alternative are binary files: C3D (documentation in [1]) used e.g. by Madery et al. in [30] for creation of theirs kit database of human motion recorded on Kinect; or the *Acclaim Motion Capture (AMC)* described by Geroch in [14]. The number of bones in the skeletal model could be different depending on the demands, e.g. Mueller et al. in [34] uses 24 joints while in documentation of AMC viewer [2] there is 29 joints used. Nevertheless it can be seen that there is a huge number of values needed to be stored for even 30 seconds length animation with 100Hz

key-frames frequency. For the model with 24 it is required $216k$ and for the one with 29 - $261k$ of values to be stored.
Against the above, in production solutions there are the same animation used for a number of characters. Therefore there appears the repeatability of the motion of different characters on the scene, which does not correspond to the reality. Exemplary, Mills in [31] shows experimentally that the R^2 measure of repeatability between two passes of walking is equal barely $R^2 = 0.531$ for knee abduction or $R^2 = 0.985$ for right knee flexion/extension and the mean for all human joint is $R^2 = 0.7797 \pm 0.153$. The confirmation of this fact can be found e.g. by White et al. in [44] or by Steinwender et al. in [39] in case of children. For the joints in the machines (e.g. industrial robot arm) the repeatability is on the higher level, but it is quite noticeable - exemplary Sepahi-Boroujeni in [36] shows that R^2 lays in the range of $[0.82, 0.84]$. Therefore introducing the repeatability to the animated characters and machines could improve the immersion of the user (player), especially in the context of Augmented and Virtual Reality industrial systems.
Except of that, the computational complexity of the algorithms is important matter, especially in the real-time application. It is unlikely to expect that a frame in a real-time solution with varying FPS, will appear in exactly the same moments as the key-frames, therefore in the vast majority of frames, the actual pose has to be computed with methods that can be theoretically sophisticated and computationally complex (e.g. Haarbach et al. [15] or mentioned Ali-Khan et al. [4]).

1.2 Motivation

The proposed notion ELSA (Euler-Lagrange Skeletal Animation) is a novel and original approach to the process of generate the poses of skeleton in consecutive frames and it is an opportunity to improve topics with data volume, computational complexity and animation repeatability. The algorithm uses a concept of Euler-Lagrange (or Lagrange second kind) PDE and its numerical solving using the notion of *phase space* (see e.g. Hand and Finch [16]) for modelling the motion of the joints in skeletal model. For the machines (like industrial robots) there are 8 types of joints (see Blake in [7], Hoffman et al. in [17]). For the skeletal human's models the most general observation is all joints behave like *ball joints* with 6 DoF *(Degrees of Freedom)* within some limitation - e.g. knee joint's mobility is on the Sagittal plane in general, but still there exists some small deviations in the remaining planes, even in case of healthy person. If one dispose the phase spaces for each of the joint type, the animation could be stored as the start and end point in the phase space and the remaining poses lays on integral curve connecting this two points, so called *phase space trajectory* and can be easily computed using the phase space mechanism, which will be described in the further part of this article.
ELSA also can introduce the limited distortions to the modelled motion. Namely, in a selected few steps it is enough to slightly disturb the acceleration component of vector pinned to the given position - velocity point in a phase space. It

changes the rotation of the vector minimally, however the trend of the motion could be maintained. By the virtue of above each transition will be different - even though it is made of the same start and end poses. The distortion operator will behave differently in different kind of phase spaces recognizable in the light of the *Phase portrait* analysis (see Jordan and Smith [20]).

2 Material and methods

The skeletal model used in computer graphics and computer vision is a hierarchic tree structure, where the nodes are called *joints* and represents the connection between **rigid** *bones*, which could be identified as the edges of this tree. The main structure of the skeleton is defined by the starting pose (or T-pose, A-pose). In that pose the position of each joint can be defined as the *offset* from the parent joint, which is the vector in the coordinate system of parent joint. The *pose* is defined as the rotations' set of bones in the coordinate system of the joint that is the bone's begin. Therefore this rotation designates the position of the joint being the end of that bone, hence its position is generated by the rotation in the parent joint's coordinate system. One obtain the position of all joints in the skeleton by carrying such a procedure from the root to the leafs of the tree. The animation arises by computing the poses for each frame of the animation. The motion of the rigid body (in the case: bone), represented abstractly by a single point (in the case: joint), in the perspective of Lagrangian mechanics (see Hand and Finch, ibidem) can be described by the function:

$$q : \mathbb{R} \longrightarrow \overline{Q} = Q \times \dot{Q}, \ \wedge \ q(t \in \mathbb{R}) = \quad \left[q_1(t), \dots, q_n(t), \dot{q}_1(t), \dots, \dot{q}_n(t)\right]^T, \quad (1)$$

where Q is called the *configuration space* of the joint. The point in the configuration space denotes the state of the joint defined by the quantities which are appropriate for the given type of joint (e.g. position in space or rotation). The point in the subspace $\dot{Q}$ denotes the change of the state in time(e.g. velocity, angular velocity). Thus, the space $\overline{Q} = Q \times \dot{Q}$ represents all possible joint's states and all possible changes for each position, which can be written as a set:

$$\overline{Q} = Q \times \dot{Q} = \left(X, \dot{X}\right) = \left(x_1, \dots, x_n, \dot{x_1}, \dots \dot{x_n}\right) \quad (2)$$

The q function, called *trajectory*, therefore represents a single joint motion, since it can be used as the representation of that joint's animation.
The Lagrange equation of that system has the general form as follows:

$$\mathcal{L} : Q \times \dot{Q} \times \mathbb{R} \longrightarrow \mathbb{R}, \ \wedge \ \mathcal{L}\big(q_1(t), \dots, q_n(t), \dot{q}_1(t), \dots, \dot{q}_n(t), t\big) = E_k - E_p, \quad (3)$$

where E_k is kinetic energy and E_p potential energy. In this work, it was assumed that the joints has unit mass, nevertheless it is possible to introduce the other masses of the arms tied to joints. Those can be computed basing on the material and volume fetched from the parameters of the mesh saved in the CAD file. The kinetic energy of the joint obviously depends on the position and velocity of the bone, while the potential energy depends on the applied force. In this paper, no gravity was assumed. Two types of force are taken into consideration: internal, resulting from the force moving the bone, and external, applied by the next element in the kinematic chain, or by the user interacting with the device. The Euler - Lagrange equation takes the form of a system of partial differential equations:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_1}\right) - \frac{\partial \mathcal{L}}{\partial q_1} = 0 \wedge \cdots \wedge \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_n}\right) - \frac{\partial \mathcal{L}}{\partial q_n} = 0 \tag{4}$$

Consider the first term of each of these equations: $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right)$. $\dot{q}_i := \dot{q}_i(t) = \dot{x}_i$ is $(n+i)th$ argument of the Lagrangian. It computes the time derivative of this argument. This produces the second derivative $\ddot{x}_i$. In specific cases, this derivative can be equal to 0, but in general it can be assumed that, as a rule, each of the above equations can be transformed in such a way that there will be a second derivative on the left side of equation and the function φ_i on the right side, depending only on the arguments of the state space:

$$\ddot{x}_1 = \varphi_1\big(x_1, \ldots, x_n, \dot{x}_1, \ldots, \dot{x}_n\big) \wedge \cdots \wedge \ddot{x}_n = \quad \varphi_n\big(x_1, \ldots, x_n, \dot{x}_1, \ldots, \dot{x}_n\big) \tag{5}$$

It can be written in a vector form:

$$\ddot{X} = \Phi(X, \dot{X}), \;\; X \in Q, \;\; \wedge \;\; \Phi(X, \dot{X}) = \big[\varphi_1(X, \dot{X}), \ldots \varphi_n(X, \dot{X})\big]^T \tag{6}$$

Thus a vector field can be created:

$$\mathbb{V} : Q \times \dot{Q} \longrightarrow \dot{Q} \times \ddot{Q} : \;\; \mathbb{V}\big(X, \dot{X}\big) = \big[\dot{X}, \ddot{X}\big] = \big[\dot{X}, \Phi(X, \dot{X})\big]. \tag{7}$$

Such a vector field is called *phase space* of the partial differential equation (PDE) from which it was defined, so it will be called *EL animation phase space*. Such fields have the property that the *Integral curve* of this field, starting at point $\overline{X} = [X, \dot{X}]$, determines the trajectory of the motion of an object whose initial state is equal to $\overline{X}$. While it denotes both the position in space and the current velocity, having such a vector field, it is easy to determine the further movement, i.e. the position in the next frame, according to the formula:

$$\overline{X} = x_1, \ldots, x_n, \dot{x}_1, \ldots \dot{x}_n, \;\; \overline{Y} = y_1, \ldots, y_n, \dot{y}_1, \ldots \dot{y}_n, \;\; \overline{Y} = \overline{X} + t\mathbb{V}(\overline{X}). \tag{8}$$

Thus, having the vector field $\mathbb{V}$ determined, computing the animation is very efficient, because it bases on multiplying the vector by the scalar and adding the vectors. Moreover, to remember the entire animation, regardless of its complexity, it is enough to remember the starting point and possibly the end point in the vector field. Summarizing this section, one can define:

Definition 1 ***Euler-Lagrange animation****, or in short* ***EL animation*** *of a rigid body, it is an integral curve of a vector field* $\mathbb{V}\big(X, \dot{X}\big) = \big[\dot{X}, \Phi(X, \dot{X})\big]$ *starting at the* ***animation start*** *point.* ***The end of the animation*** *will be named as the point on the integral curve after which the animation will no longer take place.*

Starting from any point in the phase space, it is possible to determine the point at which the animated object will be located after the specified time, by formula 8. Applying it iterative, one can obtain a set of points representing the state of the animated object in successive moments of time. In the case of interval between frames is small enough ($\Delta t \to 0$), an integral curve in the phase space is created, which is called the *phase curve*. The projection of this curve onto the configuration space is the object's motion trajectory.

3 Results and discussion

In this section we present the theoretical results in the form of the formal definition the Euler Lagrange Skeletal Animation (ELSA) . Furthermore we present the implementation of the ELSA made in the Unity 3D environment. Finally we present some visual results on an example of two 3D models: parking gate and the industrial drill.

3.1 Euler-Lagrange Skeletal Animation concept

The notion of skeletal model could be defined using the formal notion of the tree, proposed by Kuboyama in its doctoral dissertation [24]. On the other hand, the skeletal model can be mapped from a set of its joints to the tangent bundle of the phase spaces. This additional structure provides the skeletal model with a tool for animation construction.

Definition 2 ***Euler-Lagrange Skeleton Model (ELSM)*** *R is the system $R = (P, S, \varphi)$, where:*

1. *P is a set of joints: $P = \{p_1, ..., p_m, \forall i\ p_i \in \mathbb{R}^3\}$. The number m is a* ***size*** *of tree and is denoted by a symbol $|R|$.*
2. *S is a tree:*

$$S = (P, r), where: \quad r : P \to P, \ \ r^0(p_k) = p_k, \ \ \forall k \geq 0\ \exists \delta : \ r^\delta(p_k) = p_1 \quad (9)$$

The p_k joint is a ***child joint*** *of the joint p_i if and only if $r(p_k) = p_i$, which can be written equivalently: $p_i \to p_k$. The p_k joint is a* ***parent*** *of the joint p_i. The p_1 joint is the root. The natural number δ is called the joint depth.*

3. *φ is a mapping from a set of joints into a tangent bundle of the phase space:*

$$\varphi : P \to T[Q \times \dot{Q}] : \quad \varphi(p \in P) = \mathbb{V}_p(X_p, \dot{X}_p) = \left[\dot{X}_p, \ddot{X}_p\right]. \qquad (10)$$

Given the point before a certain time interval Δt elapses from now (t), according to the equation 8 one can compute the point in the space $\varphi(p_k)$ in time t

$$\overline{X_p}(t) = \overline{X_p}(t - \Delta t) + \Delta t \varphi(p) \qquad (11)$$

In some cases, e.g. in case of prismatic joint (see [7]), the X_p represents the position of joint directly. In other cases (e.g. ball joint like in case of human joints), the X_p represents the (Cardan) angles. In those cases the position of the joint $p \in P$ in time t can be computed as follows:

$$p(t) = p_{r(p)}(t) + \Pi(X_{r(p)}(t), p), \tag{12}$$

where $\Pi(X, p)$ determines the the vector which direction is given by rotation around the point $\mathbf{0}$ by an (Cardan) angle X and has the same magnitude as p. Indeed, the formula above constitutes the *kinematic chain* for the joint p which, in each step of the recurrence, computes the rotation of the given point by the given (Cardan) angles around the parent's joint's coordinate system. It can be expressed without recurrence as follows:

$$p(t) = r^{\delta}(p) + \sum_{i=1}^{\delta} \Pi\big(X_{r^i(p)}, r^{i-1}(p)\big). \tag{13}$$

Moreover, the set $P(t) = \{p_i(t) : \forall i \; p_i \in P\}$ determines the *pose* of the model in time t and $P(0) = P$ is the initial pose (A-Pose, T-Pose) that can be obtained from the BVH file. Hereby the *pose-driven initial configuration* can be constructed:

$$I_{PD} = \{\overline{X}_{p_1}(0), \ldots \overline{X}_{p_m}(0) : \forall k \; \overline{X}_{p_k} = [\mathbf{0}, \dot{X}_{p_k}(0)]\},$$

where one consider that all joints are not rotated in the initial pose and the initial distribution of joints in the 3D space is given by the initial Pose. Alternatively, one can define the initial position of the joint directly by the *explicit initial configuration*:

$$I_{EX} = \{\overline{X}_{p_1}(0), \ldots, \overline{X}_{p_m}(0)\},$$

from which the initial pose is obtained.

Definition 3 ***Euler-Lagrange Skeletal Animation (ELSA)****, embedded on given ELSM model* (P, S, φ)*, started from the initial configuration* $IC = \{\overline{X}_{p_i} : \forall i \; p_i \in P\}$ *is the set of integral curves* $\Gamma = \{\gamma_i(t) : \forall i \; \gamma_i(0) = \overline{X}_{p_i}(0) \;\wedge\; t \in [0, L]\}$*, where* L *is the* ***length*** *of the animation. The length could be designated by the time the animation lasts or by the* ***ending pose*** $P(L)$*, which if approached, ends the animation.*

Having the pose in given time $P(t)$ one can construct the pose after a sufficient short time period elapses: $P(t + \Delta t)$ applying the equation 11 to each joint separately. In case joint's phase space describes rotation and angular velocities the equations 12 or 13 should be used, while in opposite case the position of joint $p_k = X_k$ directly. To make each animation's performance unique the *distortion operator* acting on the function φ is introduced, defined as follows:

$$\begin{aligned}\mathbf{D}_\mu \varphi = [\dot{X}, \mu(\ddot{X})], s.t. : \forall \overline{X} \in U \subseteq Q : \lim_{t \to \infty} \gamma_{\mathbf{D}_\mu \varphi}\big(\overline{X}, t\big) = \gamma_\varphi\big(\overline{X}, t\big), \vee \\ \forall \overline{X} \in U \subseteq Q : \exists t \leq L \; \forall T > t \;\; \gamma_{\mathbf{D}_\mu \varphi}\big(\overline{X}, T\big) = \gamma_\varphi\big(\overline{X}, T\big)\end{aligned} \tag{14}$$

where $\gamma_\varphi(\overline{X}, t)$ is the integral curve on the field defined by φ starting on the point $\overline{X}$. The above definition characterize the distortion model applied to the skeletal model. Firstly operator acts on the acceleration (second derivative) only to ensure the consistency with the phase space definition, where the first part of the vector in the field has to be equal to the first derivative coordinate of the point it is pinned to. Secondly, the conditions ensure that the animation cannot "jump" to another, not convergent, phase curve for ever. In fact the first condition covers the area that are under the influence of approximately stable point or curve in the phase portrait, while the second one - the cases of stable (but not approximately) curves, unstable points and curves. The distortion operator does not have to act in every frame of the animation. Its acting throw the animation to other trajectory (e.g. randomly). In case it comes back to the first (2^0) or will be, after some time, sufficiently close to the original one (1^0), the operator could be applied again.

3.2 ELSA Implementation

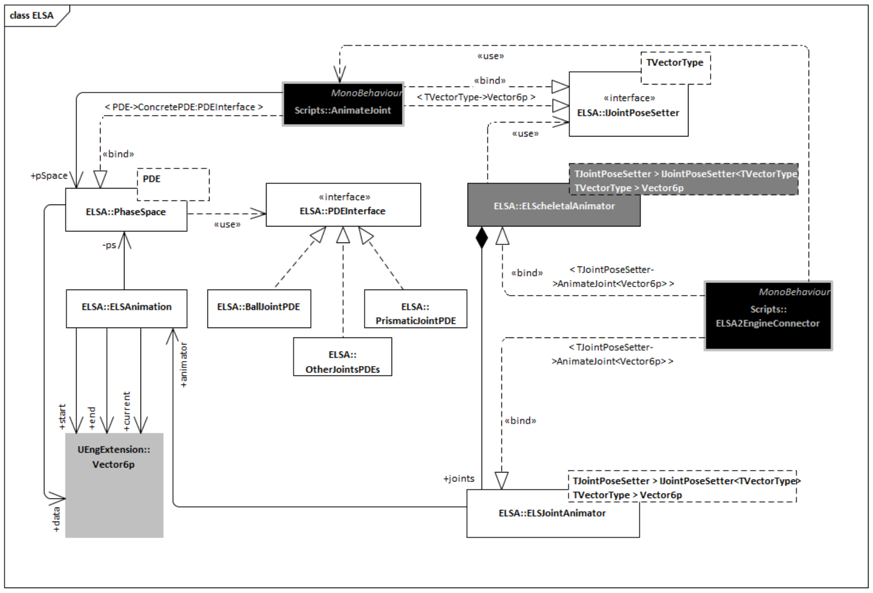

Fig. 1. UML diagram of the ELSA implementation.

ELSA was implemented in C# language as the set of scripts for the 3D graphics engine Unity3D using object oriented and generic programming method (fig. 1). The instance of template class *ELSA:PhaseSpace* within the parameter *PDE* constrained to a class implementing the interface *PDEInterface* returns the vector $[\dot{X}, \ddot{X}]^T$ for the given argument $[X, \dot{X}]^T$ according to the concrete realization

of that interface. Hence one can say, that the template declaration of this class creates the abstract structure like tangent bundle from the definition 2 (pt. 3). Moreover, the declaration of the variable with the concrete PDE as the template parameter plays the role of the function φ from this definition. Such an implementation is not only consistent with the presented definition, but it realizes as well, the generic version of the strategy GoF design pattern „Strategy" which allows for flexible generation of the desired PDE for not implemented cases of joints yet. Within the presented implementation there are PDEs for each of 8 commonly known joint types delivered. On the diagram on mentioned figure, there are shown only two exemplary concrete PDEs: *BallJointPDE* and *PrismaticJointPDE*. The PhaseSpace has constructor hidden, hence it is created by static *Generator*. It is the first of two connectors between the ELSA module and the visualization part of an application.
Subsequently, a class *ELSAnimation* represents a single ELS animation defined by: (1) *PhaseSpace*, (2) Initial pose of the joint given by *Vector6p* and (3) the end given by the final pose. This class is instantiated by the static generator as well. It avails to fetch the next frame of an animation (meaning „after time $\Delta t > 0$" from now), the previous one ($\Delta t < 0$), rewind to initial and final pose and check if the animation is completed (meaning the final pose was obtained).
ELSJointAnimator is a template class which uses the interface *IJointPoseSetter* (which will be described further) for setting the pose of the joint supposed to be animated and existing on the scene being rendered. It uses the object of class *ELSAnimation* for computing the pose in current time and delivers the methods with real rotation and position change of object existing on the scene and for pose update exclusively. In also introduces the option of determination of the animation end by its length or defining the animation as *endless*, which is intended for periodic phase curves. *ELSJointAnimator*'s instances are aggregated through composition by the *ELScheletalAnimator*, which provides the same methods but applied to the whole skeletal model, defined outside the ELSA module.
The interface *IJointPoseSetter* is the second connection between ELSA functionalities and the visualization part of the application, made with Unity Engine. It defines one method *SetPose()*, which allows to set the pose of the object, that can be seen on the 3D scene (*Unity::GameObject*) considered as joint. Marking the *GameObject* as joints is made by connection the script realizing *IJointPoseSetter* as the *Unity component*. On the picture 2 there are selected frames fetched from an animation of bottom part of humanoid skeletal model. The animation is applied to two joints.
One can make the observation that the mapping φ seems to be connected with some kind of fibration, which demand further theoretical research. If these assumptions prove to be confirmed, it would mean that each pose determines one fiber connected with φ. Hence the whole ELSA, as in the limit is the continuous, , would designate the smooth curve on the fibration and finally on the manifold that would be the base of the fibration. The similar animation of the same class, will designates similar curves on that manifold probably. The set of those curves could be utilised in many ways, e.g. as the feature vector for the classification.

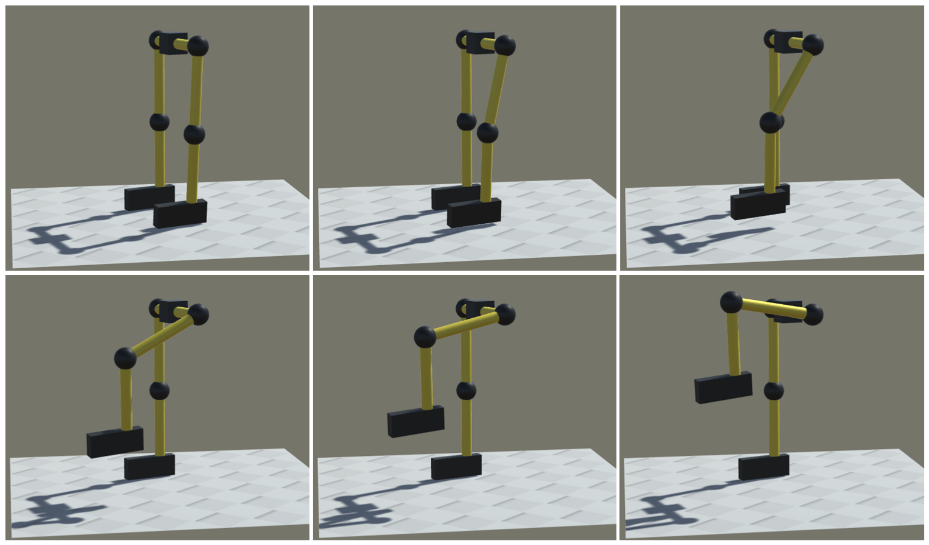

Fig. 2. Exemplary frames from animation of one leg in the bottom part of humanoid skeleton.

4 Conclusions and Acknowledgments

In the presented paper, a novel and original approach to generate skeletal animations was proposed. The notion of ELSA, prepared in game engine environment, was applied in the process of generation of skeletal poses in consecutive frames. In contrast to previous works, the presented method bases on the Euler Lagrange equations of motion, configuration and phase space notion, which for our best knowledge was applied for the first time. It enables to define the whole skeletal animation by introducing initial and final poses without any additional operations. Those features highly improve the computational complexity and reduce data volume required to store in a memory. The additional benefit of ELSA approach lies in repeatability of any animation and its unique recreation. The presented work describes future research opportunities, practical applications and is available along with source code. The presented concept of animation generation was originally used in the Platform 4.0. That project was mainly designed to create an interactive training process by application of augmented, mixed and virtual reality. In that problem the input of a system usually consists of just raw 3D model saved in any CAD format. The task of a designer is to correctly prepare model and attach correct animations which is more complicated process. Having properly joint identified, user is able to create any kind of animation.

Presented work is a part of the project "Devising a modern platform for creating interactive training and servicing instructions for machinery and technical devices using AR, VR and MR technologies". Project is co-financed by the European Union through the European Regional Development Fund under the Regional Operational Programme for Slaskie Voivodeship 2014-2020.

References

1. C3D.ORG - The biomechanics standard file format, https://www.c3d.org/
2. CS 15-464/16-464 Technical Animation, http://graphics.cs.cmu.edu/nsp/course/15-464/Fall05/assignments/StartupCodeDescription.html
3. Abu Rumman, N., Fratarcangeli, M.: Position-based skinning for soft articulated characters. Comput. Graph. Forum **34**(6), 240–250 (Sep 2015). https://doi.org/10.1111/cgf.12533
4. Ali Khan, M., Sarfraz, M.: Motion Tweening for Skeletal Animation by Cardinal Spline. In: Abd Manaf, A., Sahibuddin, S., Ahmad, R., Mohd Daud, S., El-Qawasmeh, E. (eds.) Informatics Engineering and Information Science. pp. 179–188. Communications in Computer and Information Science, Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25483-3-14
5. Asraf, S.M.H., Abdullasim, N., Romli, R.: Hybrid animation: Implementation of motion capture. IOP Conference Series: Materials Science and Engineering **767**, 012065 (mar 2020). https://doi.org/10.1088/1757-899x/767/1/012065
6. Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. ACM Trans. Graph. **26**(3), 72–es (Jul 2007). https://doi.org/10.1145/1276377.1276467
7. Blake, A.: Design of mechanical joints. No. 42 in Mechanical engineering, M. Dekker, New York (1985)
8. Burtnyk, N., Wein, M.: Computer-Generated Key-Frame Animation. Journal of the SMPTE **80**(3), 149–153 (Mar 1971). https://doi.org/10.5594/J07698
9. Castro, G.G., Athanasopoulos, M., Ugail, H.: Cyclic animation using partial differential equations. Vis Comput **26**(5), 325–338 (May 2010). https://doi.org/10.1007/s00371-010-0422-5, company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 5 Publisher: Springer-Verlag
10. Catmull, E.: The Problems of Computer-Assisted Animation **12**(3), 348–353 (Aug 1978). https://doi.org/http://doi.acm.org/10.1145/800248.807414
11. Courant, R., Hilbert, D.: Methods of Mathematical Physics, 2. Wiley-VCH, Weinheim (1989), oCLC: 609855380
12. Dai, H., Cai, B., Song, J., Zhang, D.: Skeletal Animation Based on BVH Motion Data. In: 2010 2nd International Conference on Information Engineering and Computer Science. pp. 1–4. IEEE, Wuhan, China (Dec 2010). https://doi.org/10.1109/ICIECS.2010.5678292
13. Evans, L.C.: Partial differential equations. No. v. 19 in Graduate studies in mathematics, American Mathematical Society, Providence, R.I (1998)
14. Geroch, M.S.: Motion capture for the rest of us. Journal of Comp. Sciences in Colleges p. 2004
15. Haarbach, A., Birdal, T., Ilic, S.: Survey of Higher Order Rigid Body Motion Interpolation Methods for Keyframe Animation and Continuous-Time Trajectory Estimation. In: 2018 International Conference on 3D Vision (3DV). pp. 381–389. IEEE, Verona (Sep 2018). https://doi.org/10.1109/3DV.2018.00051
16. Hand, L.N., Finch, J.D.: Analytical mechanics. Cambridge University Press, Cambridge ; New York (1998)

17. Hoffman, G., Ju, W.: Designing Robots With Movement in Mind. Journal of Human-Robot Interaction **3**(1), 89 (Mar 2014). https://doi.org/10.5898/JHRI.3.1.Hoffman
18. Jain, A., Rodriguez, G.: Diagonalized dynamics of robot manipulators. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation. pp. 334–339. IEEE Comput. Soc. Press, San Diego, CA, USA (1994). https://doi.org/10.1109/ROBOT.1994.351273
19. Jain, A., Rodriguez, G.: Diagonalized Lagrangian robot dynamics. IEEE Transactions on Robotics and Automation **11**(4), 571–584 (Aug 1995). https://doi.org/10.1109/70.406941
20. Jordan, D.W., Smith, P.: Nonlinear ordinary differential equations: an introduction for scientists and engineers. New York : Oxford University Press, Oxford [England], 4th ed edn. (2007)
21. Khan, M.A., Sarfraz, M.: Motion Tweening for Skeletal Animation by Cardinal Spline. In: Informatics Engineering and Information Science. pp. 179–188. Springer, Berlin, Heidelberg (Nov 2011). https://doi.org/10.1007/978-3-642-25483-3_14
22. Kozlowski, K.: Standard and diagonalized Lagrangian dynamics: a comparison. In: Proceedings of 1995 IEEE International Conference on Robotics and Automation. vol. 3, pp. 2823–2828. IEEE, Nagoya, Japan (1995). https://doi.org/10.1109/ROBOT.1995.525683
23. Krayevoy, V., Sheffer, A.: Boneless motion reconstruction. In: ACM SIGGRAPH 2005 Sketches. p. 122–es. SIGGRAPH '05, Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1187112.1187259
24. Kuboyama, T.: Matching and learning in trees. In: Doctoral dissertation (2007)
25. Kuo, C., Liang, Z., Fan, Y., Blouin, J.S., Pai, D.K.: Creating impactful characters: Correcting human impact accelerations using high rate imus in dynamic activities. ACM Trans. Graph. **38**(4) (Jul 2019). https://doi.org/10.1145/3306346.3322978
26. Le Naour, T., Courty, N., Gibet, S.: Skeletal mesh animation driven by few positional constraints. Computer Animation and Virtual Worlds **30**(3-4) (May 2019). https://doi.org/10.1002/cav.1900
27. Levi, Z., Gotsman, C.: Smooth rotation enhanced as-rigid-as-possible mesh animation. IEEE Transactions on Visualization and Computer Graphics **21**(2), 264–277 (2015). https://doi.org/10.1109/TVCG.2014.2359463
28. Li, J., Lu, G., Ye, J.: Automatic skinning and animation of skeletal models. The Visual Computer **27**(6), 585–594 (Jun 2011). https://doi.org/10.1007/s00371-011-0585-8, company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 6 Publisher: Springer-Verlag
29. Lobão, A., Evangelista, B., Leal de Farias, J., Grootjans, R.: Skeletal Animation. Beginning XNA 3.0 Game Programming pp. 299–336 (2009). https://doi.org/10.1007/978-1-4302-1818-0-12, publisher: Apress
30. Mandery, C., Terlemez, O., Do, M., Vahrenkamp, N., Asfour, T.: The KIT whole-body human motion database. In: 2015 International Conference on Advanced Robotics (ICAR). pp. 329–336. IEEE, Istanbul, Turkey (Jul 2015). https://doi.org/10.1109/ICAR.2015.7251476
31. Mills, P.M., Morrison, S., Lloyd, D.G., Barrett, R.S.: Repeatability of 3D gait kinematics obtained from an electromagnetic tracking system during treadmill locomotion. Journal of Biomechanics **40**(7), 1504–1511 (Jan 2007). https://doi.org/10.1016/j.jbiomech.2006.06.017
32. Min, J., Chen, Y.L., Chai, J.: Interactive generation of human animation with deformable motion models. ACM Transactions on Graphics **29**(1), 1–12 (Dec 2009). https://doi.org/10.1145/1640443.1640452

33. Mukundan, R.: Skeletal Animation. In: Advanced Methods in Computer Graphics: With examples in OpenGL, pp. 53–76. Springer London, London (2012). https://doi.org/10.1007/978-1-4471-2340-8-4
34. Muller, M., Roder, T., Clausen, M., Eberhardt, B., Kruger, B., Weber, A.: Documentation Mocap Database HDM05 p. 34
35. Parent, R.: Computer animation: algorithms and techniques. The Morgan Kaufmann series in computer graphics and geometric modeling, Morgan Kaufmann Publishers, San Francisco (2002)
36. Sepahi-Boroujeni, S., Mayer, J., Khameneifar, F.: Repeatability of on-machine probing by a five-axis machine tool. International Journal of Machine Tools and Manufacture **152**, 103544 (May 2020). https://doi.org/10.1016/j.ijmachtools.2020.103544
37. Sito, T., Whitaker, H.: Timing for animation. Focal, Oxford, 2nd ed edn. (2009), oCLC: 705760367
38. Spanlang, B., Navarro, X., Normand, J.M., Kishore, S., Pizarro, R., Slater, M.: Real time whole body motion mapping for avatars and robots. In: Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology - VRST '13. p. 175. ACM Press, Singapore (2013). https://doi.org/10.1145/2503713.2503747
39. Steinwender, G., Saraph, V., Scheiber, S., Zwick, E.B., Uitz, C., Hackl, K.: Intrasubject repeatability of gait analysis data in normal and spastic children. Clinical Biomechanics **15**(2), 134–139 (Feb 2000). https://doi.org/10.1016/S0268-0033(99)00057-1
40. Stoll, C., Aguiar, E.D., Theobalt, C., Seidel, H.: A volumetric approach to interactive shape editing. In: Saarbrücken: Max-Planck-Institut für Informatik (2007)
41. Sujar, A., Casafranca, J.J., Serrurier, A., Garcia, M.: Real-time animation of human characters' anatomy. Computers & Graphics **74**, 268–277 (Aug 2018). https://doi.org/10.1016/j.cag.2018.05.025
42. Tupling, S.J., Pierrynowski, M.R.: Use of cardan angles to locate rigid bodies in three-dimensional space. Medical and Biological Engineering and Computing **25**(5), 527–532 (Sep 1987). https://doi.org/10.1007/BF02441745, company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 5 Publisher: Kluwer Academic Publishers
43. Wang, J., Lyu, K., Xue, J., Gao, P., Yan, Y.: A markerless body motion capture system for character animation based on multi-view cameras. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8558–8562 (2019). https://doi.org/10.1109/ICASSP.2019.8683506
44. White, R., Agouris, I., Selbie, R., Kirkpatrick, M.: The variability of force platform data in normal and cerebral palsy gait. Clinical Biomechanics **14**(3), 185–192 (Mar 1999). https://doi.org/10.1016/S0268-0033(99)80003-5
45. Zhao, Y., Liu, J.: Volumetric subspace mesh deformation with structure preservation. Comput. Animat. Virtual Worlds **23**(5), 519–532 (Sep 2012). https://doi.org/10.1002/cav.1488
46. Zhou, F., Luo, X., Huang, H.: Application of 4-Point Subdivision to Generate In-Between Frames in Skeletal Animation. In: Technologies for E-Learning and Digital Entertainment, vol. 3942, pp. 1080–1084. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/11736639-134, series Title: Lecture Notes in Computer Science
47. Zollhöfer, M., Sert, E., Greiner, G., Süßmuth, J.: Gpu based arap deformation using volumetric lattices. In: Eurographics (2012)