# Direct Delta Mush Skinning Compression with Continuous Examples

BINH HUY LE, SEED - Electronic Arts, USA
KEVEN VILLENEUVE, SEED - Electronic Arts, Canada
CARLOS GONZALEZ-OCHOA, Independent Researcher, USA

Direct Delta Mush (DDM) is a high-quality, direct skinning method with a low setup cost. However, its storage and run-time computing cost are relatively high for two reasons: its skinning weights are $4 \times 4$ matrices instead of scalars like other direct skinning methods, and its computation requires one $3 \times 3$ Singular Value Decomposition per vertex.

In this paper, we introduce a compression method that takes a DDM model and splits it into two layers: the first layer is a smaller DDM model that computes a set of virtual bone transformations and the second layer is a Linear Blend Skinning model that computes per-vertex transformations from the output of the first layer. The two-layer model can approximate the deformation of the original DDM model with significantly lower costs.

Our main contribution is a novel problem formulation for the DDM compression based on a continuous example-based technique, in which we minimize the compression error on an uncountable set of example poses. This formulation provides an elegant metric for the compression error and simplifies the problem to the common linear matrix factorization. Our formulation also takes into account the skeleton hierarchy of the model, the bind pose, and the range of motions. In addition, we propose a new update rule to optimize DDM weights of the first layer and a modification to resolve the floating-point cancellation issue of DDM.

CCS Concepts: • **Computing methodologies** → **Animation**.

Additional Key Words and Phrases: deformation, skeletal animation, delta mush, linear blend skinning, data driven, example poses, sparse coding, least squares

**ACM Reference Format:**
Binh Huy Le, Keven Villeneuve, and Carlos Gonzalez-Ochoa. 2021. Direct Delta Mush Skinning Compression with Continuous Examples. *ACM Trans. Graph.* 40, 4, Article 1 (August 2021), 13 pages. https://doi.org/10.1145/3450626.3459779

Fig. 1. We compress DDM model by splitting it into two layers: the first layer is a smaller DDM model that takes the master bones (yellow) and computes a small set of virtual bone transformations (red), and the second layer is a large but cheap LBS model that computes per-vertex skinning (blue) from virtual bones.

## 1 INTRODUCTION

For character animation, skinning, especially *Linear Blend Skinning* (LBS) [Magnenat-Thalmann et al. 1988], is a de facto standard in real-time animation systems. Skinning refers to a family of methods to deform character models (the skin) by controlling a set of handles, typically a set of bones, which resemble the anatomical skeleton of the characters. Skinning is a vital piece of many animation systems because characters, especially deformable characters such as humans and animals, are normally the main focus of the story. Skinning - and LBS in particular - is widely used due to its simplicity, performance and intuitiveness.

The main limitation of LBS is deformation quality, which includes issues such as elbow collapsing and candy wrapper artifacts [Jacobson et al. 2014]. Other skinning models have been proposed to address these problems while keeping the advantages of performance and intuitiveness. Some examples are: Log-matrix Skinning (LMS) [Alexa 2002; Magnenat-Thalmann et al. 2004], Spherical Blend Skinning (SBS) [Kavan and Žára 2005], Dual Quaternion Skinning (DQS) [Kavan et al. 2008], Skinning with Optimized Centers of Rotation (CoR) [Le and Hodgins 2016]. Most of these new models belong to the direct skinning category, in which the deformation on each vertex of the character can be explicitly expressed as a non-linear function of the input bone transformations. Unfortunately, this family of new methods introduces bulging artifacts [Jacobson et al. 2014].

Notably, *Direct Delta Mush skinning* (DDM) [Le and Lewis 2019] was introduced as a high-quality direct skinning method with cheap setup costs. DDM is based on Delta Mush deformer (DM) [Mancewicz et al. 2014], a popular skinning method in movie production. DDM can produce the same deformations as the original DM, in which collapsing and bulging issues are diminished. In particular, DDM

Authors' addresses: Binh Huy Le, SEED - Electronic Arts, Redwood City, CA, USA, bbinh85@gmail.com; Keven Villeneuve, SEED - Electronic Arts, Montreal, Québec, Canada, keven.villeneuve@mail.mcgill.ca; Carlos Gonzalez-Ochoa, Independent Researcher, Los Angeles, CA, USA, cgonzoo@gmail.com.


0730-0301/2021/8-ART1 $15.00
https://doi.org/10.1145/3450626.3459779

was the first direct method to offer the skin sliding effect, thanks to the decomposition of rotation and translation in its formulation.

However, the storage and run-time computing costs of DDM are quite high compared to the traditional methods such as LBS or DQS, especially with DDM variant 0, the variant that offers the best skinning quality. The reasons are: (1) DDM stores pre-computed $4 \times 4$ multi-weights compared to scalar weights of LBS and DQS; and (2) DDM computes a $3 \times 3$ Singular Value Decomposition (SVD) per-vertex at run-time. Theoretically, the multi-weight setup suggests that DDM requires 16 times more storage than LBS and DQS (or 10 times when taking advantage of the symmetry of multi-weight matrices).

In this paper, we explore the idea of compressing DDM with a two-layer model (illustrated in Fig. 2):

- The first layer is a smaller DDM model that takes the original bone transformation and computes a set of virtual bone transformations.
- The second layer is a large but cheap LBS model that takes the virtual bone transformations from the first layer and computes per-vertex skinning.

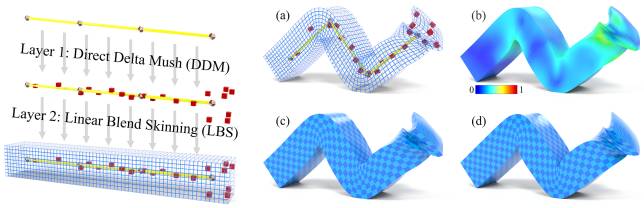


Fig. 2. Left: an illustration of our two-layer skinning model: layer 1 (DDM) computes virtual bone transformations (red) from master bone transformations (yellow) and layer 2 (LBS) computes per-vertex deformation (blue). Right: (a) our deformation with master bones and virtual bones overlay, (b) the approximation error, (c) our deformation, and (d) the ground truth DDM deformation.

The compressed two-layer model can significantly reduce the memory footprint and the run-time computation cost by sharing the expensive computation of non-linear deformation to a small set of virtual bones, and then, linearly interpolating the deformation to the high-resolution mesh. This idea is similar to the previous work by Kavan et al. [2009] and the later work by Le and Deng [2013].

The parameters of our compression model are computed by a *continuous example-based technique*, in which we formulate the problem as the compression error minimization on an uncountable set of example poses (§3). Compared to the traditional example-based techniques, which use a countable set of discrete poses, our method offers several advantages:

- The continuous formulation provides an elegant metric for the compression error over the set of example poses. Our analytical solution creates a non-linear mapping of the DDM multi-weights (Eq. (11b) and Eq. (11c)) so that the L2 norm on the mapped multi-weights resembles the compression error (Eq. (11a)).
- Our method uses unlimited amounts of data. The data sampling is controlled by high-level parameters such as the *ranges of motion* for example poses (§3.6). This approach helps to reduce the data dependency and produce more robust results.
- The time complexity of our compression does not depend on the number of example poses, therefore the run-time of our method is significantly faster compared to the traditional discrete example-based techniques.
- Our continuous formulation is linear, which allows using well-studied optimization techniques. Specifically, our optimization employs the dictionary learning and sparse coding framework (§4).

In addition to the main contribution, which is the formulation of continuous example-based DDM compression, this paper also presents:

- The minimization algorithm for the formulated compression error, including a novel DDM multi-weights update (§4.3).
- A modification to solve the floating-point cancellation issue of the DDM (§5). This modification allows more robust computation when using lower precision floating-point numbers and offers more skinning performance on less storage.
- An extensive analysis of most of the steps in our pipeline (§7).

## 2 RELATED WORK

In the previous section, we reviewed the state-of-the-art *direct skinning methods*, such as LBS, LMS, SBS, DQS, CoR, and DDM. On top of these models, the skinning quality can be further improved by adding helper bones [Mohr and Gleicher 2003; Mukai 2015; Mukai and Kuriyama 2016], decomposing transformations [Jacobson and Sorkine 2011; Kavan and Sorkine 2012], or adding corrective shapes [Kry et al. 2002; Lewis et al. 2000; Sloan et al. 2001]. These setups are typically complex. Their quality and cost heavily depend on the talent of rigging artists, where many extra helper bones, extra weights, or corrective shapes are just used to correct skinning artifacts.

Setup burdens can be reduced by using *indirect skinning methods* [Jacobson et al. 2012; Sorkine and Alexa 2007; Sumner and Popović 2004; Sumner et al. 2005; Vaillant et al. 2013, 2014]. Typically, these methods formulate non-linear problems locally, which better resemble the deformation properties of materials. However, the main limitation of these methods is the iterative computation, which can greatly impact the performance due to the overhead of global synchronizations between iterations.

A popular indirect method is Delta Mush (DM) [Mancewicz et al. 2014]. It is a versatile mesh deformer that improves any non-smooth deformation. The main idea of DM is to first apply mesh smoothing, a.k.a. "mushing", on top of an arbitrary deformed mesh, which will smooth both deformation and surface details. Then, the surface details are added back to the smooth mesh to produce the final deformation. In this step, the surface details are the difference between the original (undeformed) mesh and its smooth version in the local coordinate frame, a.k.a. the "delta". Practically, DM can be applied on top of rigid-bind skinning to produce smooth deformation. This process does not require smooth bone-vertex skinning weights. However, the main bottleneck of DM is its iterative Laplacian smoothing of every frame of the animation, which prevents

parallel implementations in a single pass. This limitation is similar to the issue of indirect skinning methods. For its characteristics, DM is widely used in industry, especially in movie production, due to the low requirement on real-time performance.

The more compute-intensive approaches with high-quality are *simulation methods* [Hahn et al. 2012, 2013; Ichim et al. 2017; Kadleček et al. 2016; Lee et al. 2009; Li et al. 2013; Liu et al. 2013; McAdams et al. 2011b; Rémillard and Kry 2013; Saito et al. 2015; Smith et al. 2018; Teran et al. 2005]. These methods can generate many complex effects such as skin sliding, collision, muscle bulging, or jiggling. However, their performance is not suitable for real-time applications.

*Skinning from examples* formulates the problem as a general data regression that learns the map from skeleton articulation to vertex displacements [Anguelov et al. 2005; Bailey et al. 2018; Feng et al. 2008; Gao et al. 2016; Jones et al. 2016; Loper et al. 2014, 2015; Wang et al. 2007]. These models can be trained with the input from simulation methods or corrective shapes made by artists, a.k.a *example poses*. These models have high approximation power once they are trained on a large amount of data. In addition, due to the complexity of the machine learning model, implementing these methods with GPU shaders is challenging.

There is a family of *example-based skinning methods* specifically designed around linear models [Hasler et al. 2010; James and Twigg 2005; Kavan et al. 2010; Le and Deng 2012, 2014]. These methods are highly compatible with graphics and animation engines due to their model representations (LBS). In addition, some units of these methods can be used in other tasks for LBS; for example, constrained linear least squares solvers are used for solving skinning weights [James and Twigg 2005] and weights sparsification [Landreneau and Schaefer 2010], and Procrustes analysis is used for solving bone transformations [Hasler et al. 2010; Le and Deng 2012].

Our idea of reducing the cost of skinning models by a two-layer compression is similar to previous attempts [Kavan et al. 2009; Le and Deng 2013]. Kavan et al. [2009] reduce the cost of DQS by splitting it into a small DQS layer and a large LBS layer. The optimization of this model is specifically designed for DQS, which requires an explicit calculation of dual quaternion derivatives. Le and Deng [2013] compress a dense LBS by splitting it into a small, dense LBS layer on top and a large, sparse, LBS layer at the bottom. The problem is formulated and solved as a sparse compression of the skinning weight matrix. While we use this optimization framework, the major difference of our work is the continuous formulation and constrained update to keep the multi-weights in a valid domain of DDM weights.

## 3 PROBLEM FORMULATION

We solve the parameters of the compressed model by minimizing the mean squared compression error over a set of example poses. Instead of using a countable set of poses, we carefully design a continuous sampling strategy of poses so that the compression error can be analytically computed from the distribution of bone transformations (Eq. (11)). In particular, this formulation maps the original DDM multi-weights to a new space so that the compression error can be measured by a linear system (Eq. (11b)). This map is composed of four consecutive steps, where each step is a map of $\mathbb{R}^{4\times4} \rightarrow \mathbb{R}^{4\times4}$:

- The *linearization map* (§3.3) relaxes the (non-linear) orthogonal constraint of DDM;
- The *hierarchical map* (§3.4) handles the skeleton structure;
- The *coordinate changing map* (§3.5) converts non-uniform transformation sampling in world space to uniform transformation sampling in local joint space;
- The *continuous sampling map* (§3.6) analytically computes the distribution of uniform transformation sampling with controllable rotation and translation ranges.

This strategy allows for constructing the objective function without explicit summing over example poses like the traditional example-based methods while still retaining the high-level control over the skeleton hierarchical and the range of motions.

### 3.1 Data

The input for our compression method is a DDM skinning character model, including:

- the *geometry* consists of $n$ vertices, where $\mathbf{u}_i \in \mathbb{R}^4$ is the homogeneous coordinate of vertex $i$ at the rest pose;
- the *DDM skinning model* with $m$ master bones and pre-computed multi-weights $\{\Omega_{ji} \in \mathbb{R}^{4\times4} \mid j = 1..m, i = 1..n\}$, where $\Omega_{ji}$ denotes the multi-weight of bone $j$ on vertex $i$;
- the *skeletal hierarchy*, defined by a hierarchical matrix $\mathbf{H} \in \{0,1\}^{m\times m}$, where:

$$h_{\gamma j} = \begin{cases} 1 & \text{if } j = \gamma \text{ or bone } j \text{ is a descendant of bone } \gamma, \\ 0 & \text{otherwise;} \end{cases} \tag{1}$$

- the *bind pose* $\{\mathbf{B}_j \in \mathbb{R}^{4\times4} \mid j = 1..m\}$, where $\mathbf{B}_j$ denotes the rigid transformation in the world coordinate of bone $j$ (world transformation) at the rest pose;
- the *ranges of motion* $\{\bar{r}_j \in \mathbb{R}$ and $\bar{t}_j \in \mathbb{R} \mid j = 1..m\}$, where $[-\bar{r}_j..\bar{r}_j]$ denotes the range of Euler angles rotation and $[-\bar{t}_j..\bar{t}_j]$ denotes the range of xyz translation of bone $j$ with respect to the bind pose.

With user-defined target virtual bones $p$, the output compressed model consists of two layers:

- The *first layer* is a DDM model that computes transformation of $p$ virtual bones, represented by multi-weights $\{\Delta_{jk} \in \mathbb{R}^{4\times4} \mid j = 1..m, k = 1..p\}$, where $\Delta_{jk}$ denotes the multi-weight of master bone $j$ on virtual bone $k$.
- The *second layer* is a LBS model that computes $n$ mesh vertex transformations, represented by scalar-weights $\{a_{ki} \in \mathbb{R} \mid k = 1..p, i = 1..n\}$, where $a_{ki}$ is the weight of virtual bone $k$ on vertex $i$.

The multi-weights of DDM skinning models are constrained to be symmetric and affine:

- $\Omega_{ji}$ and $\Delta_{jk}$ are symmetric $\forall i, j, k$;
- $\sum_{j=1}^{m}(\Omega_{ji})_{4,4} = 1, \forall i$ and $\sum_{j=1}^{m}(\Delta_{jk})_{4,4} = 1, \forall k$, where $(\cdot)_{4,4}$ denotes the element at the $4^{\text{th}}$ row and $4^{\text{th}}$ column of the matrix.

The scalar-weights of the LBS model are constrained to be sparse, non-negative, and affine:

- $\|\{a_{ki}, k = 1..p\}\|_0 \leq z, \forall i$, where $\|\cdot\|_0$ denotes the zero-norm, a.k.a. the number of non-zeros, of the set;
- $a_{ki} \geq 0, \forall i, k$;
- $\sum_{k=1}^{p} a_{ki} = 1, \forall i$.

### 3.2 Example Poses

We design the set of example poses based on the skeletal hierarchy and the bind pose.

The skeletal hierarchy, defined in Eq.(1), is a forest, which may include unorganized bones and multiple roots (i.e. bones without a parent), as illustrated in Fig. 3. We denote joint $j$ as the joint between bone $j$ and its parent. If $j$ has no parent, i.e. unorganized bone or root bone, joint $j$ and bone $j$ are the same. Note that the translation part of the bind matrix $\mathbf{B}_j$ is the position of joint $j$ at the rest pose and the rotation part of $\mathbf{B}_j$ represents the orientation of joint $j$ at the rest pose.

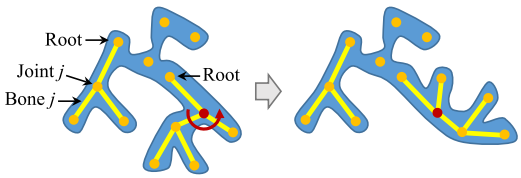


Fig. 3. Illustration of skeletal hierarchy and our example pose sampling strategy. Left: the rest pose. Right: one example pose with bone transformation sampled around the red joint and the transformation is carried down to its descendants.

Our set of example poses $\mathbb{S}$ consists of $m$ uncountable subsets $\mathbb{S} = \cup\{\mathbb{S}_\gamma \mid \gamma = 1..m\}$, where $\mathbb{S}_\gamma$ denotes the set of example poses obtained by uniformly sampling the transformation of bone $\gamma$ around its joint at the bind pose $\mathbf{B}_\gamma$. Due to the skeletal hierarchy of bones, the transformation of bone $\gamma$ will be propagated to all of its descendants (Fig. 3).

For each pose $\mathcal{M} \in \mathbb{S}$, defined by world transformations of bones $\mathcal{M} = \{\mathbf{M}_j \mid j = 1..m\}$, assuming:

- the input DDM model computes the deformation $\{\mathbf{v}_i = \Theta(\mathcal{M}, \mathcal{O}_i)\mathbf{u}_i \mid i = 1..n\}$,
- the output two-layer model computes the deformation $\{\widetilde{\mathbf{v}}_i = \sum_{k=1}^{p} a_{ki}\, \Theta(\mathcal{M}, \mathcal{D}_k)\mathbf{u}_i \mid i = 1..n\}$,

where $\mathbf{v}_i \in \mathbb{R}^4$ and $\widetilde{\mathbf{v}}_i \in \mathbb{R}^4$ are the homogeneous coordinate of vertex $i$ in the deformed pose, $\Theta(\mathcal{M}, \mathcal{O}_i)$ is the transformation of vertex $i$ and $\Theta(\mathcal{M}, \mathcal{D}_k)$ is the transformation of virtual bone $k$. For convenience, we define $\mathcal{O}_i = \{\Omega_{ji} \in \mathbb{R}^{4\times4} \mid j = 1..m\}$ as the multi-weights of vertex $i$ and $\mathcal{D}_k = \{\Delta_{jk} \in \mathbb{R}^{4\times4} \mid j = 1..m\}$ as the multi-weights of virtual bone $k$.

The mean squared compression error on set $\mathbb{S}$ is:

$$E = \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{\gamma=1}^{m}\sum_{i=1}^{n}\int_{\mathbb{S}_\gamma}\|\widetilde{\mathbf{v}}_i - \mathbf{v}_i\|_2^2$$
$$= \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{\gamma=1}^{m}\sum_{i=1}^{n}\int_{\mathbb{S}_\gamma}\Big\|\Big(\sum_{k=1}^{p} a_{ki}\,\Theta(\mathcal{M}, \mathcal{D}_k) - \Theta(\mathcal{M}, \mathcal{O}_i)\Big)\mathbf{u}_i\Big\|_2^2 \quad (2)$$

In this equation, $|\mathbb{S}_\gamma|$ denotes the size of $\mathbb{S}_\gamma$, or the (uncountable) number of example poses when sampling at joint $\gamma$. We assume that the number of examples per joint are the same, i.e. $|\mathbb{S}_\gamma| = |\mathbb{S}_j|, \forall \gamma, j$.

### 3.3 Linearization

The transformation $\Theta(\mathcal{M}, \mathcal{O}_i) = \begin{bmatrix}\mathbf{R}_i & \mathbf{t}_i\\ 0 & 1\end{bmatrix}$ is computed by Eq. 10 of the original DDM model [Le and Lewis 2019]:

$$\mathbf{R}_i = \arg\min_{\Phi}\Big\|\Phi\mathbf{Z}_i{\Lambda_i}^{1/2} - (\mathbf{Q}_i - \mathbf{q}_i\mathbf{p}_i{}^{\mathsf{T}})\mathbf{Z}_i{\Lambda_i}^{-1/2}\Big\|_F^2,$$
$$\mathbf{t}_i = \mathbf{q}_i - \mathbf{R}_i\mathbf{p}_i,$$
$$\text{subject to: } \Phi^{\mathsf{T}}\Phi = \mathbb{I}, \det(\Phi) = 1,$$
$$\text{where: } \mathbf{Z}_i\Lambda_i{\mathbf{Z}_i}^{\mathsf{T}} = \mathbf{P}_i - \mathbf{p}_i\mathbf{p}_i{}^{\mathsf{T}} \text{ is the Eigen Decomposition}$$
$$\begin{bmatrix}\mathbf{P}_i & \mathbf{p}_i\\ \mathbf{p}_i{}^{\mathsf{T}} & 1\end{bmatrix} = \sum_{j=1}^{m}\Omega_{ij}, \quad \begin{bmatrix}\mathbf{Q}_i & \mathbf{q}_i\\ \mathbf{p}_i{}^{\mathsf{T}} & 1\end{bmatrix} = \sum_{j=1}^{m}\mathbf{M}_j\Omega_{ij}.$$

Relaxing the special orthogonal constraint $\Phi^{\mathsf{T}}\Phi = \mathbb{I}, \det(\Phi) = 1$ yields the linear least square solution: $\mathbf{R}_i = (\mathbf{Q}_i - \mathbf{q}_i\mathbf{p}_i{}^{\mathsf{T}})(\mathbf{P}_i - \mathbf{p}_i\mathbf{p}_i{}^{\mathsf{T}})^{-1}$. Substituting this solution to $\mathbf{t}_i = \mathbf{q}_i - \mathbf{R}_i\mathbf{p}_i$ and simplifying the equation yields:

$$\Theta(\mathcal{M}, \mathcal{O}_i) = \begin{bmatrix}\mathbf{R}_i & \mathbf{t}_i\\ 0 & 1\end{bmatrix} = \sum_{j=1}^{m}\mathbf{M}_j\Omega_{ji}\Big(\sum_{j'=1}^{m}\Omega_{j'i}\Big)^{-1}$$

We define the *linearization maps*:

$$\Omega'_{ji} = \Omega_{ji}\Big(\sum_{j'=1}^{m}\Omega_{j'i}\Big)^{-1}, \quad \Delta'_{jk} = \Delta_{jk}\Big(\sum_{j'=1}^{m}\Delta_{j'k}\Big)^{-1} \quad (3)$$

Using these maps, we can linearize the DDM transformations $\Theta(\mathcal{M}, \mathcal{O}_i) = \sum_{j=1}^{m}\mathbf{M}_j\Omega'_{ji}$ and $\Theta(\mathcal{M}, \mathcal{D}_k) = \sum_{j=1}^{m}\mathbf{M}_j\Delta'_{jk}$. Substituting these approximations to Eq. (2) yields:

$$E = \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{\gamma=1}^{m}\sum_{i=1}^{n}\int_{\mathbb{S}_\gamma}\Big\|\sum_{j=1}^{m}\mathbf{M}_j\Big(\sum_{k=1}^{p}a_{ki}\Delta'_{jk} - \Omega'_{ji}\Big)\mathbf{u}_i\Big\|_2^2 \quad (4)$$

Note that we only linearize DDM models for compression but not at run-time. The linearization can be viewed as the extracted features for the compression.

### 3.4 Hierarchical Mapping

Let $\mathbf{M}_\gamma \in \mathbb{R}^{4\times4}$ be the world transformation matrix of bone $\gamma$ in the set of example poses $\mathbb{S}_\gamma$. This transformation is only propagated to descendent bones while all other (non-descendent) bones have no transformation (identity). The world transformation matrix for an arbitrary bone $j$ is:

$$\mathbf{M}_j = (1 - h_{\gamma j})\mathbb{I} + h_{\gamma j}\mathbf{M}_\gamma$$
$$= \mathbb{I} + h_{\gamma j}(\mathbf{M}_\gamma - \mathbb{I}),$$
$$\text{where: hierarchy } h_{\gamma j} \text{ is defined in Eq. (1),}$$
$$\mathbb{I} \text{ is the identity matrix.}$$

Substituting to Eq. (4) yields:

$$E = \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{i=1}^{n}\sum_{\gamma=1}^{m}\int_{\mathbb{S}_\gamma}\Big\|\sum_{j=1}^{m}\Big(\sum_{k=1}^{p}a_{ki}\Delta'_{jk}-\Omega'_{ji}\Big)\mathbf{u}_i + \sum_{j=1}^{m}h_{\gamma j}(\mathbf{M}_\gamma-\mathbb{I})\Big(\sum_{k=1}^{p}a_{ki}\Delta'_{jk}-\Omega'_{ji}\Big)\mathbf{u}_i\Big\|_2^2$$

We have, by construction, $\sum_{j=1}^{m}\Delta'_{jk}=\sum_{j=1}^{m}\Omega'_{ji}=\mathbb{I}$, and by the affinity constraint of the LBS layer, $\sum_{k=1}^{p}a_{ki}=1$, therefore, $\sum_{j=1}^{m}(\sum_{k=1}^{p}a_{ki}\Delta'_{jk}-\Omega'_{ji})=0$, and:

$$E = \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{i=1}^{n}\sum_{\gamma=1}^{m}\int_{\mathbb{S}_\gamma}\Big\|(\mathbf{M}_\gamma-\mathbb{I})\sum_{j=1}^{m}h_{\gamma j}\Big(\sum_{k=1}^{p}a_{ki}\Delta'_{jk}-\Omega'_{ji}\Big)\mathbf{u}_i\Big\|_2^2$$
$$= \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{i=1}^{n}\sum_{\gamma=1}^{m}\int_{\mathbb{S}_\gamma}\Big\|(\mathbf{M}_\gamma-\mathbb{I})\Big(\sum_{k=1}^{p}a_{ki}\Delta''_{\gamma k}-\Omega''_{\gamma i}\Big)\mathbf{u}_i\Big\|_2^2, \quad (5)$$

where $\Omega''_{\gamma i}$ and $\Delta''_{\gamma k}$ are *hierarchical maps*:

$$\Omega''_{\gamma i}=\sum_{j=1}^{m}h_{\gamma j}\Omega'_{ji}, \quad \Delta''_{\gamma k}=\sum_{j=1}^{m}h_{\gamma j}\Delta'_{jk} \quad (6)$$

### 3.5 Coordinate Changing

For each set $\mathbb{S}_\gamma$, we want to sample bone $\gamma$ around joint $\gamma$ so that the sample mean of $\mathbf{M}_\gamma$ is $\mathbf{B}_\gamma$. It can be done by performing a coordinate change $\mathbf{M}_\gamma=\mathbf{B}_\gamma\overline{\mathbf{M}}_\gamma{\mathbf{B}_\gamma}^{-1}$, and then sampling on $\overline{\mathbf{M}}_\gamma$.

From this coordinate change, we infer: $\mathbf{M}_\gamma-\mathbb{I}=\mathbf{B}_\gamma\overline{\mathbf{M}}_\gamma{\mathbf{B}_\gamma}^{-1}-\mathbb{I}=\mathbf{B}_\gamma(\overline{\mathbf{M}}_\gamma-\mathbb{I}){\mathbf{B}_\gamma}^{-1}$. Because the leading $\mathbf{B}_\gamma$ is a rigid transformation, we can remove it from the objective function (5) without changing its L2 norm:

$$E = \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{i=1}^{n}\sum_{\gamma=1}^{m}\int_{\mathbb{S}_\gamma}\Big\|(\overline{\mathbf{M}}_\gamma-\mathbb{I}){\mathbf{B}_\gamma}^{-1}\Big(\sum_{k=1}^{p}a_{ki}\Delta''_{\gamma k}-\Omega''_{\gamma i}\Big)\mathbf{u}_i\Big\|_2^2$$
$$= \frac{1}{n\sum_{\gamma=1}^{m}|\mathbb{S}_\gamma|}\sum_{i=1}^{n}\sum_{\gamma=1}^{m}\int_{\mathbb{S}_\gamma}\Big\|(\overline{\mathbf{M}}_\gamma-\mathbb{I})\Big(\sum_{k=1}^{p}a_{ki}\Delta'''_{\gamma k}-\Omega'''_{\gamma i}\Big)\mathbf{u}_i\Big\|_2^2, \quad (7)$$

where $\Omega'''_{\gamma i}$ and $\Delta'''_{\gamma k}$ are *coordinate changing maps*:

$$\Omega'''_{\gamma i}={\mathbf{B}_\gamma}^{-1}\Omega''_{ji}, \quad \Delta'''_{\gamma k}={\mathbf{B}_\gamma}^{-1}\Delta''_{jk} \quad (8)$$

### 3.6 Continuous Sampling

Letting $\textcircled{\scriptsize ☺} = \Big(\sum_{k=1}^{p}a_{ki}\Delta'''_{\gamma k}-\Omega'''_{\gamma i}\Big)\mathbf{u}_i$, the mean squared error on example pose set $\mathbb{S}_\gamma$ is:

$$\frac{1}{|\mathbb{S}_\gamma|}\int_{\mathbb{S}_\gamma}\big\|(\overline{\mathbf{M}}_\gamma-\mathbb{I})\,\textcircled{\scriptsize ☺}\big\|_2^2=\frac{1}{|\mathbb{S}_\gamma|}\int_{\mathbb{S}_\gamma}\mathrm{tr}\left(\textcircled{\scriptsize ☺}^{\mathsf{T}}(\overline{\mathbf{M}}_\gamma-\mathbb{I})^{\mathsf{T}}(\overline{\mathbf{M}}_\gamma-\mathbb{I})\,\textcircled{\scriptsize ☺}\right)$$
$$=\mathrm{tr}\left(\textcircled{\scriptsize ☺}^{\mathsf{T}}\left(\frac{1}{|\mathbb{S}_\gamma|}\int_{\mathbb{S}_\gamma}(\overline{\mathbf{M}}_\gamma-\mathbb{I})^{\mathsf{T}}(\overline{\mathbf{M}}_\gamma-\mathbb{I})\right)\textcircled{\scriptsize ☺}\right) \quad (9)$$

Given the rotation range $[-\bar{r}_\gamma..\bar{r}_\gamma]$ for bone $\overline{\mathbf{M}}_\gamma$, we can represent the rotation part of $\overline{\mathbf{M}}_\gamma$ as the product of 3 Euler angles rotation matrices $\mathbf{R}_x\mathbf{R}_y\mathbf{R}_x$, where $\mathbf{R}_x,\mathbf{R}_y,\mathbf{R}_z$ are uniformly sampled of x, y, z rotations in range of $[-\bar{r}_\gamma..\bar{r}_\gamma]$. Also, the x, y, z translation parts of $\overline{\mathbf{M}}_\gamma$ are uniformly sampled in the given translation range of $[-\bar{t}_\gamma..\bar{t}_\gamma]$. We can directly compute the multiple integral:

$$\mathbf{S}_\gamma{}^2=\frac{1}{|\mathbb{S}_\gamma|}\int_{\mathbb{S}_\gamma}(\overline{\mathbf{M}}_\gamma-\mathbb{I})^{\mathsf{T}}(\overline{\mathbf{M}}_\gamma-\mathbb{I})$$
$$=\frac{1}{8\bar{r}_\gamma{}^3\bar{t}_\gamma{}^3}\iiint_{[-\bar{t}_\gamma..\bar{t}_\gamma]^3}\iiint_{[-\bar{r}_\gamma..\bar{r}_\gamma]^3}\left(\begin{bmatrix}\mathbf{R}&\mathbf{t}\\0&1\end{bmatrix}-\mathbb{I}\right)^{\mathsf{T}}\left(\begin{bmatrix}\mathbf{R}&\mathbf{t}\\0&1\end{bmatrix}-\mathbb{I}\right)\mathrm{d}\mathbf{R}\,\mathrm{d}\mathbf{t}$$
$$=\begin{bmatrix}\alpha_\gamma{}^2&0&0&0\\0&\alpha_\gamma{}^2&0&0\\0&0&\alpha_\gamma{}^2&0\\0&0&0&\beta_\gamma{}^2\end{bmatrix},\ \text{where: }\alpha_\gamma{}^2=2-2\frac{\sin^2\bar{r}_\gamma}{\bar{r}_\gamma{}^2},\ \beta_\gamma{}^2=4\bar{t}_\gamma{}^2.$$

Note that the rotation range of each Euler angle can be set independently. This could be useful to handle hinge joints (one Euler angle is locked), or to add discrete example poses (three Euler angles are fixed). We will leave this extension for the future.

Substituting $\mathbf{S}_\gamma{}^2$ to Eq. (9) and then substituting Eq. (9) and ☺ to Eq. (7) yields:

$$E=\frac{1}{nm}\sum_{i=1}^{n}\sum_{\gamma=1}^{m}\Big\|\mathbf{S}_\gamma\Big(\sum_{k=1}^{p}a_{ki}\Delta'''_{\gamma k}-\Omega'''_{\gamma i}\Big)\mathbf{u}_i\Big\|_2^2$$

We define the *continuous sampling maps*:

$$\Omega''''_{\gamma i}=\mathbf{S}_\gamma\Omega'''_{ji}, \quad \Delta''''_{\gamma k}=\mathbf{S}_\gamma\Delta'''_{jk} \quad (10)$$

Combining all maps (3), (6), (8), and (10) yields the final objective function:

$$E=\frac{1}{nm}\sum_{i=1}^{n}\sum_{\gamma=1}^{m}\Big\|\Big(\sum_{k=1}^{p}a_{ki}\Delta''''_{\gamma k}-\Omega''''_{\gamma i}\Big)\mathbf{u}_i\Big\|_2^2, \quad (11a)$$

$$\text{where: }\Omega''''_{\gamma i}=\mathbf{S}_\gamma{\mathbf{B}_\gamma}^{-1}\Big(\sum_{j=1}^{m}h_{\gamma j}\Omega_{ji}\Big)\Big(\sum_{j=1}^{m}\Omega_{ji}\Big)^{-1}, \quad (11b)$$

$$\Delta''''_{\gamma k}=\mathbf{S}_\gamma{\mathbf{B}_\gamma}^{-1}\Big(\sum_{j=1}^{m}h_{\gamma j}\Delta_{jk}\Big)\Big(\sum_{j=1}^{m}\Delta_{jk}\Big)^{-1}, \quad (11c)$$

$$\mathbf{S}_\gamma=\begin{bmatrix}\alpha_\gamma&0&0&0\\0&\alpha_\gamma&0&0\\0&0&\alpha_\gamma&0\\0&0&0&\beta_\gamma\end{bmatrix},\alpha_\gamma=\sqrt{2-2\frac{\sin^2\bar{r}_\gamma}{\bar{r}_\gamma{}^2}},\beta_\gamma=2\bar{t}_\gamma \quad (11d)$$

## 4 OPTIMIZATION

Generally, the objective function (11a) resembles a matrix factorization problem, and it can be solved by different strategies, depending on the constraints, such as multiplicative updates [Lee and Seung 2000], alternating least squares [Kim and Park 2008], or block coordinate descent [Aharon et al. 2006]. We choose the sparse coding optimization for skinning solution [Le and Deng 2013]. An overview of our optimization is presented in Algorithm 1, and all steps are detailed in the following sections.

**ALGORITHM 1:** Sparse Coding Optimization

**input** : $\{\Omega''''_{\gamma i} | \gamma = 1..m, i = 1..n\}$, $\{\mathbf{u}_i | i = 1..n\}$, parameter $p$,
**output**: $\{\Delta_{\gamma k} | \gamma = 1..m, k = 1..p\}$, $\{a_{ki} | k = 1..p, i = 1..n\}$.

```
Initialize a feasible solution ;                                  // §4.1
repeat
    foreach vertex i = 1..n do
        Constrained least squares solve {a_ki | k = 1..p} ;       // §4.2
    end
    foreach virtual bone k = 1..p do
        Update {Δ_γk | γ = 1..m} ;                                // §4.3
    end
    foreach virtual bone k = 1..p do
        Re-initialize bone k if Σ_{i=1}^{n} a_ik^2 < ε ;          // §4.4
    end
until converged or maximum number of iterations reached;
```

### 4.1 Initialization

In this step, we generate the set of $p$ virtual bones and their multi-weights for the first DDM layer, where $p$ is provided by the user. The second LBS layer is initialized as the rigid bind of each vertex to the best virtual bone, i.e. LBS weights $a_{ki} \in \{0, 1\}$. Our initialization includes two steps:

First, we use farthest points sampling [Schlömer et al. 2011] to select $p$ vertices and assign multi-weights of each selected vertex $\mathcal{O}_i = \{\Omega_{ji} \,|\, j = 1..m\}$ to the multi-weights of one virtual bone $\mathcal{D}_k = \{\Delta_{jk} \,|\, j = 1..m\}$. The distance between virtual bone $k$ and vertex $i$ is computed by:

$$\text{distance}(k, i) = \sum_{\gamma=1}^{m} \left\| (\Delta''''_{\gamma k} - \Omega''''_{\gamma i}) \mathbf{u}_i \right\|_2^2$$

Then, we perform K-means clustering [Lloyd 1982] to refine the initialization. The K-means clustering alternatively performs an assignment step and an update step. In the assignment step, each vertex $i$ is assigned to the closest virtual bone, i.e. setting LBS weight $a_{\hat{k}i} = 1$ for closest virtual bone $\hat{k}$ and $a_{ki} = 0, \forall k \neq \hat{k}$. In the update step, the multi-weights of each virtual bone $k$ are updated as the average multi-weights of vertices assigned to $k$, specifically:

$$\Delta_{jk} = \frac{1}{\sum_{i=1}^{n} a_{ki}} \sum_{i=1}^{n} a_{ki} \Omega_{ji}$$

Although the above $\Delta_{jk}$ update is sub-optimized, $\Delta$ will be further refined (§4.3). Therefore, this simple update is sufficient.

### 4.2 Scalar-weights ($a$) Update

For robustness, we use the solution proposed by Le and Deng [2014], which adds a smoothness regularization to guide the selection of non-zero weights:

$$E' = E + \alpha \mathbf{A}\mathbf{L}\mathbf{A}^{\mathsf{T}} ,$$

where: $\mathbf{A} \in \mathbb{R}^{p \times n}$ is the matrix form of LBS weights $a$,
$\mathbf{L} \in \mathbb{R}^{n \times n}$ is the Laplacian matrix of the mesh,
$\alpha$ is the user-defined regularization strength.

This objective function is optimized by sequentially solving scalar skinning weights for each vertex, where each per-vertex solver is formulated as a non-negative least squares problem with an affinity constraint.

### 4.3 Multi-weights ($\Delta$) Update

In this step, we need to minimize the objective function (11a) with respect to $\{\Delta_{\gamma k} \,|\, \gamma = 1..m, k = 1..p\}$ when $\{a_{ki} \,|\, k = 1..p, i = 1..n\}$ are fixed. Unfortunately, this function is linear with respect to $\Delta''''_{\gamma k}$ but not to $\Delta_{\gamma k}$. In addition, we cannot directly solve the inverse of the map in Eq. (11c) to get $\Delta_{\gamma k}$ from $\Delta''''_{\gamma k}$ due to the symmetric constraints on $\Delta_{\gamma k}$.

Similar to [Le and Deng 2013; Mairal et al. 2010], $\Delta''''_{\gamma k}$ can be sequentially updated for each virtual bone $k$ while keeping others $\{\Delta''''_{\gamma k'}, \forall k' \neq k\}$ fixed. The update rule for $\Delta''''_{\gamma k}$ to reduce the objective function (11a) is:

$$\Delta''''^{\{t+1\}}_{\gamma k} \leftarrow {\Phi_{kk}}^{-1} \Big( \Gamma''''_{\gamma k} - \sum_{\lambda=1}^{p} \Delta''''^{\{t\}}_{\gamma \lambda} \Phi_{\lambda k} \Big) + \Delta''''^{\{t\}}_{\gamma k} ,$$

$$\text{where: } \Phi_{\lambda k} = \sum_{i=1}^{n} a_{\lambda i} a_{ki} \mathbf{u}_i^{\mathsf{T}} \mathbf{u}_i , \quad \Gamma''''_{\gamma k} = \sum_{i=1}^{n} \Omega''''_{\gamma i} a_{ki} \mathbf{u}_i^{\mathsf{T}} \mathbf{u}_i .$$

Because the hierarchical map (Eq. (6)), the coordinate changing map (Eq. (8)), and the continuous sampling map (Eq. (10)) are linear and they have the same action on $\gamma$, we can invert these maps in the above update rule and yield:

$$\Delta'^{\{t+1\}}_{\gamma k} \leftarrow {\Phi_{kk}}^{-1} \Big( \Gamma'_{\gamma k} - \sum_{\lambda=1}^{p} \Delta'^{\{t\}}_{\lambda k} \Phi_{\lambda k} \Big) + \Delta'^{\{t\}}_{\gamma k} ,$$

$$\text{where: } \Gamma'_{\gamma k} = \sum_{i=1}^{n} \Omega'_{\gamma i} a_{ki} \mathbf{u}_i^{\mathsf{T}} \mathbf{u}_i .$$

Multiplying both sides with $\sum_{j=1}^{m} \Delta_{jk}^{\{t+1\}}$ and using approximations $\sum_{j=1}^{m} \Delta_{jk}^{\{t+1\}} \approx \sum_{j=1}^{m} \Delta_{jk}^{\{t\}}$ yields:

$$\begin{aligned}
\Delta_{\gamma k}^{\{t+1\}} &\leftarrow \Big( {\Phi_{kk}}^{-1} \Big( \Gamma'_{\gamma k} - \sum_{\lambda=1}^{p} \Delta'^{\{t\}}_{\gamma \lambda} \Phi_{\lambda k} \Big) + \Delta'^{\{t\}}_{\gamma k} \Big) \Big( \sum_{j=1}^{m} \Delta_{jk}^{\{t+1\}} \Big) \\
&\approx \Big( {\Phi_{kk}}^{-1} \Big( \Gamma'_{\gamma k} - \sum_{\lambda=1}^{p} \Delta'^{\{t\}}_{\gamma \lambda} \Phi_{\lambda k} \Big) + \Delta'^{\{t\}}_{\gamma k} \Big) \Big( \sum_{j=1}^{m} \Delta_{jk}^{\{t\}} \Big) \\
&= {\Phi_{kk}}^{-1} \Big( \Gamma'_{\gamma k} - \sum_{\lambda=1}^{p} \Delta'^{\{t\}}_{\gamma \lambda} \Phi_{\lambda k} \Big) \Big( \sum_{j=1}^{m} \Delta_{jk}^{\{t\}} \Big) + \Delta_{\gamma k}^{\{t\}} \\
&\approx \frac{1}{\det(\Phi_{kk})} \Big( \Gamma'_{\gamma k} - \sum_{\lambda=1}^{p} \Delta'^{\{t\}}_{\gamma \lambda} \Phi_{\lambda k} \Big) \det \Big( \sum_{j=1}^{m} \Delta_{jk}^{\{t\}} \Big) + \Delta_{\gamma k}^{\{t\}} ,
\end{aligned}$$

where: $\det(\cdot)$ denotes the determinant of the matrix.

The determinants are used to approximate the pre-multiplied matrix inverse ${\Phi_{kk}}^{-1}$ and the post-multiplied matrix $\big( \sum_{j=1}^{m} \Delta_{jk}^{\{t\}} \big)$ so that the updated $\Delta_{\gamma k}^{\{t+1\}}$ is closer to a symmetric matrix.

Finally we normalize the multi-weights to enforce symmetry and affinity constraints:

$$\Delta_{\gamma k}^{\{t+2\}} \leftarrow \frac{1}{2\sum_{j=1}^{m}\left(\Delta_{\gamma k}^{\{t+1\}}\right)_{4,4}}\left(\Delta_{\gamma k}^{\{t+1\}} + \Delta_{\gamma k}^{\{t+1\}^{\mathsf{T}}}\right)$$

For robustness, the multi-weights update above is skipped for virtual bone $k$ if $\det(\Phi_{kk}) < \epsilon$. Most likely, these bones will be degenerate and they will be re-initialized (§4.4).

### 4.4 Re-initialization

Similar to the bone pruning idea proposed by Le and Deng [2014], we can remove a virtual bone $k$ if its contribution is insignificant, i.e. $\sum_{i=1}^{n} {a_{ik}}^2 < \varepsilon$. Then, we can re-initialize bone $k$ by assigning its multi-weights to the vertex with largest compression error and resolve the LBS weights of the second layer.

## 5 FIXING FLOATING-POINT CANCELLATION

Assuming all vertex coordinates $\mathbf{u}_i$ are in range of $\pm r$. By construction, each multi-weight $\Delta_{jk}$ is a weighted sum of $\mathbf{u}_i{\mathbf{u}_i}^{\mathsf{T}}$, therefore, elements in the $3\times3$ top left corner of $\Delta_{jk}$ are in range of $\pm r^2$. From the DDM transformation of virtual bone $k$ on the first layer:

$$\begin{aligned}
&\mathbf{R}_k = \mathcal{U}_k{\mathcal{V}_k}^{\mathsf{T}},\ \mathbf{t}_k = \mathbf{q}_k - \mathbf{R}_k\mathbf{p}_k\ ,\\
&\text{where: } \mathbf{Q}_k - \mathbf{q}_k{\mathbf{p}_k}^{\mathsf{T}} = \mathcal{U}_k\mathcal{S}_k{\mathcal{V}_k}^{\mathsf{T}} \text{ is the SVD}\ ,\\
&\begin{bmatrix}\mathbf{Q}_k & \mathbf{q}_k\\ {\mathbf{p}_k}^{\mathsf{T}} & 1\end{bmatrix} = \sum_{j=1}^{m}\mathbf{M}_j\Delta_{jk}\ ,
\end{aligned} \tag{12}$$

we can infer that both $\mathbf{Q}_k$ and $\mathbf{q}_k{\mathbf{p}_k}^{\mathsf{T}}$ are in range of $\pm r^2$. Computing the difference $\mathbf{Q}_k - \mathbf{q}_k{\mathbf{p}_k}^{\mathsf{T}}$ may cause a loss of significance and inaccurate rotation. In practice, we observe this issue when using single precision floating-point numbers as illustrated in Fig 4. The issue is more noticeable when the bone translations or the rest pose coordinates are further from the origin because these translations cause more accumulated error in $\mathbf{Q}_k$.

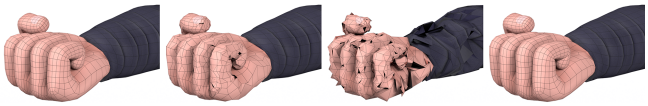

Fig. 4. An example of the floating-point cancellation issue of DDM on Ninja model. From left to right: using double precision weights, using single precision weights, using single precision weights and translating the root bone by 10m (the height of the model is 175cm), and lastly, using single precision weights with our fix. We noticed no artifacts when translating the root bone using double precision weights or using single precision weights with our fix.

Instead of using double precision numbers to fix this issue, we perform a coordinate change to the center of rotation $\mathbf{p}_k$, which is equivalent to performing translation $-\mathbf{p}_k$:

$$\Delta_{jk}^{*} = \begin{bmatrix}\mathbb{I} & -\mathbf{p}_k\\ 0 & 1\end{bmatrix}\Delta_{jk}\begin{bmatrix}\mathbb{I} & 0\\ -\mathbf{p}_k{}^{\mathsf{T}} & 1\end{bmatrix} \Rightarrow \Delta_{jk} = \begin{bmatrix}\mathbb{I} & \mathbf{p}_k\\ 0 & 1\end{bmatrix}\Delta_{jk}^{*}\begin{bmatrix}\mathbb{I} & 0\\ \mathbf{p}_k{}^{\mathsf{T}} & 1\end{bmatrix}$$

Substituting $\Delta_{jk}$ to Eq. (12) and simplifying equations yields:

$$\begin{bmatrix}\mathbf{Q}_k - \mathbf{q}_k{\mathbf{p}_k}^{\mathsf{T}} & \mathbf{q}_k\\ 0 & 1\end{bmatrix} = \sum_{j=1}^{m}\mathbf{M}_j\begin{bmatrix}\mathbb{I} & \mathbf{p}_k\\ 0 & 1\end{bmatrix}\Delta_{jk}^{*}$$

Letting $\mathbf{Q}_k^{*} = \mathbf{Q}_k - \mathbf{q}_k{\mathbf{p}_k}^{\mathsf{T}}$ yields:

$$\mathbf{R}_k = \mathcal{U}_k{\mathcal{V}_k}^{\mathsf{T}},\ \mathbf{t}_k = \mathbf{q}_k - \mathbf{R}_k\mathbf{p}_k\ , \tag{13a}$$

$$\text{where: } \mathbf{Q}_k^{*} = \mathcal{U}_k\mathcal{S}_k{\mathcal{V}_k}^{\mathsf{T}} \text{ is the SVD}\ , \tag{13b}$$

$$\begin{bmatrix}\mathbf{Q}_k^{*} & \mathbf{q}_k\\ 0 & 1\end{bmatrix} = \sum_{j=1}^{m}\mathbf{M}_j\begin{bmatrix}\mathbb{I} & \mathbf{p}_k\\ 0 & 1\end{bmatrix}\Delta_{jk}^{*}\ , \tag{13c}$$

$$\Delta_{jk}^{*} = \begin{bmatrix}\mathbb{I} & -\mathbf{p}_k\\ 0 & 1\end{bmatrix}\Delta_{jk}\begin{bmatrix}\mathbb{I} & 0\\ -\mathbf{p}_k{}^{\mathsf{T}} & 1\end{bmatrix}\ , \tag{13d}$$

$$\begin{bmatrix}\mathbf{P}_k & \mathbf{p}_k\\ \mathbf{p}_k{}^{\mathsf{T}} & 1\end{bmatrix} = \sum_{j=1}^{m}\Delta_{jk}\ . \tag{13e}$$

From the above equations, we modify the DDM model by these steps:

- Computing and storing the center of rotation $\mathbf{p}_k \in \mathbb{R}^3$ by Eq. (13e). This step requires storing three extra floating-point numbers per virtual bone compared to the original model.
- Transforming and storing new multi-weights $\Delta_k^{*}$ by Eq. (13d). Note that in our new model, we perform all calculations in double precision and truncate to single precision when storing new multi-weights. This reduces the storage requirements by nearly half compared to the original model.

At run-time, we compute the transformation by:

- Computing the matrix blending $\begin{bmatrix}\mathbf{Q}_k^{*} & \mathbf{q}_k\\ 0 & 1\end{bmatrix}$ in Eq. (13c).
- Computing the SVD of $\mathbf{Q}_k^{*}$ in Eq. (13b) and using it to compute the rotation and translation (Eq. (13a)). This is the key step to fix the floating-point cancellation issue, where the difference of two matrices with large entries in the old model is replaced by a direct computation of matrix $\mathbf{Q}_k^{*}$ with smaller entries.

## 6 RESULTS

We demonstrate the results on various LBS-rigged 3D models and skeletal animations as listed in Table 1. Ninja model, Ortiz model, and their animations are acquired from *Mixamo*[1]. Each model is converted to DDM skinning as reported by Le and Lewis [2019], where the LBS model is converted to rigid bind, and the smoothing parameters of DDM are manually tuned using the distances from the vertices to the associated bone. For convenience, we will use DDM to refer to DDM variant 0 for the rest of this section.

We run the experiments with the same set of parameters for all models: the number of non-zero virtual bones per vertex is set to $z = 8$, the LBS weight regularizer (§4) is set to $\alpha = 0.01$, the virtual bone update skipping threshold (§4.3) is set to $\epsilon = 16cm^2$, and the re-initialization threshold (§4.4) is set to $\varepsilon = 1$. We run the optimization for a maximum of 500 iterations or until the objective function $E$ changes less than $10^{-5}E$ in 5 consecutive iterations. However, we recommend adjusting $\alpha$, $\epsilon$, and $\varepsilon$ for models with different sizes or mesh resolutions than our tested models.

[1] https://www.mixamo.com

Table 1. The test models: $n$ denotes the number of vertices, $m$ denotes the number of (master) bones, and $h$ denotes the height of the model. Each model was tested with three different numbers of virtual bones $p$. The root mean squared fitting errors $RMSE_{fit} = \sqrt{E}$, where $E$ is described in Eq. (11a). Our compression times are reported on a computer with an 8-core 3.00 GHz CPU. Our compression algorithm was implemented in C++ using the *Eigen* library[2].

| *Model* | *Test* | $p$ | $RMSE_{fit}$ | *Time* |
|---|---|---|---|---|
|  $n$ = 12465 | $\text{Ninja}_{100}$ | 100 | 0.406 cm | 1.8 mins |
| $m$ = 52 | $\text{Ninja}_{200}$ | 200 | 0.248 cm | 2.9 mins |
| $h$ = 175 cm | $\text{Ninja}_{500}$ | 500 | 0.159 cm | 2.7 mins |
|  $n$ = 24798 | $\text{Ortiz}_{100}$ | 100 | 1.060 cm | 3.9 mins |
| $m$ = 65 | $\text{Ortiz}_{200}$ | 200 | 0.594 cm | 5.8 mins |
| $h$ = 185 cm | $\text{Ortiz}_{500}$ | 500 | 0.391 cm | 15.2 mins |
|  $n$ = 13694 | $\text{Generic}_{200}$ | 200 | 0.250 cm | 3.2 mins |
| $m$ = 100 | $\text{Generic}_{500}$ | 500 | 0.130 cm | 14.0 mins |
| $h$ = 151 cm | $\text{Generic}_{1K}$ | 1000 | 0.073 cm | 77.7 mins |

For Ninja and Ortiz models, we set the rotation ranges for the hip and all spine joints to $[-\bar{r}_Y..\bar{r}_Y] = [-30°..30°]$ and rotation ranges for other joints to $[-120°..120°]$. For Generic model, we compute the rotation ranges from the tested animation. These different settings are purely for testing purposes. In addition, we set all translation ranges $[-\bar{t}_Y..\bar{t}_Y]$ to 0 because our test models contain a single root bone.

## 6.1 Virtual Bones

In this paper, we use red cubes to visualize virtual bones. At the rest pose, virtual bones are aligned with the coordinate axes, and the center for each virtual bone is put at the weighted centroid of the vertices associated with it: $\mathbf{c}_k = \left(\sum_{i=1}^{n} {a_{ki}}^2 \mathbf{u}_i\right) / \left(\sum_{i=1}^{n} {a_{ki}}^2\right)$. This setting is purely for visualization purposes and has no effect on the quality or the performance of the results. Fig. 5 visualizes the virtual bones for all tests. This visualization shows the adaptation of our compression model to different deformation behaviours of the input DDM models, where highly deformable regions have a higher density of virtual bones.

## 6.2 DDM Effects

Our compression can closely approximate DDM, and it can reproduce common effects of DDM, such as skin sliding (Fig. 6) or negative bulging (Fig. 7). For better visibility of these effects in motion, see our accompanying video. Note that using pre-computed weights DDM variant 5 for LBS can slightly improve the skinning quality; however, it comes with the cost of extra skinning weights, i.e. a denser model. We also include this setup and refer to it as "Dense LBS" as opposed to "Sparse LBS", which is the original LBS model acquired from Mixamo or rigged by our artists.

## 6.3 Storage

We report the data storage compression in Table 2. Because the multi-weights of DDM are symmetric, they can be stored in a compact form using only 10 floats per multi-weight (instead of 16 floats). The weights are stored in sparse structures with indices to non-zero elements. However, efficient GPU implementations may use more on-device storage due to memory access and scheduling problems, e.g. storing full $4 \times 4$ matrices or denser structures.

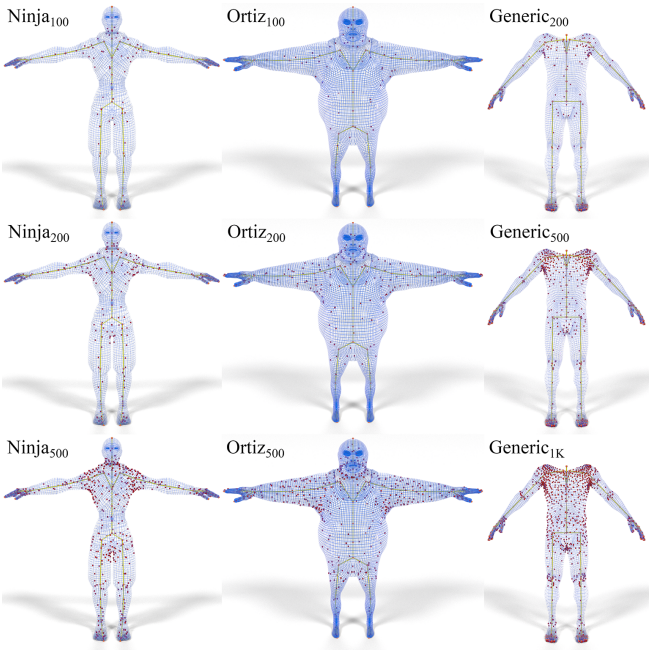


Fig. 5. Visualization of virtual bone distributions. Notice how virtual bones (red) are densely distributed in highly deformable regions, such as shoulders, hips, arm joints and leg joints.

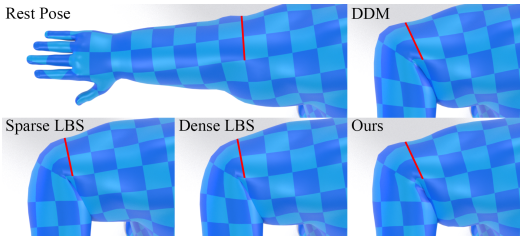


Fig. 6. Our compression model can closely approximate DDM deformation, therefore, it can also create skin sliding effects (indicated by the slope of red lines). Notice how DDM and our model can create a skin sliding effect while not over smoothing the elbow's silhouette like LBS. Our method is shown on $\text{Ortiz}_{500}$ test data.


Fig. 7. Utilizing "negative bulging" effect, DDM (variant 0) and our model can resolve the collapsing issue near the hip joint. Dense LBS, which is DDM variant 5, shows less artifacts than Sparse LBS, which is the original data from Mixamo. Our method is shown on $\text{Ninja}_{500}$ test data.

Table 2. Storage cost comparison between our model, DDM and LBS. In the last two columns: Sparse LBS denotes the original data, and Dense LBS denotes the LBS generated by DDM variant 5. Model storage uses 32 bit floats for scalar values and 32 bit integers for indices.

| *Test* | *DDM* | *Ours* | *Ratio* | *Sparse LBS* | *Dense LBS* |
|---|---|---|---|---|---|
| $\text{Ninja}_{100}$ | 7119 KB | 590 KB | 12.1x | 284 KB | 697 KB |
| $\text{Ninja}_{200}$ | | 740 KB | 9.6x | | |
| $\text{Ninja}_{500}$ | | 993 KB | 7.2x | | |
| $\text{Ortiz}_{100}$ | 21697 KB | 1201 KB | 18.1x | 547 KB | 2245 KB |
| $\text{Ortiz}_{200}$ | | 1576 KB | 13.8x | | |
| $\text{Ortiz}_{500}$ | | 1965 KB | 11.2x | | |
| $\text{Generic}_{200}$ | 8245 KB | 714 KB | 11.5x | 623 KB | 943 KB |
| $\text{Generic}_{500}$ | | 1028 KB | 8.0x | | |
| $\text{Generic}_{1K}$ | | 1508 KB | 5.5x | | |

## 6.4 Run-time Performance

In Table 3, we compare the run-time performance of different models. For the CPU benchmark, all methods were implemented in C++ using the *Eigen* library[2]. For DDM and our model, we store the full multi-weight matrix (16 floats). For the GPU benchmark, all methods were implemented by DirectX 12's HLSL shaders. Each symmetric multi-weight is stored as 10 floats. For both benchmarks, we used the $3 \times 3$ Singular Value Decomposition proposed by McAdams et al. [2011a] with 4 iterations.

From this table, we can see an approximated performance improvement of 2 times when using our compression compared to the original DDM model. In some cases, the performance of our model is on par with LBS. Note that the performance of LBS also depends on the sparseness of the skinning weights. Generally, denser weights produce smoother deformation.

Table 3. Run-time performance comparison between our model, DDM and LBS. In the last two columns: Sparse LBS denotes the original data, and Dense LBS denotes the LBS generated by DDM variant 5. The average skinning time per vertex (in nanoseconds) and the ratio are reported in the form of $g/c$, where $g$ is the performance recorded on an NVIDIA GeForce RTX 3090 GPU and $c$ is the performance reported on a single-core Intel i7-5960X 3.00 GHz CPU. The average CPU times were calculated on a batch of 100 characters and the average GPU times were calculated on a batch of 300 characters.

| *Test* | *DDM* | *Ours* | *Ratio* | *Sparse LBS* | *Dense LBS* |
|---|---|---|---|---|---|
| $\text{Ninja}_{100}$ | 1.45/952 | 0.79/270 | 1.8x/3.5x | 0.81/116 | 0.83/189 |
| $\text{Ninja}_{200}$ | | 0.79/283 | 1.8x/3.4x | | |
| $\text{Ninja}_{500}$ | | 0.83/366 | 1.7x/2.6x | | |
| $\text{Ortiz}_{100}$ | 1.13/1114 | 0.48/256 | 2.4x/4.4x | 0.48/110 | 0.52/310 |
| $\text{Ortiz}_{200}$ | | 0.51/320 | 2.2x/3.5x | | |
| $\text{Ortiz}_{500}$ | | 0.54/337 | 2.1x/3.3x | | |
| $\text{Generic}_{200}$ | 1.44/958 | 0.73/264 | 2.0x/3.6x | 0.71/198 | 0.73/255 |
| $\text{Generic}_{500}$ | | 0.77/333 | 1.9x/2.9x | | |
| $\text{Generic}_{1K}$ | | 0.86/426 | 1.7x/2.2x | | |

[2]https://eigen.tuxfamily.org/

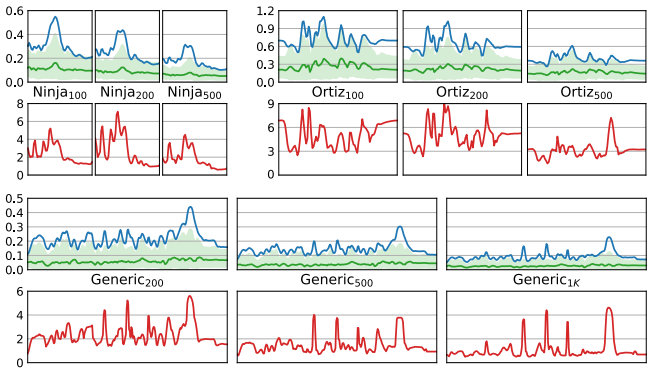


Fig. 8. Plot of reconstruction errors on animation sequences, i.e. the distance of deformed positions between our model and DDM. x-axes represent time frames and y-axes represent errors (in cm). At each frame: green lines indicate median errors of all vertices, light green regions indicate error values between the first quartile ($25^{\text{th}}$ percentile) to the third quartile ($75^{\text{th}}$ percentile), blue lines indicate root mean squared errors of all vertices, red lines indicate the maximum errors. The length for animations are: 55 frames (Ninja), 96 frames (Ortiz), and 261 frames (Generic).

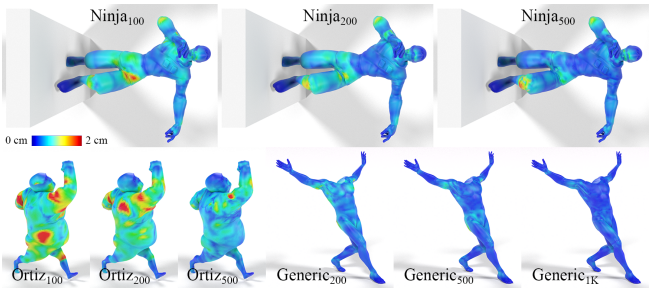


Fig. 9. Visualization of reconstruction error on selected poses, i.e. the distance of deformed positions between our model and DDM.

## 6.5 Deformation Error

Fig. 8 plots the reconstruction error over test animation sequences and Fig. 9 shows selected poses in these animations. Please see our accompanying video for these animations. Because our compression does not use explicit example poses, all test animations are considered novel to our model as judged by the common standards of supervised learning. From this figure, we can clearly see that our approximations get better as we increase the number of virtual bones. Because our model solves the least squares problem, maximum errors are significantly larger than mean errors or root mean squared errors. But we believe this issue can be improved using a smaller norm for the objective function, e.g. L1 norm, and the new function can be optimized with other L1 optimization techniques, e.g. iteratively re-weighted least squares [Bjorck 1996].

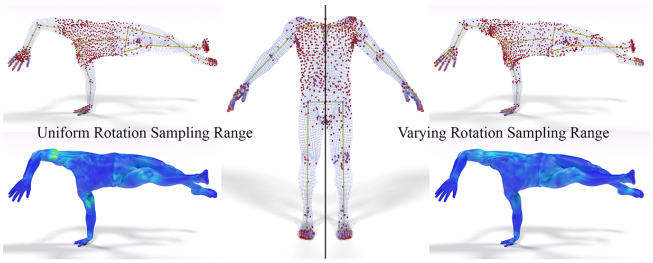


Fig. 10. The effect on using different range of motions (ROMs). Left: rotation ROMs for every joints are uniformly set to $[-120°..120°]$. Right: rotation ROMs are computed from the tested animation. With a good approximation for ROMs, our method can find a better distribution of virtual bones, e.g. using more virtual bones on shoulders, arm joints and leg joints and less virtual bones on the body, resulting lower compression error as visualized at the bottom.

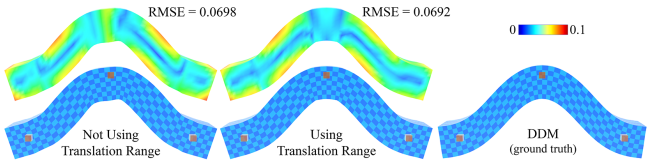


Fig. 11. Setting translation sampling range for a model with unorganized bones can generate slightly better approximation near the bones. The result on the left was generated with fixed bone translations (zero range) while the result in the middle was generated by setting the bone translation sampling ranges to $[-2..2]$, where the height of the bar is 1. In both tests, we set the number of virtual bones to 10 and rotation sampling ranges to $[-90°..90°]$.

## 7 VALIDATION AND DISCUSSION

In this section, we break down the algorithm to show the effect of individual steps. Generally, we compare the results when running our method with or without each formulation. We test our method with an extra example (a bar) with $p = 20$ virtual bones using the same parameters as we mentioned previously. This model is shown in Fig. 2.

### 7.1 Range of Motion Map

Fig. 10 shows a comparison between setting the same range of motions (ROMs) and setting different ROMs for every joint. From the continuous sampling maps in Eq. (10), we can see that using the same ROMs is equivalent to multiplying all terms in the objective function with the same constant, which is the same as omitting this map in the formulation. We can see that omitting this map significantly affects the quality of the result, as shown in Fig. 10.

In Fig. 11, we demonstrate the effect of setting the translation range $[-\bar{t}_\gamma..\bar{t}_\gamma]$ (in §3.6) for a model with unorganized bones. This setup only provided a small improvement on the approximation error for poses with large bone translations and small bone rotations. The improvement is more noticeable on vertices that are closer to the bones. One possible reason is that rotating bones around their center of rotation moves vertices further from the bones more significantly, which can overcome the effect of translating bones in example poses.

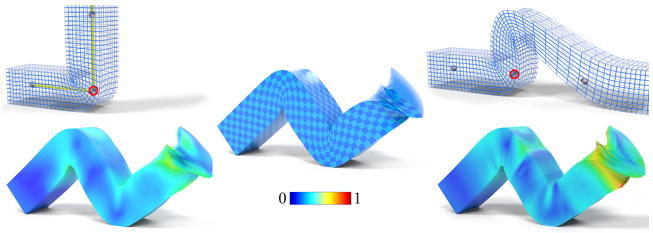


Fig. 12. Our result with hierarchical map (bottom left) is significantly better than the result without hierarchical map (bottom right) due to proper example poses, e.g. top left. The example poses generated without hierarchical map might not be in the proper animation range, e.g. top right. Both example poses here are generated by rotating the red joint by 90°. The ground truth (DDM) is shown in the middle.

### 7.2 Hierarchical Map

In Fig. 12, we show an example of running our algorithm without hierarchical map (Eq. (6)). This is equivalent to discarding the skeleton structure so the transformation on one bone will not propagate down the stream when sampling the pose space. As the result, the set of training examples will include poses that do not appear in the valid range of motion, which will bias the compression to sub-optimized models.

### 7.3 Coordinate Changing Map

Fig. 13 shows an example of discarding both coordinate changing map (Eq. (8)) and hierarchical map from our formulation. As shown in the figure, the result gets worse than discarding hierarchical map alone. This is because the training data deviates further from the valid range of animation.

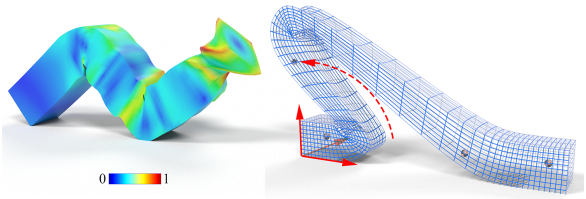


Fig. 13. The result without coordinate changing map and without hierarchical map (left) is worse than the one with coordinate changing map (as shown in Fig. 9). The example poses generated without coordinate changing map only rotate bones around the origin as the example showed on the right.

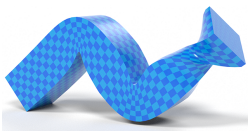

Fig. 14. The result of only LBS weights update for $p = 4$ virtual bones (same as number of master bones) shows that our mapping, including the linearization map, works reasonably well.

### 7.4 Linearization Map

Theoretically, our linearization (Eq. (3)) is not optimized for non-linear operators in DDM. However, it works reasonably well for our purpose. In Fig. 14, we show a test where we generate only $p = 4$ virtual bones, which is the same as the number of master bones $m = 4$. We manually initialize each virtual bone to the group of vertices that mostly follow the rigid transformation of one master bone. Then, we update LBS weights for the second layer without any smoothness regularizer ($\alpha = 0$) and without any multi-weight update. The deformation of the output model in Fig. 14 is smooth and fairly estimates the DDM deformation. Because the virtual bones are sparsely distributed in this test, we can see that all mappings (Eq. (11b) and Eq. (11c)) work well. Combining with the tests in the previous sections (§7.2 and §7.3), we can reason that our linearization map also works.

### 7.5 Multi-weights (Δ) Update

In Fig. 15 we show the difference in approximation errors of models generated with and without multi-weights update (§4.3), where we see that our multi-weights update can noticeably improve the quality.

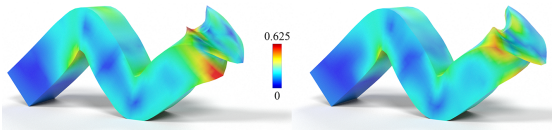


Fig. 15. Left: approximation error on the model generated without performing multi-weights Δ update. Right: result of the same pipeline but with multi-weights update.

### 7.6 Comparison with Helper Joint Rig

In Fig. 16, we show a comparison with a LBS model equipped with helper joints. Note that this comparison is neither fair nor scientifically accurate. The model with helper joints is setup and animated by professional artists with a well-known pipeline. Conversely, DDM model is quite new and there is not much knowledge about the setup. Visually, we can see some correspondences between our virtual bones and helper joints. We also notice that our model tends to put no virtual bone at biceps or triceps. The possible reason is that DDM does not handle muscle bulging, and therefore out model cannot resolve this information.

### 7.7 Comparison with Elasticity-Inspired Skinning

In this section, we compare our DDM-based skinning with the elasticity-inspired skinning method [Kavan and Sorkine 2012]. This technique computes good skinning weights for classical skinning methods such as LBS or DQS by decomposing the rotation at joints into independent components (swing and twist) and computing separate skinning weights for these components. The final deformation combines the best of both worlds where LBS is applied on swing motions and spherical blending is applied on twist motions, thus avoiding both candy-wrapper artifacts (for twisting) and bulging artifacts (for swinging), as shown on the left of Fig. 17.

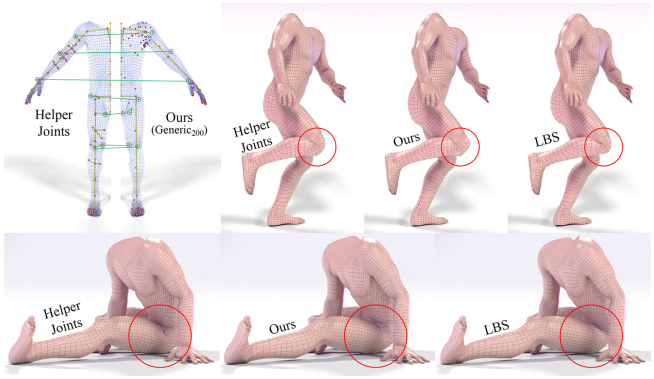


Fig. 16. A subjective comparison between our model and the professional rig with helper joints. We can see some correspondences between our virtual bones and helper joints that add non-linear correction for LBS. Subjectively, our model can produce more features of the professional rig compared to the pure LBS model setup on the master bones.

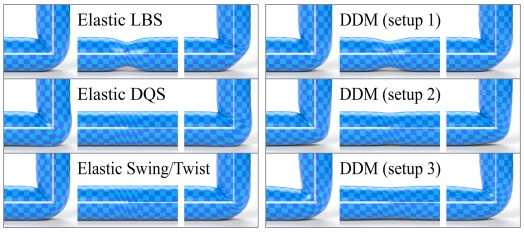


Fig. 17. Left: elasticity-inspired skinning [Kavan and Sorkine 2012] with different setups: directly using computed skinning weights for LBS (Elastic LBS), for DQS (Elastic DQS), or for the combined deformer (Elastic Swing/Twist). Right: our DDM-based skinning with different setups by changing the global rotation and translation smoothness. Each setup shows a group of three poses in order: bending (swinging), twisting, and both bending+twisting.

By combining different levels of smoothness for rotation and translation, DDM and our proposed model can achieve a similar effect, as shown on the right-hand side of Fig. 17. Compared to elasticity-inspired skinning, DDM-based methods provide better customization as rotation smoothness and translation smoothness can be controlled at both global and local levels. However, elasticity-inspired skinning can completely eliminate volume loss on twisting, where DDM-based methods can only reduce it to a certain level.

Another advantage of DDM-based methods is the ability to mix in the "negative bulging" effect [Le and Lewis 2019], which allows DDM and our proposed model to approximate different deformation behavior, including as-rigid-as-possible (ARAP) deformations [Sorkine and Alexa 2007] as shown in Fig. 18.

A limitation of swing/twist deformers [Kavan and Sorkine 2012] is the gimbal lock issue [Baerlocher 2001], which prevents the extraction of unique twist rotation and leads to deformation artifacts (Fig. 19). This issue can be handled by explicitly specifying twist angles as the first component in the joint rotation order. However, this solution can potentially limit practical setups in production. Also, it does not work for models with unorganized bones (Fig. 18).

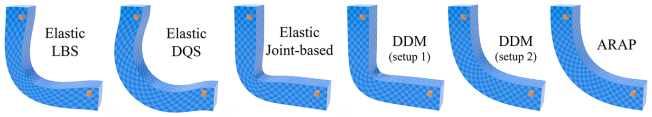


Fig. 18. The deformation space of elasticity-inspired skinning [Kavan and Sorkine 2012] is constrained to a combination of linear blending and spherical blending. Because the middle part of the bar is always assigned the skinning weight of (0.5, 0.5), it protrudes outward from the bend and cannot approximate the behavior of ARAP deformation [Sorkine and Alexa 2007]. In addition, swing/twist motion cannot be extracted from unorganized bones, so Kavan and Sorkine's model [2012] is dominated by the joint-based deformer, which is LBS with joint handle-based bounded biharmonic weights [Jacobson et al. 2011].

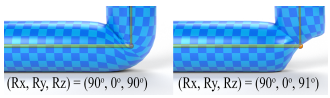


Fig. 19. The gimbal lock issue when decomposing the joint rotation to swing and twist motions causes deformation artifacts for elasticity-inspired skinning near rotations at multiple of 90° where the decomposed swing/twist can be flipped.

## 8 CONCLUSION

In this paper, we have presented a compression method to convert an expensive DDM model into a cheaper skinning model with two layers: a smaller DDM model on top of a large but sparse LBS model. This compression can significantly reduce the storage cost and improve run-time performance.

The compressed model is automatically computed from the DDM multi-weights. We have formulated a novel continuous example-based problem for optimizing model parameters. Our problem formulation is simple and elegant while still offering high-level controls such as using the skeleton hierarchy or controlling the range of motions for sampling. We have also introduced practical techniques to optimize our DDM multi-weights in the first layer and a fix for the floating-point cancellation issue of the original DDM. Our extensive analysis showed that the proposed model offers similar deformation quality to DDM and comparable run-time performance with LBS.

In the future, we would like to target some existing issues and explore new directions, such as using different norm for the objective function to reduce the maximum compression error and mixing continuous examples with traditional discrete examples.

## ACKNOWLEDGMENTS

We thank Seigo Tanaka, Daniel Doluntap, and other members of the UFC team for providing the Generic model and the accompanying animation. We also thank Colin Barré-Brisebois, Jim Royal, Chris Lewin, Henrik Halén, Mathieu Lamarre, Jenna Frisk, and Konrad Tollmar from the SEED team for proofreading the paper. Finally, we thank the anonymous SIGGRAPH reviewers for their constructive comments and suggestions.

## REFERENCES

M. Aharon, M. Elad, and A. Bruckstein. 2006. K -SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Trans. Sig. Proc.* 54, 11 (Nov. 2006), 4311–4322. https://doi.org/10.1109/TSP.2006.881199

Marc Alexa. 2002. Linear Combination of Transformations. *ACM Trans. Graph.* 21, 3 (July 2002), 380–387. https://doi.org/10.1145/566654.566592

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: Shape Completion and Animation of People. *ACM Trans. Graph.* 24, 3 (July 2005), 408–416. https://doi.org/10.1145/1073204.1073207

Paolo Baerlocher. 2001. *Inverse kinematics techniques of the interactive posture control of articulated figures.* Ph.D. Dissertation. École Polytechnique Fédérale de Lausanne. https://doi.org/10.5075/epfl-thesis-2383

Stephen W. Bailey, Dave Otte, Paul Dilorenzo, and James F. O'Brien. 2018. Fast and Deep Deformation Approximations. *ACM Trans. Graph.* 37, 4, Article 119 (July 2018). https://doi.org/10.1145/3197517.3201300

Ake Bjorck. 1996. *Numerical methods for least squares problems.* https://doi.org/10.1137/1.9781611971484.fm

Wei-Wen Feng, Byung-Uck Kim, and Yizhou Yu. 2008. Real-time Data Driven Deformation Using Kernel Canonical Correlation Analysis. *ACM Trans. Graph.* 27, 3, Article 91 (Aug. 2008). https://doi.org/10.1145/1360612.1360690

Lin Gao, Yu-Kun Lai, Dun Liang, Shu-Yu Chen, and Shihong Xia. 2016. Efficient and Flexible Deformation Representation for Data-Driven Surface Modeling. *ACM Trans. Graph.* 35, 5, Article 158 (July 2016). https://doi.org/10.1145/2908736

Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-space Physics. *ACM Trans. Graph.* 31, 4, Article 72 (July 2012). https://doi.org/10.1145/2185520.2185568

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner, and Markus Gross. 2013. Efficient Simulation of Secondary Motion in Rig-space. In *Proc. of ACM/Eurographics Symposium on Computer Animation (SCA 2013).* 165–171. https://doi.org/10.1145/2485895.2485918

Nils Hasler, Thorsten Thormählen, Bodo Rosenhahn, and Hans-Peter Seidel. 2010. Learning Skeletons for Shape and Pose. In *Proc. of ACM Symposium on Interactive 3D Graphics and Games (I3D 2010).* Association for Computing Machinery, New York, NY, USA, 23–30. https://doi.org/10.1145/1730804.1730809

Alexandru-Eugen Ichim, Petr Kadleček, Ladislav Kavan, and Mark Pauly. 2017. Phace: Physics-based Face Modeling and Animation. *ACM Trans. Graph.* 36, 4, Article 153 (July 2017). https://doi.org/10.1145/3072959.3073664

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast Automatic Skinning Transformations. *ACM Trans. Graph.* 31, 4, Article 77 (July 2012). https://doi.org/10.1145/2185520.2185573

Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded Biharmonic Weights for Real-Time Deformation. *ACM Trans. Graph.* 30, 4, Article 78 (July 2011). https://doi.org/10.1145/2010324.1964973

Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. 2014. Skinning: Real-time Shape Deformation. In *ACM SIGGRAPH 2014 Courses.* http://skinning.org/

Alec Jacobson and Olga Sorkine. 2011. Stretchable and Twistable Bones for Skeletal Shape Deformation. *ACM Trans. Graph.* 30, 6 (Dec. 2011). https://doi.org/10.1145/2070781.2024199

Doug L. James and Christopher D. Twigg. 2005. Skinning Mesh Animations. *ACM Trans. Graph.* 24, 3 (July 2005), 399–407. https://doi.org/10.1145/1073204.1073206

Ben Jones, Nils Thuerey, Tamar Shinar, and Adam W. Bargteil. 2016. Example-based Plastic Deformation of Rigid Bodies. *ACM Trans. Graph.* 35, 4, Article 34 (July 2016). https://doi.org/10.1145/2897824.2925979

Petr Kadleček, Alexandru-Eugen Ichim, Tiantian Liu, Jaroslav Křivánek, and Ladislav Kavan. 2016. Reconstructing Personalized Anatomical Models for Physics-based Body Animation. *ACM Trans. Graph.* 35, 6, Article 213 (Nov. 2016). https://doi.org/10.1145/2980179.2982438

Ladislav Kavan, Steven Collins, and Carol O'Sullivan. 2009. Automatic Linearization of Nonlinear Skinning. In *Proc. of ACM Symposium on Interactive 3D Graphics and Games (I3D 2009).* 49–56. https://doi.org/10.1145/1507149.1507157

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2008. Geometric Skinning with Approximate Dual Quaternion Blending. *ACM Trans. Graph.* 27, 4, Article 105 (Nov. 2008). https://doi.org/10.1145/1409625.1409627

L. Kavan, P.-P. Sloan, and C. O'Sullivan. 2010. Fast and Efficient Skinning of Animated Meshes. *Computer Graphics Forum* 29, 2 (2010), 327–336. https://doi.org/10.1111/j.1467-8659.2009.01602.x

Ladislav Kavan and Olga Sorkine. 2012. Elasticity-Inspired Deformers for Character Articulation. *ACM Trans. Graph.* 31, 6, Article 196 (Nov. 2012). https://doi.org/10.1145/2366145.2366215

Ladislav Kavan and Jiří Žára. 2005. Spherical Blend Skinning: A Real-time Deformation of Articulated Models. In *Proc. ACM Symposium on Interactive 3D Graphics and Games (I3D 2005).* 9–16. https://doi.org/10.1145/1053427.1053429

Hyunsoo Kim and Haesun Park. 2008. Nonnegative Matrix Factorization Based on Alternating Nonnegativity Constrained Least Squares and Active Set Method. *SIAM J. Matrix Anal. Appl.* 30, 2 (July 2008), 713–730. https://doi.org/10.1137/07069239X

Paul G. Kry, Doug L. James, and Dinesh K. Pai. 2002. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *Proc. of ACM/Eurographics Symposium on Computer Animation (SCA 2002)*. 153–159. https://doi.org/10.1145/545261.545286

Eric Landreneau and Scott Schaefer. 2010. Poisson-Based Weight Reduction of Animated Meshes. *Computer Graphics Forum* 29, 6 (2010), 1945–1954. https://doi.org/10.1111/j.1467-8659.2010.01661.x

Binh Huy Le and Zhigang Deng. 2012. Smooth Skinning Decomposition with Rigid Bones. *ACM Trans. Graph.* 31, 6, Article 199 (Nov. 2012). https://doi.org/10.1145/2366145.2366218

Binh Huy Le and Zhigang Deng. 2013. Two-layer Sparse Compression of Dense-weight Blend Skinning. *ACM Trans. Graph.* 32, 4, Article 124 (July 2013). https://doi.org/10.1145/2461912.2461949

Binh Huy Le and Zhigang Deng. 2014. Robust and Accurate Skeletal Rigging from Mesh Sequences. *ACM Trans. Graph.* 33, 4, Article 84 (July 2014). https://doi.org/10.1145/2601097.2601161

Binh Huy Le and Jessica K. Hodgins. 2016. Real-time Skeletal Skinning with Optimized Centers of Rotation. *ACM Trans. Graph.* 35, 4, Article 37 (July 2016). https://doi.org/10.1145/2897824.2925959

Binh Huy Le and J P Lewis. 2019. Direct Delta Mush Skinning and Variants. *ACM Trans. Graph.* 38, 4, Article 113 (July 2019). https://doi.org/10.1145/3306346.3322982

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-Negative Matrix Factorization. In *Proc. of International Conference on Neural Information Processing (NIPS 2000)*. 535–541.

Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2009. Comprehensive Biomechanical Modeling and Simulation of the Upper Body. *ACM Trans. Graph.* 28, 4, Article 99 (Sept. 2009). https://doi.org/10.1145/1559755.1559756

J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-driven Deformation. In *Proc. of International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000)*. 165–172. https://doi.org/10.1145/344779.344862

Duo Li, Shinjiro Sueda, Debanga R. Neog, and Dinesh K. Pai. 2013. Thin Skin Elastodynamics. *ACM Trans. Graph.* 32, 4, Article 49 (July 2013). https://doi.org/10.1145/2461912.2462008

Libin Liu, KangKang Yin, Bin Wang, and Baining Guo. 2013. Simulation and Control of Skeleton-driven Soft Body Characters. *ACM Trans. Graph.* 32, 6, Article 215 (Nov. 2013). https://doi.org/10.1145/2508363.2508427

S. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (Sept. 1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

Matthew Loper, Naureen Mahmood, and Michael J. Black. 2014. MoSh: Motion and Shape Capture from Sparse Markers. *ACM Trans. Graph.* 33, 6, Article 220 (Nov. 2014). https://doi.org/10.1145/2661229.2661273

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-person Linear Model. *ACM Trans. Graph.* 34, 6, Article 248 (Oct. 2015). https://doi.org/10.1145/2816795.2818013

N. Magnenat-Thalmann, F. Cordier, Hyewon Seo, and G. Papagianakis. 2004. Modeling of bodies and clothes for virtual environments. In *IEEE International Conference on Cyberworlds (CW 2004)*. 201–208. https://doi.org/10.1109/CW.2004.47

N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. 1988. Joint-dependent Local Deformations for Hand Animation and Object Grasping. In *Proc. on Graphics Interface (GI 1988)*. 26–33. http://dl.acm.org/citation.cfm?id=102313.102317

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2010. Online Learning for Matrix Factorization and Sparse Coding. *Journal of Machine Learning Research* 11 (March 2010), 19–60. http://dl.acm.org/citation.cfm?id=1756006.1756008

Joe Mancewicz, Matt L. Derksen, Hans Rijpkema, and Cyrus A. Wilson. 2014. Delta Mush: Smoothing Deformations While Preserving Detail. In *Proc. of ACM Symposium on Digital Production (DigiPro 2014)*. 7–11. https://doi.org/10.1145/2633374.2633376

Aleka McAdams, Andrew Selle, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011a. *Computing the singular value decomposition of 3x3 matrices with minimal branching and elementary floating point operations.* Technical Report. University of Wisconsin-Madison.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011b. Efficient Elasticity for Character Skinning with Contact and Collisions. *ACM Trans. Graph.* 30, 4, Article 37 (July 2011). https://doi.org/10.1145/2010324.1964932

Alex Mohr and Michael Gleicher. 2003. Building Efficient, Accurate Character Skins from Examples. *ACM Trans. Graph.* 22, 3 (July 2003), 562–568. https://doi.org/10.1145/882262.882308

Tomohiko Mukai. 2015. Building Helper Bone Rigs from Examples. In *Proc. of ACM Symposium on Interactive 3D Graphics and Games (I3D 2015)*. 77–84. https://doi.org/10.1145/2699276.2699278

Tomohiko Mukai and Shigeru Kuriyama. 2016. Efficient Dynamic Skinning with Low-rank Helper Bone Controllers. *ACM Trans. Graph.* 35, 4, Article 36 (July 2016). https://doi.org/10.1145/2897824.2925905

Olivier Rémillard and Paul G. Kry. 2013. Embedded Thin Shells for Wrinkle Simulation. *ACM Trans. Graph.* 32, 4, Article 50 (July 2013). https://doi.org/10.1145/2461912.2462018

Shunsuke Saito, Zi-Ye Zhou, and Ladislav Kavan. 2015. Computational Bodybuilding: Anatomically-based Modeling of Human Bodies. *ACM Trans. Graph.* 34, 4, Article 41 (July 2015). https://doi.org/10.1145/2766957

Thomas Schlömer, Daniel Heck, and Oliver Deussen. 2011. Farthest-Point Optimized Point Sets with Maximized Minimum Distance. In *Proc. of ACM SIGGRAPH Symposium on High Performance Graphics (HPG 2011)*. 135–142. https://doi.org/10.1145/2018323.2018345

Peter-Pike J. Sloan, Charles F. Rose, III, and Michael F. Cohen. 2001. Shape by Example. In *Proc. of ACM Symposium on Interactive 3D Graphics and Games (I3D 2001)*. 135–143. https://doi.org/10.1145/364338.364382

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (March 2018). https://doi.org/10.1145/3180491

Olga Sorkine and Marc Alexa. 2007. As-Rigid-as-Possible Surface Modeling. In *Proc. of Eurographics Symposium on Geometry Processing (SGP 2007)*. 109–116.

Robert W. Sumner and Jovan Popović. 2004. Deformation Transfer for Triangle Meshes. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 399–405. https://doi.org/10.1145/1015706.1015736

Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. 2005. Mesh-based Inverse Kinematics. *ACM Trans. Graph.* 24, 3 (July 2005), 488–495. https://doi.org/10.1145/1073204.1073218

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust Quasistatic Finite Elements and Flesh Simulation. In *Proc. of ACM/Eurographics Symposium on Computer Animation (SCA 2005)*. 181–190. https://doi.org/10.1145/1073368.1073394

Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. 2013. Implicit Skinning: Real-time Skin Deformation with Contact Modeling. *ACM Trans. Graph.* 32, 4, Article 125 (July 2013). https://doi.org/10.1145/2461912.2461960

Rodolphe Vaillant, Gäel Guennebaud, Loïc Barthe, Brian Wyvill, and Marie-Paule Cani. 2014. Robust Iso-surface Tracking for Interactive Character Skinning. *ACM Trans. Graph.* 33, 6, Article 189 (Nov. 2014). https://doi.org/10.1145/2661229.2661264

Robert Y. Wang, Kari Pulli, and Jovan Popović. 2007. Real-time Enveloping with Rotational Regression. *ACM Trans. Graph.* 26, 3, Article 73 (July 2007). https://doi.org/10.1145/1276377.1276468