CGI2017 manuscript No.
(will be inserted by the editor)

Data-driven Subspace Enrichment for Elastic Deformations with Collisions

Duosheng Yu · Takashi Kanai

Abstract We propose an efficient data-driven enrichment approach to adaptively enhance the expressivity of subspaces for elastic deformations with novel collisions. In general, subspace integration method (also known as model reduction) for elastic deformations can greatly increase simulation speed. However, when the deformations are beyond the expressivity of subspaces such as novel external collisions, obvious artifacts will appear. First, we construct a position-based database of subspaces through full-space collided simulations. We then select small sets of basis vectors to enrich existing subspaces for incoming collided deformations. We also demonstrate that cubature can easily be exploited by our subspace database, and we propose a novel post-processing scheme for refining the cubature weights for more accurate and faster deformations. Our method can achieve well approximated full-space deformations when novel collisions occur. From our experiment results, we further show that our method is applicable to large deformations and large-steps in real-time.

Keywords Elastic deformation · Subspace · Collision · Cubature

1 Introduction

The simulation of deformable elastic objects is becoming more and more important for films, computer games, virtual environments and related fields. Among the methods of deformable elastic object simulation, the Finite Element Method (FEM) is widely used, due to its versatility in representing objects and diverse materials. However, since full-space FEM simulation is required to calculate internal forces and force differentials of all 3D volumetric elements, it is too expensive to use FEM to simulate high resolution deformable objects in real-time.

To address this issue, the subspace integration method, also known as model reduction, uses pre-computed basis vectors to project the original high-dimensional system into the low-dimensional space spanned by those bases. For this reason, the simulation of the subspace system only depends on the number of basis vectors which is much smaller than the original full-space system, enabling high simulation acceleration to be achieved. However, the expressivity in subspaces is also considerably smaller than that in full-spaces. Especially when novel collisions occur, simulations in subspaces always cause unrealistic artifacts due to the lack of expressivity. The purpose of our work is to solve this problem.

Previous approaches which attempt to solve this problem mainly fall into two categories. One combines full-spaces and subspaces into one object, while the other attempts to add more basis vectors to existing subspaces to capture the collided deformation which is going to occur. The former approach can achieve more accurate results but the benefits of subspaces are lost, since the collided deformation in this approach is actually simulated in full-spaces. Our basic idea is based on the latter approach. State-of-the-art approaches however use analytical solutions of displacement as additional basis vectors which are limited to small deformations and small time-steps.

In this paper, we propose a novel method which can overcome these limitations and is efficient for interactive applications (see Figure 1). Our idea here is to enrich the expressivity of the subspace for collided deformation using a subspace database. This idea is based on the observation that when the same rigid object collides near a specific position

Duosheng Yu
The University of Tokyo, 3-8-1 Komaba, Meguro-ku, Tokyo 153-8902, Japan.

Takashi Kanai
The University of Tokyo, 3-8-1 Komaba, Meguro-ku, Tokyo 153-8902, Japan.

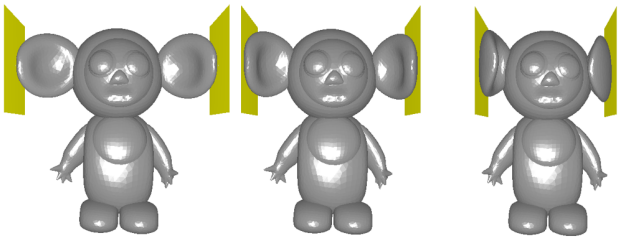


Fig. 1: Two planes push the ears of cheb. Our subspace simulation fully running at 64 fps, and 310× speed up compared with full-space simulation can be achieved.

on an elastic object, under the situation that a rigid object moves straight at a constant speed, the resulting deformations are very similar. If we have the bases that span the space of incoming collisions, we can simulate this deformation in subspaces that well approximate those in full-spaces. For collisions that occur not exactly where we train our database, we use an interpolation scheme to generate new basis vectors. For the purpose of real-time simulation, our method also exploits cubature - a fast evaluation method of subspace internal forces and their differentials. However, unstable artifacts occur with the original cubature approach. We also propose a cubature weights refinement method to solve this problem.

2 Related Work

Subspace integration methods were first introduced to deformable object simulation by Pentland and Willams [14], but were applied only for linear materials. James et al. [9] used precomputed bases of modal analysis to efficiently simulate dynamic deformation of muscles in character animation. Hauser et al. [8] showed that subspace modal framework can easily be coupled with external constraints such as manipulation, collision, etc. Barbič and James [2] presented modal derivatives for fast subspace integration of reduced nonlinear St. Venant-Kirchhoff material. Subspace integration methods are proven to be effective for many graphics applications, such as shape interpolation [21], skinning character dynamics [22], animation editing [3, 13] and fluid simulation [19].

A variety of approaches have been proposed to address the expressivity limitation issue of subspace. Barbič et al. [4] and Kim et al. [12] used multi-domain technologies. The basis vectors usually have global support, so the object is partitioned into several connected domains and subspace simulation is performed for each sub-domain to localize the influence of basis vectors. Harmon et al. [7] used analytic solution to calculate additional bases restricted to small deformations and small time-steps. In contrast, our additional bases are computed using full-space simulation data, which are independent of the scale of deformations. Kim et al. [11] proposed error estimation for subspace simulation, detecting when a subspace is capable of performing the next time-step and switching to full-space simulation when the expressivity of subspace is not enough. Teng et al. [18] combines subspace and full-space at the same time. In contrast, our method can fully run in a subspace, which results in consistent run-time speed and achieves faster frame rates.

For data-driven subspace integration methods, as far as we know, the only work is by Hahn et al. [6], who constructs a subspace database for cloth simulation. However, their approach cannot be used for collided elastic object simulation directly. They perform full-space simulation for character animations and construct a pose-based subspace database of cloth motions. Since character's motions are limited, only a small amount of bases are needed in their situation. In contrast, our position-based database contains a large amount of bases created by doing a lot of full-space collided deformations. Moreover, their approach needs to evaluate full-space internal forces and stiffness matrices, avoided by using cubature.

Cubature proposed by An et al. [1] is a method that fastly evaluates both reduced forces and reduced stiffnesses based on subspace. Cubature can reduce the cost of evaluating subspace forces from $O(\gamma^4)$ to $O(\gamma^2)$, where γ is the dimension of subspaces. To compute such cubature elements, there is a need to solve the best subset selection optimization problem. An et al. [1] use a greedy scheme that builds a set of cubature elements incrementally. Von Tycowicz et al. [20] substitute greedy scheme by a Non-Negativity-constrained Hard Thresholding Pursuit (NN-HTP) algorithm to achieve faster converge and better accuracy. Besides elastic object simulations, cubature has successfully been applied to other applications such as fluid simulation [10].

3 Subspace integration method and Enrichment

We use tetrahedral meshes for FEM deformable object simulation. Given a tetrahedral mesh with N vertices, the deformation $\boldsymbol{u} \in \mathbb{R}^{3N}$ is the displacement of vertices away from the rest configuration X of mesh in world coordinates. The quasistatic equation that governs the motion of a deformable object can be written as:

$$\boldsymbol{K}(\boldsymbol{u})\Delta\boldsymbol{u} = \boldsymbol{f}_{int}(\boldsymbol{u}) + \boldsymbol{f}_{ext}, \tag{1}$$

where $\boldsymbol{K} \in \mathbb{R}^{3N\times 3N}$ is the material stiffness matrix of the object, $\boldsymbol{f}_{int}, \boldsymbol{f}_{ext} \in \mathbb{R}^{3N}$ is the internal force and external force of this object respectively. Δ also means the increment operator.

Subspace integration method is used to accelerate the solution of Equation (1). The original high-dimensional system is projected to a customized low-dimensional system

by a basis matrix $\boldsymbol{U} \in \mathbb{R}^{3N\times\gamma}$ as shown by the following equations, where γ ($\gamma \ll 3N$) is the dimension of this subspace:

$$\begin{aligned}\bar{\boldsymbol{K}}(\boldsymbol{u}) &= \boldsymbol{U}^T\boldsymbol{K}(\boldsymbol{u})\boldsymbol{U},\\ \bar{\boldsymbol{f}}_{int}(\boldsymbol{u}) &= \boldsymbol{U}^T\boldsymbol{f}_{int}(\boldsymbol{u}),\\ \bar{\boldsymbol{f}}_{ext} &= \boldsymbol{U}^T\boldsymbol{f}_{ext}.\end{aligned} \tag{2}$$

The original $3N$ high-dimensional equation then becomes γ low-dimensional equation:

$$\bar{\boldsymbol{K}}(\boldsymbol{u})\Delta\boldsymbol{q} = \bar{\boldsymbol{f}}_{int}(\boldsymbol{u}) + \bar{\boldsymbol{f}}_{ext}, \tag{3}$$

where $\Delta\boldsymbol{q}$ is the reduced increment of displacement. After solving $\Delta\boldsymbol{q}$ in the subspace, $\Delta\boldsymbol{u}$ is calculated by projecting $\Delta\boldsymbol{q}$ back to the full-space:

$$\Delta\boldsymbol{u} = \boldsymbol{U}\Delta\boldsymbol{q}. \tag{4}$$

Subspace basis matrix $\boldsymbol{U}$ can be computed by a variety of standard methods, such as modal analysis [15], modal derivatives [2], modal extension [20], etc. Subspaces computed by those methods have good global support.

The affine space of the subspace matrix $\boldsymbol{U}$ can be considered to be the linear combination of basis vectors in $\boldsymbol{U}$ attached to the undeformed configuration of an object. Adding more basis vectors to $\boldsymbol{U}$ means adding more combinations of bases, and this can make the affine space express a more variety of deformations, and then any proper subset of precomputed basis vectors can be chosen for run-time simulation.

The Enriched subspace is defined as [7],

$$\boldsymbol{U} = [\boldsymbol{G}\ \boldsymbol{L}] \in \mathbb{R}^{3N\times(\gamma+s)},$$

where $\boldsymbol{G}$ is the original γ global basis matrix created by the standard method to support global deformation. By contrast, $\boldsymbol{L}$ is an additional local basis matrix and contains s bases for enriching the affine space of our run-time subspaces. The projection of subspaces then becomes,

$$\bar{\boldsymbol{K}}(\boldsymbol{u}) = \boldsymbol{U}^T\boldsymbol{K}(\boldsymbol{u})\boldsymbol{U} = \begin{bmatrix}\boldsymbol{G}^T\boldsymbol{K}\boldsymbol{G} & \boldsymbol{G}^T\boldsymbol{K}\boldsymbol{L}\\ \boldsymbol{L}^T\boldsymbol{K}\boldsymbol{G} & \boldsymbol{L}^T\boldsymbol{K}\boldsymbol{L}\end{bmatrix}, \tag{5}$$

$$\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}) = \boldsymbol{U}^T\boldsymbol{f}_{int}(\boldsymbol{u}) = \begin{bmatrix}\boldsymbol{G}^T\boldsymbol{f}_{int}(\boldsymbol{u})\\ \boldsymbol{L}^T\boldsymbol{f}_{int}(\boldsymbol{u})\end{bmatrix}. \tag{6}$$

Other variants can be projected by the same way.

4 Data-driven Subspace Enrichment

4.1 Our collision setting and basic idea

By constructing $\boldsymbol{L}$ described in Section 3, certain deformations of collision for an object can be retrieved appropriately. However, it is quite difficult to construct $\boldsymbol{L}$ which can be used for any type of collided deformations.

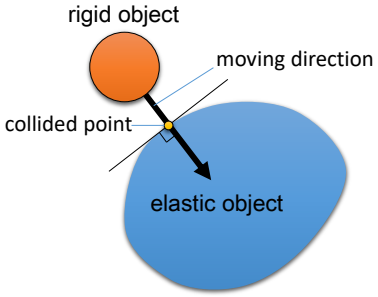


Fig. 2: Our method for setting a colliding rigid object to an elastic object.

We then focus on a simple collided deformation to construct $\boldsymbol{L}$. That is, we consider the situation where a rigid object collides to an elastic object as illustrated in Figure 2. A rigid object is moved along the inverse direction of the normal vector of a point on the surface of an elastic object at a constant speed and is collided at such a point. We call this point *collided point*.

In this situation, the results of our experiments revealed that two $\boldsymbol{L}$s, which are calculated from collided deformations at two very close collided points, have similar basis vectors. Conversely speaking, by using $\boldsymbol{L}$ constructed from a collided deformation at a certain collided point, motion at a point close to that point can be approximated.

This fact contributed to our idea as follows: We calculate a lot of collided deformations at different collided points on the surface of an elastic object and store $\boldsymbol{L}$s in a database. Then we can approximate a collided deformation at any collided point by using a new local subspace interpolated from $\boldsymbol{L}$s at several nearby collided points.

4.2 Overview of our method

Figure 3 shows the visual pipeline of our method. Here we briefly describe each stage of our system.

Pre-computation. The input of our system is a tetrahedra mesh of a deformable object the user wants to simulate and a rigid object that creates collisions. Furthermore, we assume that some regions where typical collisions may occur during run-time simulation are provided. We refer to those regions as *predict-region*.

In the pre-computation, we first compute our global subspace $\boldsymbol{G}$ using modal analysis. We also sample a set of vertices on the predict-region as collided points for the training stage. At each collided point, we then perform full-space simulation with colliding a rigid object to deformable object along the inverse direction of the normal of such a point, as described in the previous subsection. The resulting deformation is thus associated with the corresponding point.

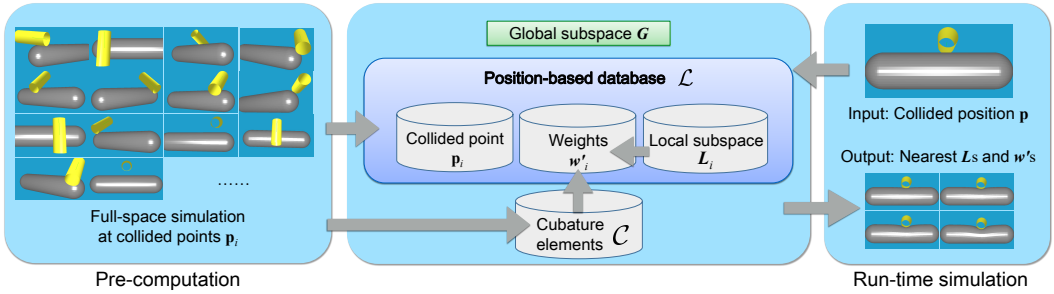


Fig. 3: Overview of our system.

For each motion, we compute a local subspace $\boldsymbol{L}$ and cubature weights $\boldsymbol{w}'$, and store them to a database. We will describe the database construction method in Section 4.3 and cubature weights refinement in Section 4.4.

Run-time simulation. In run-time simulation, collision detection is performed for each frame. If collision does not occur, we simulate an object without using local subspaces from the database, but with the global subspace only. If collision occurs, we specify a hit point on the surface of an elastic object, and search nearby collided points and their associated $\boldsymbol{L}$s and $\boldsymbol{w}'$s. We then construct $\boldsymbol{L}$ and $\boldsymbol{w}$ respectively by interpolation and use them for retrieving collided deformation. We will describe the detail of the run-time process in Section 4.5.

4.3 Position-based database construction

Subspace database is responsible for providing low-dimensional local subspace $\boldsymbol{L}$ for real-time simulation that can well capture the incoming collided deformations of the elastic object. We first have to decide how to train our full-space collided deformation data and how to process these data to extract the information we need. We also have to find a way to associate data with the collided deformations. At the initial state, we only have undeformed configuration of the object. Our database should be adaptive for regions where the user predicts collisions will occur at run-time on an elastic object.

Here we construct a database named as *position-based database* $\mathcal{L} = \{(\mathbf{p}_i, \boldsymbol{L}_i, \boldsymbol{w}'_i)\}(i = 1 \ldots N_p)$ where N_p is the number of collided elements. Each element is a triplet of parameters, where $\mathbf{p}_i$ is the collided point on the surface of an elastic object, $\boldsymbol{L}_i$ is the local subspace of collided deformation at $\mathbf{p}_i$, and $\boldsymbol{w}'_i$ is a set of cubature weights for this deformation. Namely, $\boldsymbol{L}_i$ and $\boldsymbol{w}'_i$ are associated with $\mathbf{p}_i$. So in run-time simulation, $\mathbf{p}_i$ is used as a key to search $\boldsymbol{L}_i$ and $\boldsymbol{w}'_i$. That is, when we perform collided simulation at a hit point $\mathbf{p}$, we find nearest points $\mathbf{p}_i$ from the database and obtain the corresponding $\boldsymbol{L}_i$ and $\boldsymbol{w}'_i$.

We first sample points uniformly distributed in the predict-region where collisions will occur. To do this, we can use the geometry analysis tools such as Monte Carlo sampling and Poisson-disk sampling. These sampled vertices are used as collided points $\mathbf{p}_i$. The number of sample points is independent from the resolutions of a tetrahedral mesh and is determined by a geometric parameter (e.g. a Poisson disk radius). Note that the predict-region can be specified by the user and does not necessary have to be the entire surface of an elastic object. Figure 6 shows the resulting vertices of Poisson-disk sampling on a capsule-shaped object. We then perform full-space simulation by colliding a rigid object to the elastic object at each collided point.

After the training stage, use of the deformations of all frames as our local subspaces will result in a massive amount of data that is impractical to reuse. We thus prefer to use a small amount of data that well captures the deformation. On the other hand, constructing small bases from all training data by PCA or other data processing algorithms will eliminate the locality of training data. Our choice here is to compute local bases for each training data of collided deformation at a collided point. Thus our seperately computed bases are distributed across our collided points.

Our method aims to capture several keyframes for a motion. We extract m keyframes at equal intervals from a collided deformation (see Figure 4). A set of keyframes and local subspace of a motion i are then defined as $\mathcal{X}_i = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m\}$ and $\boldsymbol{L}_i$ respectively. The number in subscripts of $\boldsymbol{u}$ indicates a chronological order of this deformation.

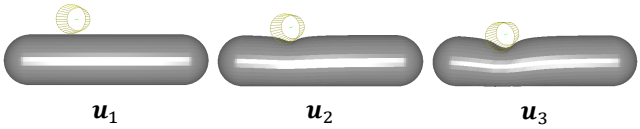


Fig. 4: Three key-frames of a collided deformation.

For the computation of bases, we use the method in [21]. We first calculate the affine space which linearly spans this deformation using the displacement between keyframes.

$$\boldsymbol{u}_k - \boldsymbol{u}_{k-1} \quad \text{subject to} \quad k \in \{2, 3, \ldots, m\}. \tag{7}$$

These displacements represent the deformation of this motion. We add all resulting $m-1$ vectors to a local subspace $\boldsymbol{L}_i$.

We also compute the derivatives of all these key-frames. The derivative of a configuration with respect to another configuration is a vector in $\mathbb{R}^{3N}$. As discussed in [21], all derivative vectors of a keyframe configuration linearly span a $m-1$ subspace that best approximates other keyframe configurations. We can regard this affine space as the tangent space at keyframe configuration $\boldsymbol{u}_i$.

The derivatives are calculated by solving the following equation:

$$\boldsymbol{K}(\boldsymbol{u}_i)\boldsymbol{D}_j(\boldsymbol{u}_i) = \boldsymbol{F}_j(\boldsymbol{u}_i), \tag{8}$$

where $\boldsymbol{K}(\boldsymbol{u}_i)$ is the tangent stiffness matrix in configuration $\boldsymbol{u}_i$, $\boldsymbol{F}_j(\boldsymbol{u}_i)$ is the internal force deforming $\boldsymbol{u}_i$ into $\boldsymbol{u}_j$, and $\boldsymbol{D}_j(\boldsymbol{u}_i)$ is the resulting derivatives. We do this for all key-frames, so the resulting tangent subspace has $m \times m - m$ vectors. We finally add all these vectors to $\boldsymbol{L}_i$. Totally, L_i will contain $m^2 - 1$ vectors.

Algorithm 1 Database Construction.

Input: Tetrahedra mesh, rigid object and predict-region
Output: Position-based Database $\mathcal{L}$

procedure DATABASE CONSTRUCTION
  Sample collided points in predict-region;
  $\mathcal{L} = \{\}$;
  for each collided point $\mathbf{p}_i$ **do**
    Do full-space collided simulation at $\mathbf{p}_i$;
    Capture m key-frames $\mathcal{X}_i = \{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_m\}$;
    $\boldsymbol{L}_i = \{\}$;
    for $k = 2, 3, ..., m$ **do**
      $\boldsymbol{L}_i \leftarrow \boldsymbol{L}_i \cup \{\boldsymbol{u}_k - \boldsymbol{u}_{k-1}\}$;
    end for
    for $j, k = 1, 2, 3, ..., m; j \neq k$ **do**
      Solve $\boldsymbol{K}(\boldsymbol{u}_k)\boldsymbol{D}_j = \boldsymbol{F}_j(\boldsymbol{u}_k)$;
      $\boldsymbol{L}_i \leftarrow \boldsymbol{L}_i \cup \{\boldsymbol{D}_j\}$;
    end for
  end for
  Compute cubature elements C using all key-frames $\{\mathcal{X}_1, \mathcal{X}_2, ...\}$;
  for each collided point $\mathbf{p}_i$ **do**
    $\boldsymbol{w}'_i \leftarrow$ WEIGHTS REFINEMENT$(C, \mathcal{X}_i, \boldsymbol{L}_i)$;
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathbf{p}_i, \boldsymbol{L}_i, \boldsymbol{w}'_i)\}$;
  end for
end procedure

The construction of the database is listed in Algorithm 1. After we collect the difference vectors and the derivative vectors of a deformation to $\boldsymbol{L}_i$, a truncated PCA over each local subspace $\boldsymbol{L}_i$ to orthogonalize this local basis is optionally used, then the size of database can further be reduced.

4.4 Cubature weights Refinement

In this subsection, we first describe the problem which occurs when applying the original cubature to our subspace database and then describe how we solve this problem to make the cubature suitable to our subspace database.

Cubature is a method that uses a small number of key elements to approximately evaluate both the reduced internal forces $\bar{\boldsymbol{f}}_{int}(\boldsymbol{u})$ and reduced stiffness matrix $\bar{\boldsymbol{K}}(\boldsymbol{u})$. We denote C, $\boldsymbol{w}$ as a set of original cubature elements and their weights respectively, then

$$\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}) \approx \sum_{i \in C} w_i \boldsymbol{U}_i \tilde{\boldsymbol{f}}^i_{int}(\boldsymbol{u}), \tag{9}$$

$$\bar{\boldsymbol{K}}(\boldsymbol{u}) \approx \sum_{i \in C} w_i \boldsymbol{U}_i^T \boldsymbol{K}_i \boldsymbol{U}_i, \tag{10}$$

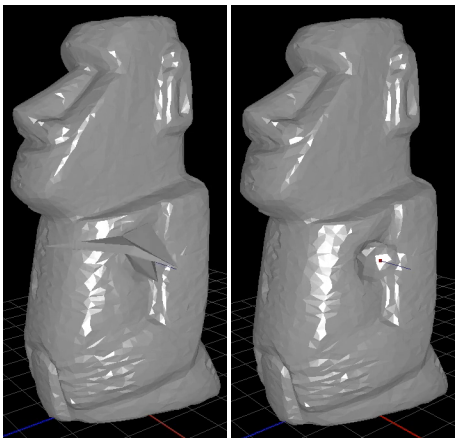
where w_i is the weight of cubature element i, and $\boldsymbol{U}_i, \tilde{\boldsymbol{f}}^i_{int}, \boldsymbol{K}_i$ are the bases, internal forces and stiffness matrix of cubature element i respectively.

We use Non-Negativity-constrained Hard Thresholding Pursuit (NN-HTP) algorithm [20] to determine cubature weights. In the training stage, we have trained a lot of collided deformations, thus all key-frames of all deformations $\{\mathcal{X}_1, \mathcal{X}_2, ...\}$ which we used to compute basis vectors are naturally chosen as the samples for the training of original cubature.

However, we found that the original cubature makes unstable simulations as shown in the left of Figure 5. In the run-time simulation, we hope that our cubature weights can well capture the incoming collided deformation. Since we train cubature weights using deformations at all of collided points. The subspace internal force of element i in NN-HTP is calculated by $\tilde{\boldsymbol{f}}^i_{int}(\boldsymbol{u}) = \boldsymbol{U}^i \boldsymbol{f}^i_{int}(\boldsymbol{u})$ with $\boldsymbol{U} = [\boldsymbol{G}]$, but $\boldsymbol{U} = [\boldsymbol{G}\ \boldsymbol{L}]$ is used in our run-time simulation. The resulting cubature loses the locality of each deformation. Thus the resulting cubature will contain a lot of unrelated but non-zero weights, yielding large error for the evaluation of reduced internal forces.

Our solution here is to refine weights based on original cubature elements. We have to find unrelated elements, and set their weights to zero (elements with zero weight values do not need to be evaluated), and change other non-zero weights to make them more suitable for a specific collided deformation. To this end, our refined weights are more suitable for specific deformations and can achieve faster frame rates as shown in the right side of Figure 5. Our ingredients for refining weights are the m key-frames components $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_m\}$ of a collided deformation, global subspace $\boldsymbol{G}$ and corresponding local subspace $\boldsymbol{L}$. The internal force for a cubature element is calculated by

$$\tilde{\boldsymbol{f}}^i_{int}(\boldsymbol{u}) = [\boldsymbol{G}\ \boldsymbol{L}]^i \boldsymbol{f}^i_{int}(\boldsymbol{u}), \tag{11}$$



Original. Refined.

Fig. 5: Our refined cubature weights solve unstable artifact and achieve faster frame rate at 195 fps than the original at 39 fps.

where $[\boldsymbol{G}\ \boldsymbol{L}]^i$ is the corresponding part of cubature element i in subspace $[\boldsymbol{G}\ \boldsymbol{L}]$. This makes the reduced internal forces more suitable for this local collided deformation. We assume that C contains o elements. A new set of weights $\boldsymbol{w}'$ is then computed by solving a Non-Negative Least-Squares (NNLS) problem:

$$\boldsymbol{A}\boldsymbol{w}' = \boldsymbol{b} \quad \text{subject to} \quad \boldsymbol{w}' \geq 0, \tag{12}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \frac{\tilde{\boldsymbol{f}}^1_{int}(\boldsymbol{u}_1)}{||\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_1)||} & \cdots & \frac{\tilde{\boldsymbol{f}}^o_{int}(\boldsymbol{u}_1)}{||\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_1)||} \\ \vdots & \ddots & \vdots \\ \frac{\tilde{\boldsymbol{f}}^1_{int}(\boldsymbol{u}_m)}{||\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_m)||} & \cdots & \frac{\tilde{\boldsymbol{f}}^o_{int}(\boldsymbol{u}_m)}{||\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_m)||} \end{bmatrix}_{(\gamma+s)m\times o},$$

$$\boldsymbol{b} = \begin{bmatrix} \frac{\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_1)}{||\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_1)||} \\ \vdots \\ \frac{\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_m)}{||\bar{\boldsymbol{f}}_{int}(\boldsymbol{u}_m)||} \end{bmatrix}_{(\gamma+s)m}.$$

We refine weights for all collided deformations using Algorithm 2. The output is a set of cubature weights $\{\boldsymbol{w}'_i\}$ associated with each collided point.

4.5 Run-time process

Once a position-based database is constructed, the remaining question is how to select proper local subspaces at run-time simulation. This subspace should be low-dimensional and well capture the incoming collided deformation.

One possible solution is to compare the current state with the vectors in the database using evaluation metrics such as L2 norm, then gather a constant number of vectors that best approximate the current state to construct our run-time subspace. However in our experiment, we found that this solution is not practical. Since our database contains a large number of basis vectors, updating the subspace in each time step will take too much time. Also, updating a subspace with a regular interval frame will result in a discontinuous subspace which makes the simulation unstable.

Algorithm 2 Cubature weights refinement for a deformation.

Input: A set of key-frames $\mathcal{X}_i$, local subspace $\boldsymbol{L}_i$ and original cubature elements C.
Output: Refined weights $\boldsymbol{w}'_i$

```
procedure WEIGHTS REFINEMENT(C, X_i, L_i)
    for each key-frame in a set do
        Compute f̃^i_int(u) of cubature elements in C using Equation (11);
        Compute f̄_int(u) using Equation (2) with U = [G L_i];
    end for
    Assemble matrix A and vector b;
    Solve A w'_i = b for refined weights w'_i;
    return w'_i;
end procedure
```

In the case of the subspace cloth simulation proposed in [6], the basis vectors in the database is aligned to the full-space internal forces of the current configuration of cloth and the projected length is scaled in the inverse distance. Several bases with the longest length are selected. However with their method, full-space internal forces of an object have to be evaluated, thus it does not fit our situation. We want to simulate the collided deformations as fast as possible. Moreover, we use the cubature scheme, and only evaluate subspace internal forces in our run-time simulation.

For these reasons, our current solution chooses to select several whole subspaces $\boldsymbol{L}_i$ according to the position where collision occurs in run-time simulation. In the training stage, we associated each local subspace with a collided point. When collision occurs, we search collided points from the position where collision occurs. Next, we select the m closest collided points and the corresponding $\boldsymbol{L}_i$ for our run-time subspace.

We then interpolate these subspaces to create a run-time local subspace:

$$\boldsymbol{L} = \sum_{i=1}^{m} \zeta(d)\boldsymbol{L}_i, \tag{13}$$

where $\zeta(d)$ is a weight function taking the distance between collision position and selected collided points as the input, and a Gaussian kernel function or any other types of kernel functions can be used for such a weight function.

The selected subspaces of collided points are very close to each other and are constructed in the same space. For the construction of a local subspace, the same algorithm (Algorithm 1) is used which makes the same sorting order of basis vectors in local subspaces, i.e. each basis vector in

a local subspace corresponds to that in other local subspace. So no post-processing is required before we interpolate them.

In run-time simulation, the weights are also selected in the same way as local subspaces and we use the same interpolation scheme to interpolate the weights as local subspace does:

$$\boldsymbol{w} = \sum_{i=1}^{m} \zeta(d)\boldsymbol{w}_i'. \tag{14}$$

5 Results and Discussion

All our simulations run on a desktop PC with Intel® Core™ i7-4790 3.60GHz CPU and 32GB RAM. We use the co-rotational material introduced in [16] and implicit Euler time discretization for all our experiments. In full-space simulation, we used a conjugate gradient method to solve a large sparse linear equation, and in subspace simulation LU-decomposition is used to solve a small dense linear equation. Poisson-disk sampling by [5] with different radii are used to create sampling vertices (based on the average length r of edges of objects) as collided points. The resulting vertices are tightly packed, but not closer to each other than Poisson disk radius. In the training stage, we perform full simulation with 60 frames for all collided points and capture three key-frames for computing local subspaces. For cubature training, we use NN-HTP algorithm [20] with a fitting error value being 0.05. For the run-time simulation, we use the method in [17] for collision detection and select four local subspaces using the k-nearest neighbors algorithm.

Capsule. We first test our method on a capsule-shape mesh with 5,312 vertices and 18,604 tetrahedra. We manually specify the top region of the capsule as the predict-region. We use another cylinder as a rigid object to create database at collided points as shown in Figure 6.

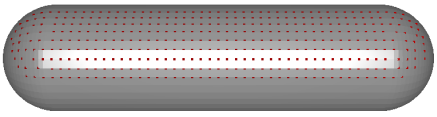


Fig. 6: Sampling vertices (collided points) for capsule created by Poisson-disk sampling with a constraint value being r.

Figure 7 shows the comparison results of full-space simulation with subspace simulations of previous methods and ours. In these experiments, we randomly chose three collided points on the predict-region. It can clearly seen that our method can well approximate the full-space simulation compared with other methods. In contrast, subspace integration method with $\boldsymbol{G}$ only cannot approximate the full-spae simulation since using only global subspace $\boldsymbol{G}$ cancels the local feature of collided deformation. In the result by [7], the corresponding local feature is well represented, however, several unrelated places are also deformed together.

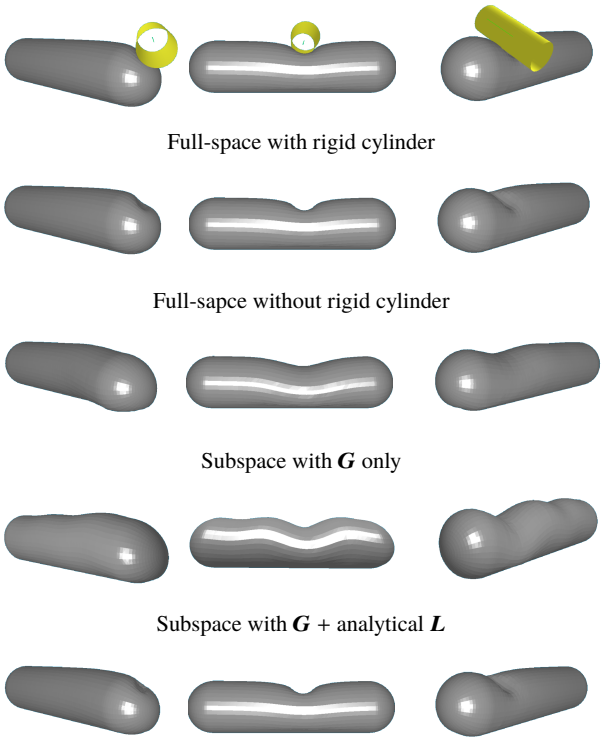


Fig. 7: Comparisons of full simulation with previous methods and ours. From top to bottom rows: full-space simulation with rigid cylinder (in yellow), full-space simulation without rigid cylinder, subspace simulation with $\boldsymbol{G}$ only, subspace simulation with $\boldsymbol{G}$+ analytical $\boldsymbol{L}$ in [7], and subspace simulation with $\boldsymbol{G} + \boldsymbol{L}$ from database (ours).

Moai. We also test our method on a moai-shape mesh with 12,354 vertices and 45,102 tetrahedra. We manually specify the middle region of the moai as the predict-region. To evaluate the relationship between the size of database (determined by the Poisson-disk radius r) and the approximation errors, we have created three databases according to different Poisson-disk radii. Figure 8 shows the display results of collided points with different Poisson-disk radii $r, 2r, 3r$, where r is the average length of edges in a mesh. We name such databases with different Poisson-disk radii as database r (DB r), database $2r$ (DB $2r$), and database $3r$ (DB $3r$) hereafter.

To evaluate our results, here we use L2 error between displacements of full-space simulation results and those of

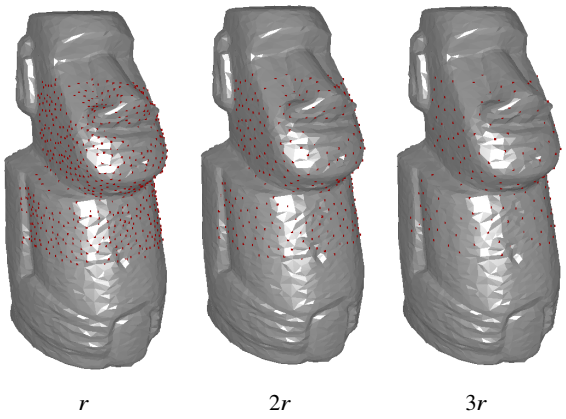


Fig. 8: Collided points (in red) with different Poisson-disk radii $r, 2r, 3r$, where r is the average length of edges in a moai mesh.

obj	pos	DB	ε_{20}	ε_{30}	ε_{40}	ε_{50}
moai	1	r	0.103	0.463	0.796	1.046
		$2r$	0.087	0.528	1.025	1.433
		$3r$	0.3	0.558	0.970	1.326
	2	r	0.28	0.587	0.826	0.98
		$2r$	0.375	0.578	0.780	1.059
		$3r$	0.469	0.865	1.294	1.87

Table 1: L2 errors of collided deformations at two hitted points for 20^{th} frame (ε_{20}), 30^{th} frame (ε_{30}), 40^{th} frame (ε_{40}) and 50^{th} frame (ε_{50}) with different databases.

subspace simulation results measured in $\mathbb{R}^{3N}$,

$$\varepsilon_i = ||\boldsymbol{u}^i_{full} - \boldsymbol{u}^i_{sub}||_2, \tag{15}$$

where $\boldsymbol{u}^i_{full}$ is the displacement of the i^{th} frame in full-space simulation, and $\boldsymbol{u}^i_{sub}$ is the corresponding displacements in the subspace.

We randomly pick up two hit points for the run-time simulations. The results of run-time collided deformations with such two hit points are shown in Figure 9, and the L2 errors for several frames are shown in Table 1. It can be seen that better results are achieved with database r than others. Denser collided points make the collided point in the run-time simulation closer to the collided points in the database and at a higher probability. Therefore, a database with more dense collided points yields good approximation to the full-space simulation result. It can also be seen that the error increases as the frame number increases. This is because errors are accumulated as the simulation progresses.

Cheb. We construct two orthogonal local subspaces $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ for a cheb object which contains 11,989 vertices and 43,813 tetrahedra. They are created from full-space collided deformations by using a plane rigid object to push cheb's left and right ears respectively. In the run-time simulation, we simulate a scene of two planes pushing the left and right ears at the same time using the subspace $\boldsymbol{U} = [\boldsymbol{G}\ \boldsymbol{L}_1\ \boldsymbol{L}_2]$ as shown in Figure 1. The result shows that multiple collisions are also possible with our method in cases where two collided deformations do not affect each other.

Statistical results. We list the statistical results for pre-computation and performance for run-time simulation in Table 2. At the run-time simulation, we measure the average frame rate for all frames. Though more denser collided points result inachieve bettermore approximated results, the size of database is also larger and pre-computation will take more time.

Our weight refinement method for cubature elements provides good approximation to full-space simulation. As shown in Figure 10, the fitting errors of our weights are smaller than the original weights. Furthermore, since the original cubature takes all keyframes of local deformations as input, the resulting cubature elements take all local deformations into account, i.e. contain unrelated elements for each local deformation. Our refinement can set the weights of such unrelated elements to zero. As a result, our refined weights contain $81\% - 97\%$ zeros which do not need to be calculated, and our method thus becomes faster than using the original weights. Consequently, our method is $99\times$ faster than full-space simulation with a capsule database and $98 - 131\times$ faster with moai databases.

Limitations. We proposed an efficient database method that resolves the problem of artifacts when deformation exceeds the expressivity of subspaces. However, there are several limitations in our method. First, like other data-driven methods, our method cannot capture the local deformation outside the region as we have not trained it for local deformation. Secondly, our database is most suitable for the shape of rigid object used for creating this database. The other rigid objects with different shape may result undesired deformations. Thirdly, the size of our database becomes very large because there are too many collided points or the resolution of a mesh is very large. Finally, our method assumes simple situations where a straightly-moving rigid object collides into an elastic object when constructing a subspace database. Therefore, the real-time simulation may produce unexpected deformations when situations are outside such assumptions, i.e. the direction of collision of rigid bodies changes during simulation, etc.

6 Conclusion and Future Work

In this paper, we propose a scheme that achieves well-approximated local deformations of collided deformable simulation using a carefully-constructed subspace database to enhance the expressivity of existing subspaces. Our

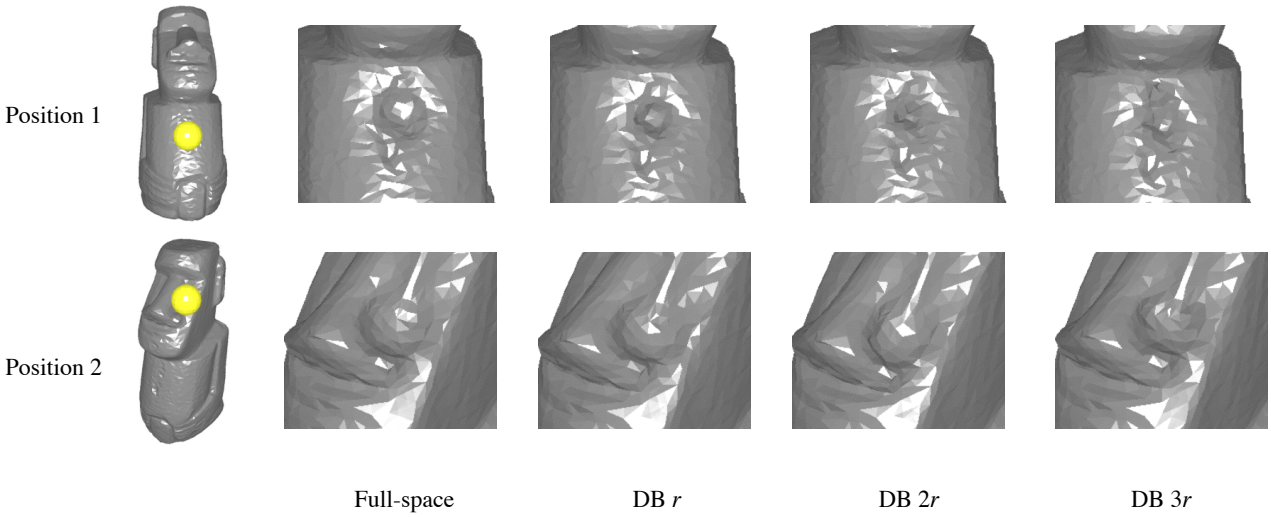


Fig. 9: Comparison of results with different dense collided points.

obj	d-ctr	s-vtx	tn-t	b-t	DB-s	cub-t	cubs	rw-t	rt-f	rt-w	rt-rw	sp-w	sp-rw
capsule	r	417	11302s	1147s	361MB	111s	395	85	3.5	135	348	38×	99×
moai	r	726	38994s	4660s	1055MB	2731s	2712	298	1.7	34	167	20×	98×
	$2r$	354	23461s	2126s	515MB	3180s	2975	204	1.4	31	184	22×	131×
	$3r$	193	10579s	1169s	280MB	534s	2235	111	1.7	40	190	23×	111×

Table 2: **Simulation statistics of our database.** From left to right: Poisson-disk radius (d-ctr), the number of collided points (s-vtx), time for training collided deformations (tn-t), time for basis computation (b-t), database size (DB-s), time for cubature training (cub-t), resulting number of cubature elements (cubs), time for refining cubature weights (rw-t), average frame rate (fps) of full-space (rt-f), average frame rate of our subspace without refined cubature weights (rt-w), average frame rate of our subspace with refined cubature weights (rt-rw), speed up rate without refined weights (sp-w) over full-space simulation, speed up rate with refined cubature weights (sp-rw) over full-space simulaiton.

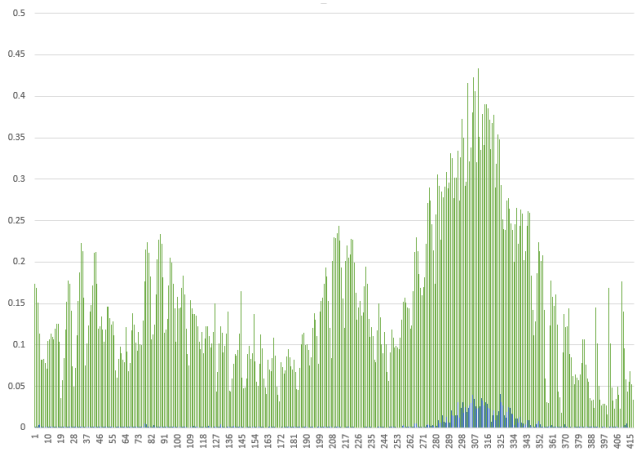


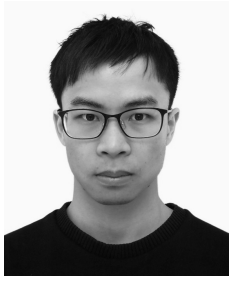
Fig. 10: Fitting errors measured using our capsule database. The horizontal axis shows the index of sampling vertex, and the vertical axis shows fitting error values. The green bar shows the values of original cubature weights, and the blue bar is those of our refined weights.

database is position-based which can capture more details for a deformation than pose-based. The experimental results show that our method achieves the deformation that well approximates full-space simulation. With position-based selection, we can keep the simulation in a very low-dimensional subspace. Furthermore, we refine original cubature weights for sparse collided deformations, thus making cubature more suitable for our local subspace database and realizing faster performance. These advantages make our scheme well-suited for real-time applications that involve collisions of deformable objects such as video games, surgery simulation, etc.

There are several aspects that we should consider in our future work. First, geometric properties of simulation meshes should be considered in the creation of collided points. In the region where curvature changes drastically, more collided points must be created. In addition, only a single rigid object is used to collide in our current scheme, and we plan to combine different databases and use different rigid objects to collide elastic objects at the same time. Moreover, we are also considering reusing subspaces for simulating local deformation and for different regions.

References

1. An, S.S., Kim, T., James, D.L.: Optimizing cubature for efficient integration of subspace deformations. ACM Trans. Graph. **27**(5), 165:1–165:10 (2008)
2. Barbič, J., James, D.L.: Real-time subspace integration for St. Venant-Kirchhoff deformable models. ACM Trans. Graph. **24**(3), 982–990 (2005)
3. Barbič, J., Sin, F., Grinspun, E.: Interactive editing of deformable simulations. ACM Trans. Graph. **31**(4), 70:1–70:8 (2012)
4. Barbič, J., Zhao, Y.: Real-time large-deformation substructuring. ACM Trans. Graph. **30**(4), 91:1–91:8 (2011)
5. Corsini, M., Cignoni, P., Scopigno, R.: Efficient and flexible sampling with blue noise properties of triangular meshes. IEEE Transactions on Visualization and Computer Graphics **18**(6), 914–924 (2012)
6. Hahn, F., Thomaszewski, B., Coros, S., Sumner, R.W., Cole, F., Meyer, M., DeRose, T., Gross, M.: Subspace clothing simulation using adaptive bases. ACM Trans. Graph. **33**(4), 105:1–105:9 (2014)
7. Harmon, D., Zorin, D.: Subspace integration with local deformations. ACM Trans. Graph. **32**(4), 107:1–107:10 (2013)
8. Hauser, K.K., Shen, C., O'Brien, J.F.: Interactive deformation using modal analysis with constraints. In: Proc. Graphics Interface, pp. 247–256. Canadian Human-Computer Communications Society, Mississauga, Ontario, CA (2003)
9. James, D.L., Pai, D.K.: Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. ACM Trans. Graph. **21**(3), 582–585 (2002)
10. Kim, T., Delaney, J.: Subspace fluid re-simulation. ACM Trans. Graph. **32**(4), 62:1–62:9 (2013)
11. Kim, T., James, D.L.: Skipping steps in deformable simulation with online model reduction. ACM Trans. Graph. **28**(5), 123:1–123:9 (2009)
12. Kim, T., James, D.L.: Physics-based character skinning using multidomain subspace deformations. IEEE Transactions on Visualization and Computer Graphics **18**(8), 1228–1240 (2012)
13. Li, S., Huang, J., de Goes, F., Jin, X., Bao, H., Desbrun, M.: Space-time editing of elastic motion through material optimization and reduction. ACM Trans. Graph. **33**(4), 108:1–108:10 (2014)
14. Pentland, A., Williams, J.: Good vibrations: Modal dynamics for graphics and animation. SIGGRAPH Comput. Graph. **23**(3), 207–214 (1989)
15. Shabana, A.A.: Theory of Vibration: Volume II: Discrete and Continuous Systems. Springer Science & Business Media (2012)
16. Sifakis, E., Barbič, J.: FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In: ACM SIGGRAPH 2012 Courses, SIGGRAPH '12, pp. 20:1–20:50. ACM, New York, NY, USA (2012)
17. Tang, M., Manocha, D., Otaduy, M.A., Tong, R.: Continuous penalty forces. ACM Trans. Graph. **31**(4), 107:1–107:9 (2012)
18. Teng, Y., Meyer, M., DeRose, T., Kim, T.: Subspace condensation: Full space adaptivity for subspace deformations. ACM Trans. Graph. **34**(4), 76:1–76:9 (2015)
19. Treuille, A., Lewis, A., Popovič, Z.: Model reduction for real-time fluids. ACM Trans. Graph. **25**(3), 826–834 (2006)
20. von Tycowicz, C., Schulz, C., Seidel, H.P., Hildebrandt, K.: An efficient construction of reduced deformable objects. ACM Trans. Graph. **32**(6), 213:1–213:10 (2013)
21. von Tycowicz, C., Schulz, C., Seidel, H.P., Hildebrandt, K.: Real-time nonlinear shape interpolation. ACM Trans. Graph. **34**(3), 34:1–34:10 (2015)
22. Xu, H., Barbič, J.: Pose-space subspace dynamics. ACM Trans. Graph. **35**(4), 35:1–35:14 (2016)



Duosheng Yu is a master student in the Graduate School of Arts and Sciences, the University of Tokyo, Japan. His research interests include run-time rendering and physics simulation in computer graphics. He received his bachelor's degree of computer science from the Jilin University of China in 2014.

Takashi Kanai is an associate professor in the Interfaculty Initiative in Information Studies, the University of Tokyo, Japan. His research interests include geometry processing and physics-based animation in computer graphics. He received his doctor degree of engineering from the University of Tokyo in 1998. He is a member of ACM, ACM SIGGRAPH, IPSJ (Information Processing Society of Japan), JSPE (Japan Society for Precision Engineering).