Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Pose Correction by Space-Time Integration

J. Martinez Esturo[1], C. Rössl[1], S. Fröhlich[2], M. Botsch[2], and H. Theisel[1]

[1] Visual Computing Group, University of Magdeburg
[2] Graphics & Geometry Group , University of Bielefeld

Abstract

The deformation of a given model into different poses is an important problem in computer graphics and computer animation. In a typical workflow, a carefully designed reference surface is deformed into a couple of poses, which can then act as a basis for interpolating arbitrarily intermediate poses. To this end the input poses should be free of geometric artifacts like self-intersections, since these degeneracies will be reproduced or even amplified by the interpolation. Not only are the resulting artifacts visually disturbing, they typically cause severe numerical problems for further downstream applications.

In this paper we present an automatic approach for removing these geometric artifacts from a given set of mesh poses, while maintaining the original mesh connectivity. The deformation from the rest pose to a target pose is faithfully reproduced by integration of a smooth space-time vector field, which by construction guarantees the absence of self-intersections in the repaired target pose. Our approach is computationally efficient, and its effectiveness is demonstrated on a range of typical animation examples.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Geometric Algorithms

1. Introduction

A central task in computer graphics, geometric modeling, and computer animation is the deformation of a shape from a given reference pose to a variety of new poses. To satisfy the demand for ever more realistic deformations without significantly increasing computation or simulation time, several recent approaches incorporate training data in the form of example poses: Given a rest shape and several example poses, new shape variations are created by custom-tailored interpolation or extrapolation within a suitable shape space [KMP07]. Example-based approaches have been proposed, e.g., for shape interpolation [XZWB05, LSLC05, KG08, WDAH10, FB11], skeletal skinning [LCF00, WPP07, WSLG07], shape deformations [SZGP05, FKY08, PJS06], as well as physical simulations [FYK10].

The reference model typically has been carefully designed and/or acquired from a physical object, often requiring a lot of manual work to check for and remove geometric inconsistencies, such as degenerate triangles, fold-overs, or self-intersections. The example poses are then created by deforming the rest pose and fine-tuning the result. In most cases, the rest pose and the example poses share the same structure, i.e., the same mesh connectivity, which we assume in the following for our approach.

Unfortunately, in practice many example poses suffer from geometric artifacts — even if the rest pose is a clean mesh. Oftentimes the degeneracies are introduced when deforming the rest pose into the target poses, e.g., by linear blend skinning [LCF00]. For similar reasons many models obtained from shape-databases also contain geometric inconsistencies. These artifacts are then reproduced (sometimes even amplified) in example-based mesh processing techniques. They do not only corrupt the visual appearance, they may even cause severe numerical problems for more sophisticated nonlinear approaches [FB11], where they slow down or even spoil convergence of optimization schemes.

In this paper we present an automatic post-processing technique for detecting and resolving geometric inconsistencies in given example poses. Based on a linear (hence robust) mesh interpolation technique, we first generate a set of intermediate key-frames between rest pose and target pose. From those we derive smooth vertex trajectories, to which we fit a 4D space-time vector field. Integrating this vector field through time deforms the rest pose mesh

DOI: 10.2312/PE/VMV/VMV11/033-040

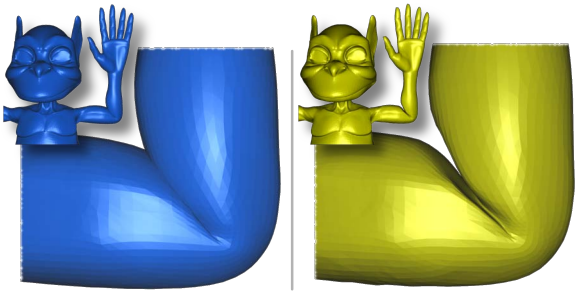

Figure 1: *Elbow deformation of the Goblin model. Self-penetrations of an articulated target pose (left •) are corrected by our approach locally* without *altering the connectivity of the underlying mesh (right •).*

into the target shape, but since path lines never cross each other in space-time [TWHS05] the resulting mesh is free of self-intersections by construction. Regions that initially contained self-intersections are repaired, the remaining parts are faithfully reproduced. Figure 1 shows an example.

2. Related Work

In this section we review related work. Our general scope is mesh repair, and in our approach we apply vector field based shape deformations and scattered data interpolation by radial basis functions.

Mesh repair. There is a variety of methods for repairing inconsistencies in geometric models like holes, self-intersections, folds, and topological noise in different types of geometric models. Methods can roughly be classified into surface based methods and volumetric methods (cf. the recent survey by Ju [Ju09]).

Surface based methods modify the surface meshes directly. In an early work, Bohn [Boh93] proposes a topological shell-closure operator that guarantees orientable surfaces. In order to also repair geometric inconsistencies, Barequet et al. [BDK98] employ heuristics for solving ambiguous geometric configurations. A frequent problem for scanned objects is topological noise. Guskov and Wood [GW01] remove small handles by a local wave-front mesh analysis and non-separating cuts.

Volumetric methods are based on implicit surface models that guarantee surfaces without self-intersections. For instance, Nooruddin and Turk [NT03] use voxelization to obtain an implicit surface, the repaired surface is generated from iso-contours. Space partitions can lead to efficient algorithms. In [Ju04] an octree space voxelizations of an in/out volume classification is used for feature adaptive mesh repair. Bischoff et al. [BPK05] present an octree-based mesh repair framework which provides high accuracy. A follow-up [BK05a] focuses on locality such that mesh connectivity is modified only locally in the vicinity of inconsistencies. Recently, nested space partitions have been proposed by Campen and Kobbelt [CK10] for model repair.

All these methods operate on a single surface, and all of them modify the internal structure, e.g., the mesh connectivity. In contrast to this, the input to our method consists of poses of the same mesh with fixed connectivity. Only the geometry of the meshes is corrected by a *deformation* approach: our algorithm does not modify mesh connectivity. We remark that there are deformation approaches which suppress intersections directly in the modeling process [BK03] or in a physical simulation of cloths [BWK03]. However, these approaches aim at interactive or physically plausible shape deformation rather than automatic model repair as a post-process.

Vector field based deformations. Our approach can be considered as a deformation method that exploits special properties of an underlying continuous deformation vector field. This vector field is determined automatically by the input poses. Surface deformations based on vector fields have been used by Funck et al. [vFTS06, vFTS07] for interactive and volume preserving shape editings. In their approach they exploit divergence-free fields to enable volume preservation. Martinez et al. [MRT10] extend this idea to simultaneous deformation of an arbitrary number of implicit surfaces. Kilian et al. [KMP07] use a different type of vector fields to obtain near isometric deformations.

Radial basis function interpolation. We apply radial basis functions (RBF) as a model for space-time vector fields. RBFs are a general and widely used tool for mesh-less scattered data interpolation. They are used in numerous applications on different sorts of data. Examples related to geometry processing include surface-fitting based on point samples [CBC*01, OBS05] and also real-time surface deformations [BK05b].

3. General Idea and Overview

This section gives an overview of our method, which is also illustrated in Figure 2. Details on the particular steps are provided in the subsequent Section 4.

Given are two poses of a surface mesh, a reference mesh and a target mesh that both share the same connectivity. The target shows defects such as self-intersections, and our goal is to remove defects and to repair the mesh.

We propose a deformation approach to repair the target pose. The deformation is represented as a space-time vector field integration, and we exploit the fact that path lines of integrated surface points do not intersect in space-time (see, e.g., [TWHS05]). This way we obtain a surface that approximates the given target and by construction is free of self-intersections.

We want to find a continuous time-dependent vector field describing the motion of the surface under deformation from the reference pose to the target pose. We do this by fitting a smooth vector field to a set of discrete samples. In our case,

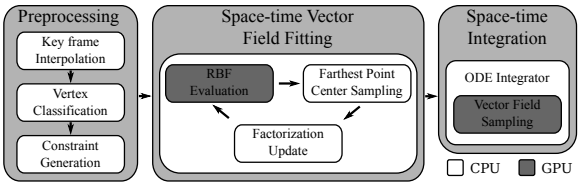

Figure 2: *Overview of our method. After space-time constraints are generated from the preprocessed reference and target poses, we fit a space-time vector field which describes the deformation. Then a space-time shape integration yields the repaired pose. Steps highlighted in • use GPU hardware acceleration.*

these samples are tangent vectors of the trajectories of surface points moving in space-time. We obtain the tangent vector samples by first generating a sequence of *key frames* that represent some intermediate surfaces. Then we fit trajectories as C^1-continuous cubic spline curves at each vertex to obtain continuous curves which describe the motion of each vertex.

Large parts of the target surface do not require any repair or modification. Our goal is to reproduce such regions as good as possible, whereas regions with defects in the target pose should be repaired. We achieve this by an automatic *classification* of surface regions. The reproduction of well-behaved surface regions is due to constraints on the vector field fitting. The key idea is that regions with defects do not generate any constraints on the vector field, i.e., we use the extrapolation provided by the underlying RBF model for repair. This can roughly be compared to hole filling in applications to surface reconstruction (see, e.g., [CBC*01]).

The vector field is represented as a radial basis function over the space-time domain (see, e.g., [TWHS05]). The function interpolates the constraints given by key frames and vertex classification. Then *fitting* the vector field leads to solving a linear system.

Finally we *integrate* all vertices of the reference pose through the vector field to obtain the corrected result pose. The integration exploits the property that path lines never cross each other in space-time to guarantee intersection-free time-surfaces.

4. Algorithmic Details

Our approach proceeds in three steps: generation of space-time constraints, vector field fitting, and integration. This is illustrated in Figure 2. In this Section we describe the algorithmic steps in detail.

We consider meshes $\mathcal{M}_k$ that share the same connectivity: the surfaces are defined by vertex positions $\mathbf{x}_i \in \mathbb{R}^3, i \in \mathcal{V}$, where $\mathcal{V}$ is the set of vertices. Let $\mathcal{M}_R$ be the reference pose, and $\mathcal{M}_T, T \neq R$ are target poses. We assume that the reference $\mathcal{M}_R$ does not contain defects, and the target poses should be repaired. Note that our goal is to maintain a consistent triangulation over all poses, i.e., the connectivity remains fixed. This is in contrast to other methods that focus on repairing a single mesh like [CK10].

Figure 3: *Preprocessing: key frame interpolation. From the reference (•) and one target shape (•) we generate a dense set of intermediate key frame shapes (transparent •).*

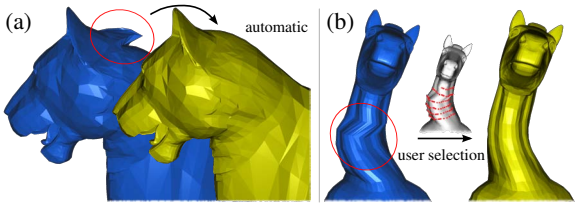

Figure 4: *Repair of common skinning artifacts. (a) Badly chosen skinning weights lead to intersection artifacts at regions of neighboring skeleton joints (• mesh), which are automatically detected and repaired by our method (• mesh). (b) Other types of skinning artifacts can also be repaired, but they require manual selection (• vertices). Note that the original skinning skeleton is not required for any correction.*

4.1. Preprocessing

Our algorithm works on the reference pose $\mathcal{M}_R$ and one single target pose $\mathcal{M}_T$ at a time. It does not require or use data of other target poses. The input is a pair of meshes $(\mathcal{M}_R, \mathcal{M}_T)$. The preprocessing step consists of key frame interpolation, intersection detection, and constraint generation.

Key frame interpolation. Our shape interpolation is based on interpolation in gradient space (see [XZWB05]). However, we could also use any other method for the generation of key frames (see, e.g., [SP04, KG08, WDAH10]). Note that the key frames may contain defects and artifacts. In fact, if the target pose contains defective regions, the intermediate key frames are likely to have defects, too. Figure 3 shows an example for key frame shapes. Every surface refers to a time step.

Vertex classification. We determine the surface points $\mathcal{V}^C \subset \mathcal{V}$ that refer to well-behaved regions as follows. Initially all surface parts are considered well-behaved, and we

initialize $\mathcal{V}^{\mathcal{C}} = \mathcal{V}$. Then we generate a number of key frame surfaces and scan these meshes for self-intersections. (We used 11 key frames in our examples.) We need to detect self-intersections in all intermediate key frames — not only in the target mesh — to capture the global support of evolving self-intersections. The intersection tests are performed efficiently using a space hierarchy of axis-aligned bounding boxes enclosing the triangles and exact geometric predicates provided by the CGAL library [CGA09]. Whenever we find self-intersecting triangles, their vertices and all vertices in their 1-ring are removed from $\mathcal{V}^{\mathcal{C}}$. Our experiments show that the removal of one additional ring of vertices works well in practice. Removing a larger neighborhood can lead to smoother transitions between regions. However, there is a trade-off between smoothness and reproduction. The classification can be modified manually to correct other types of artifacts that are not detected automatically (see Figure 4 for an example).

Space-time constraint generation. The key frames define trajectories of vertices moving in space-time. For vertex positions $\mathbf{x}_j(t_i), j \in \mathcal{V}$, we fit cubic C^1 spline curves $\mathbf{c}_j$ such that $\mathbf{c}_j(t_i) = \mathbf{x}_j(t_i)$. We use Hermite interpolation with tangent directions approximated by finite differences w.r.t. time. The trajectories $\mathbf{c}_j$ can be evaluated at arbitrary times $t \in [0,1]$, where $\mathbf{c}_j(0)$ and $\mathbf{c}_j(1)$ evaluate to points on the reference and the target surface, respectively.

For vector field fitting we require sampling of positions $\mathbf{c}_j(t)$ to place RBF centers and sampling of tangents $\frac{d}{dt}\mathbf{c}(t) = \dot{\mathbf{c}}(t)$ to evaluate constraints. The space-time deformation vector field is a 3D function on a 4D space-time domain, $\mathbf{f}(\mathbf{x}) : \mathbb{R}^4 \to \mathbb{R}^3$. Given the vertex classification $\mathcal{V}^{\mathcal{C}}$ and smooth trajectories $\mathbf{c}_j$ we can define pointwise interpolation constraints for the vector field function. Let $\mathbf{y}_j(t_i) = (\mathbf{c}_j(t_i), t_i)^{\mathrm{T}} \in \mathbb{R}^4$ be a point in space-time. Then we have the interpolation condition $\mathbf{f}(\mathbf{y}_j(t_i)) = \dot{\mathbf{c}}_j(t_i)$. We denote the set of constraints $\mathcal{C}$ as

$$\mathcal{C} = \left\{ \left(\mathbf{y}_j(t_i), \dot{\mathbf{c}}_j(t_i)\right) \mid j \in \mathcal{V}^{\mathcal{C}},\ t_i = \tfrac{i}{s},\ i = 0,\ldots,s \right\} .$$

$\dot{\mathbf{c}}_j(t)$ are the tangents to trajectories of vertices $j \in \mathcal{V}^{\mathcal{C}}$. We used $s = 12$ for our examples.

4.2. Space-time Vector Field Fitting

Finding the space-time vector field which interpolates all constraints in $\mathcal{C}$ is the most crucial step in our method. The quality of the vector field directly relates to the quality of the deformed shapes. Furthermore, fitting of a continuous space-time vector field that is smooth and well-behaved between constrained points dominates the runtime of our approach.

RBF interpolation. We represent the space-time vector field as a sum of radial basis functions (RBFs) that interpolate the constraints: $\mathbf{f}(\mathbf{y}_j(t_i)) = \dot{\mathbf{c}}_j(t_i)$. Our four-variate three-dimensional RBF model $\mathbf{f}$: $\mathbb{R}^4 \to \mathbb{R}^3$ is defined by a set of centers $\gamma_j \in \mathbb{R}^4$ and corresponding weights $\mathbf{w}_j \in \mathbb{R}^3$:

$$\mathbf{f}(\mathbf{y}) = \sum_j \mathbf{w}_j \phi_j(\mathbf{y}) + \pi(\mathbf{y}) .$$

The function $\phi_j(\mathbf{y}) = \phi\left(\left\|\gamma_j - \mathbf{y}\right\|\right)$ is the basic function corresponding to the jth center γ_j. We employ the triharmonic kernel $\phi(r) = r^2 \ln(r)$ in combination with a quadratic polynomial $\pi(\mathbf{y}) \in \Pi_2$, since then the resulting function $\mathbf{f}$ minimizes spline-like fairness energies and provides desirable extrapolation properties.

In order to interpolate n constraints $\mathbf{V} = [\mathbf{v}_1,\ldots,\mathbf{v}_n]^{\mathrm{T}} \in \mathbb{R}^{n\times 3}$ (corresponding to $\dot{\mathbf{c}}_j(t_i)$), we place the centers $\gamma_1,\ldots,\gamma_n$ at the constrained positions (corresponding to $\mathbf{y}_j(t_i)$) and solve a dense linear system for the coefficients of the basic functions $\mathbf{W} = [\mathbf{w}_1,\ldots,\mathbf{w}_n]^{\mathrm{T}} \in \mathbb{R}^{n\times 3}$ and of the quadratic polynomial $\mathbf{U} \in \mathbb{R}^{15\times 3}$:

$$\begin{pmatrix} \Phi & \Pi \\ \Pi^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{W} \\ \mathbf{U} \end{pmatrix} = \begin{pmatrix} \mathbf{V} \\ \mathbf{0} \end{pmatrix} .$$

$\Phi \in \mathbb{R}^{n\times n}$ and $\Pi \in \mathbb{R}^{n\times 15}$ are defined by $\Phi_{ij} = \phi_j(\gamma_i)$ and $\Pi_{ij} = \pi_j(\gamma_i)$, and $\{\pi_1,\ldots,\pi_{15}\}$ is a basis of Π_2. The 15 rows at the bottom augment the system with the orthogonality condition $\Pi^{\mathrm{T}}\mathbf{W} = \mathbf{0}$ (see [CBC*01]).

RBF center selection. The above system is dense, and we are limited to a few thousand centers in practice. Therefore the selection of centers plays a key role in RBF interpolation. We propose an iterative strategy for center selection which is an extension of the strategy used in [CBC*01]. The idea is to satisfy a prescribed fitting accuracy ε by selecting interpolation constraints.

Center selection is an iterative process. We start with the two most distant space-time points in $\mathcal{C}$ as the initial set of RBF centers. In each iteration we evaluate the fitting error, and we stop if the maximum error drops below a treshold ε, i.e., if $\max\{\,||\mathbf{f}(\mathbf{y}) - \dot{\mathbf{c}}||^2 \mid (\mathbf{y},\dot{\mathbf{c}}) \in \mathcal{C}\,\} < \varepsilon$. In each iteration, Carr et al. [CBC*01] enlarge the set of centers by the m space-time points with highest errors. We extend the original approach and add a farthest point sampling step: We select the m most distant points within p candidate center points with highest error, where $p \gg m$. We use a kd-tree to partition the 4D space-time domain for efficient farthest point queries. The rationale is to avoid generation of clusters of centers that would lead to ill-conditioned linear systems (see [BK05b]). It turns out that our strategy significantly reduces the number of centers required to achieve a prescribed error bound ε and increases the numerical stability (see Section 5). For all examples we used $m = 128$, $p = 1024$, and a fitting accuracy of $\varepsilon = 10^{-4} \cdot d$, where d is the length of the spatial bounding box diagonal of the reference pose. The algorithm terminated for all examples, i.e., the maximum error is reduced until it falls below ε.

We remark that numerical stability and convergence rate of iterative fitting are additionally improved by the vertex classification that excludes contradictory constraints.

Blocked factorization update. For the factorization of the resulting linear system we exploit symmetry and combine center selection with the factorization of the system. This way, we obtain an efficient update scheme for factorization. In each iteration we update an $(n+15)\times(n+15)$ LU factorization with partial pivoting of the RBF system which is given as

$$\mathbf{A} = \begin{pmatrix} \Phi_{0,0} & \Pi_0 \\ \Pi_0^{\mathrm{T}} & \mathbf{0} \end{pmatrix} = \mathbf{P}^{\mathrm{T}}\mathbf{L}\mathbf{U}\mathbf{R}\,.$$

The system matrix $\mathbf{A}$ corresponds to the n selected centers of the previous iteration. The matrices $\mathbf{P}$ and $\mathbf{R}$ are permutations (in the first iteration $\mathbf{R}=\mathbf{I}$), $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular. Permutations and triangular factors are updated in each iteration. The current iteration leads to a new $(n+m+15)\times(n+m+15)$ system $\mathbf{A}'$, which incorporates m additional centers. We rewrite $\mathbf{A}'$ by first applying global row and column permutations $\mathbf{G}$ to the new system matrix $\mathbf{A}'$ such that we can reuse the factorization of $\mathbf{A}$:

$$\mathbf{A}' = \begin{pmatrix} \Phi_{0,0} & \Phi_{1,0} & \Pi_0 \\ \Phi_{1,0}^{\mathrm{T}} & \Phi_{1,1} & \Pi_1 \\ \Pi_0^{\mathrm{T}} & \Pi_1^{\mathrm{T}} & \mathbf{0} \end{pmatrix} = \mathbf{G}\begin{pmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^{\mathrm{T}} & \Phi_{1,1} \end{pmatrix}\mathbf{G},$$

where the block matrix $\mathbf{C}^{\mathrm{T}} = \begin{pmatrix} \Phi_{1,0}^{\mathrm{T}} & \Pi_1 \end{pmatrix}$ was substituted. The new factorization of $\mathbf{A}' = \mathbf{P}'^{\mathrm{T}}\mathbf{L}'\mathbf{U}'\mathbf{R}'$ is then given by

$$\underbrace{\mathbf{G}\begin{pmatrix} \mathbf{P}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{P}}^{\mathrm{T}} \end{pmatrix}}_{\mathbf{P}'^{\mathrm{T}}} \underbrace{\begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \tilde{\mathbf{P}}\mathbf{S} & \tilde{\mathbf{L}} \end{pmatrix}}_{\mathbf{L}'} \underbrace{\begin{pmatrix} \mathbf{U} & \mathbf{T} \\ \mathbf{0} & \tilde{\mathbf{U}} \end{pmatrix}}_{\mathbf{U}'} \underbrace{\begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}\mathbf{G}}_{\mathbf{R}'}\,.$$

Its evaluation requires back-substitutions with known triangular factors to obtain $\mathbf{S} = \mathbf{C}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{U}^{-1}$ and $\mathbf{T} = \mathbf{L}^{-1}\mathbf{P}\mathbf{C}$. A complete LU factorization (with partial pivoting) $\tilde{\mathbf{P}}^{\mathrm{T}}\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \Phi_{1,1} - \mathbf{S}\mathbf{T}$ is required only for a $m\times m$ matrix. (*The supplemental material to this submission contains details of the derivation.*)

After each iteration the RBF coefficients are found using the updated factorization:

$$\begin{pmatrix} \mathbf{W} \\ \mathbf{U} \end{pmatrix} = \mathbf{R}'^{\mathrm{T}}\mathbf{U}'^{-1}\mathbf{L}'^{-1}\mathbf{P}'\begin{pmatrix} \mathbf{V} \\ \mathbf{0} \end{pmatrix},$$

where the matrix $\mathbf{V}$ contains the constrained tangent vectors at the current centers. Blocked factorization updates improve performance significantly as demonstrated in Section 5.

4.3. Space-time Shape Integration

The RBF interpolation defines a smooth vector field $\mathbf{f}$ in space-time. Interpolation of space-time constraints ensures both spatial and temporal continuity of $\mathbf{f}$. We integrate this vector field for all vertices of the reference surface starting at time $t=0$, and we expect an approximation of the target surface at $t=1$. The idea is to reconstruct the trajectories $\mathbf{c}_j(t)$ of vertices j with path lines in $\mathbf{f}$ starting at $\mathbf{x}_j(0)$. For well-behaved regions, the path lines are close to the original trajectories, and integration ends in $\mathbf{x}_j(1)$. However, as path lines never cross each other in space time (see, e.g., [TWHS05]) self-intersections cannot be reproduced — instead they are repaired.

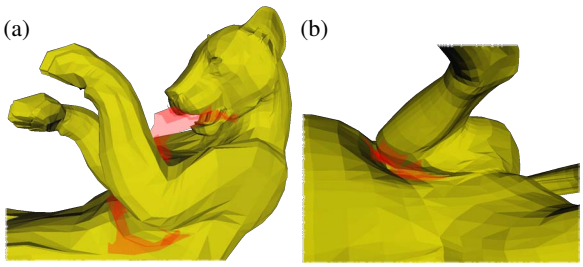

Figure 5: *Relocation of intersecting triangles. Deformation results where triangles in • indicate the locations of intersecting triangles of the target pose. If possible, relocation distance is minimal (b) while larger relocations are required for deeper initial penetration distances (a).*

For the numerical integration we use a standard fifth order Runge-Kutta integrator with adaptive step size control (see, e.g., [PTVF07]). This integrator performed well in all our experiments as the fitted vector fields turned out to be very smooth. The computation cost is dominated by evaluation of the RBF. To speed up computations we apply a parallel implementation on the GPU based on NVIDIA's CUDA framework. Compared to a parallel CPU implementation on a quad-core processor we obtain a speedup of 320% in double-precision compute mode. We apply the GPU evaluation similarly for setting up the linear systems.

We remark that by using global vector field integration our approach can only remove self-intersections that are not already present in the reference pose. Yet these do not impede our method otherwise: after vertex classification we simply re-add the corresponding vertices of the reference pose to $\mathcal{V}^{\mathcal{C}}$. Indeed, this way initial self-intersection are reproduced with no effects on the remainder of our method.

5. Analysis and Results

In this Section we analyze properties of our approach and its implementation and present results. Please see the accompanying video and additional material for more deformation results and examples of continuously deforming time integration shapes.

Target shape reproduction. Our goal is to repair defects. However, regions of the target mesh without any defects should be reproduced. Our experiments show that our method introduces geometric differences only locally at vertices $i \notin \mathcal{V}^{\mathcal{C}}$. Figure 6 shows an example were the overall shape of the target cat pose is maintained while all self-intersection are corrected. The example in Figure 7 shows color coded local geometric differences of some target models and our repaired models. The regions which show high

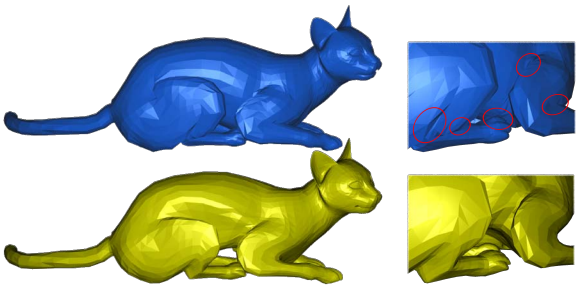

Figure 6: *Shape reproduction. The shape of the target cat model (●) is correctly reproduced in the non-intersecting regions of the head, neck, back and tail. The close-ups show the regions corrected by our method (●).*

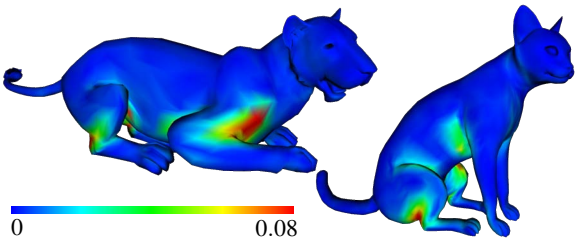

Figure 7: *Geometric differences. Shown are corrected surfaces generated by our method. The color code indicates the geometric differences to the given target shapes. (Models were scaled to the unit sphere to enable comparison). The regions where shapes differ significantly are regions with defects in the initial pose and which are repaired by our method. The experiment confirms accurate reproduction and smooth transitions to repaired regions.*

geometric differences in the plots are regions which have been repaired. All other regions are reproduced accurately.

Resolving intersections. Deformations of continuous shapes guided by a smooth space-time vector field cannot introduce new intersections of the shape during space-time integration. The reason for this is that path lines, trajectories of surface points in space-time, do not intersect. In all our experiments there are no newly introduced self-intersections. Figure 8 shows an example with non-trivial self-intersections. The left target shape contains 523 self-intersecting triangles. In contrast, our result on the right contains 22 self-intersecting triangles in the ears of the lion. All of these 22 self-intersections are already present in the reference shape. Therefore, our method successfully removes intersections which had been introduced by initially modeling the target shape. Figure 5 illustrates how the surface regions of $\mathcal{V}^C$ are relocated to their corrected intersection-free rest positions. The corrections are proportional to the penetration depths in order to dissolve the self-intersections correctly.

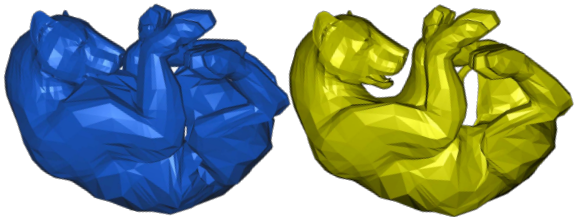

Figure 8: *Resolving complex self-intersections. We compare the given target mesh (●) to the repaired mesh (●). All self-intersections which were not already present in the reference pose have been successfully removed. The overall shape was maintained.*

Discretization. Integration cannot introduce self-intersections. This property holds strictly only for integration of a continuous surface. In the discrete setting, we are integrating only vertex positions, such that we have to assume a sufficiently dense sampling of the surface. However, none of our experiments suffered from under-sampling and resulting self-intersections. This issue is discussed similarly in [vFTS06] in a different context of zero divergence implying volume preservation.

Benefits for subsequent processing. The motivation for surface repairing is not restricted to improving visual appearance. Even more important is the improvement and support of other geometry processing methods which either suffer from defects or strictly require "clean" surface meshes. These are, e.g., methods based on nonlinear optimization. Here, the numerical computations frequently rely on well-behaved distribution of inner and dihedral angles. In our experiments we examine the recent example-driven deformation and interpolation approach [FB11], which applies Gauss–Newton iterations for nonlinear optimization. The optimization is sensitive to extreme dihedral angles (corresponding to local self-intersections) leading to increased number of iterations or even failure of convergence. Our experiments show that using the corrected poses the optimization required up to 60% fewer iterations until convergence. Moreover, for poses like the lion pose in Figure 7, the nonlinear optimization fails to converge for the original data but converges for our results.

Timings. We compare our center selection scheme to the scheme used in [CBC*01] experimentally in Table 1 (top). All timings were measured on Linux PC equipped with a quad core AMD Phenom II CPU running at 3.4GHz, an NVIDIA GTX 460 GPU and 4GB of host memory. Our experiments show that our scheme leads to better conditioning of linear systems, which in turn leads to fewer centers and a significantly faster overall fitting process. The plot in the bottom of Table 1 demonstrates the runtime speedup achieved when using our block factorization update scheme. Note that the speedup rate grows super-linearly with the number of centers.

In all our experiments, the preprocessing stage required on average between 1.5s (lion), 1.9s (horse), and 2.3s (cat)

Poses ($\lvert\mathcal{V}\rvert$)	[CBC*01]	Ours
Lion ($5 \times 5k$)	1×10^{10} / $12k$ / 2.2×10^3	2×10^6 / $2.7k$ / 13
Cat ($5 \times 7k$)	2×10^{11} / $15k$ / 3.4×10^3	2×10^7 / $4.5k$ / 42
Horse ($10 \times 8k$)	3×10^{10} / $13k$ / 3×10^3	7×10^6 / $4.6k$ / 43
Goblin ($86k$)	exceeds memory limit	6×10^8 / $10k$ / 603

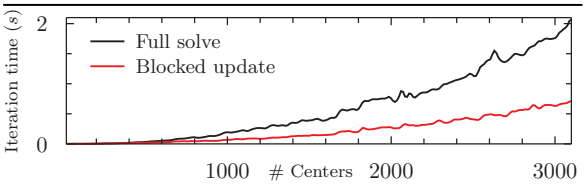

Table 1: *RBF center selection. Top: The table shows maximal condition number $\kappa(\mathbf{A}_i)$, average number of centers, and average total run times for vector field fitting. We compare the center selection scheme in [CBC*01] (left) and our new scheme (right), with blocked factorization update and GPU based evaluation. The rows show results for different data sets (averages of multiple poses). All times are given in seconds. Bottom: The plot compares our blocked factorization update to traditional factorization.*

for the smaller models; it required 19s for the goblin. Typical timings for integration are 29s (lion), 32s (cat), and 39s for the goblin (10k centers). Note that the integration does not only depend on the number of vertices and centers but also on the smoothness of the vector field: For instances integration required only 19s for the horse model.

6. Discussion

From a designer's point of view self-intersections may indeed be desirable: they tend to generate a more natural *visual appearance* especially at joints of neighboring articulated parts (see Figure 1), although the actual surface does not represent the outer hull of a solid object anymore. However, we believe that other kinds of self-intersections (see Figures 4,6,8) require corrections even for visual plausibility. This is especially necessary for multiple poses of a single model as all interpolations or combinations of these shapes would inherit the visual artifacts. Even more important than visual aspects is the improvement of the triangulation in view of further algorithmic processing stages. This is a central goal of any mesh repair method. Our experiments show that numerical properties of a typical non-linear mesh blending are significantly improved.

Our approach has several limitations. Self-intersections that are already present in the reference pose are reproduced, they are not corrected. In order to guarantee no self-intersections in the target, we require no self-intersections in the reference. This is not a severe limitation because the reference surface is usually modeled from scratch, and the additional effort to manually remove defects is often not significant.

Our method reproduces the shape of well-behaved regions. However, the reproduction is only nearly exact and slight variations in the final vertex coordinates are possible since the fitted vector field is only exact up to the error bound ε. Moreover, the tangent curves are only approximated in a *pointwise* way by the interpolation constraints. Yet, if higher quality reproductions are necessary the fitting error bound ε and the number of interpolation time steps s can always be adjusted accordingly to improve the results.

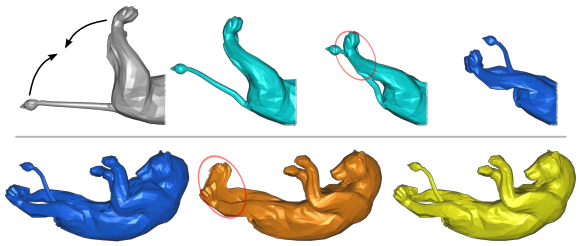

Figure 9: *Limitations. The example shows key frames with opposing directions of motion. This leads to global self-intersections in the interpolated key frames: the lion's tail moves through the leg on the way to the target pose; there is no intersection between tail and leg in the target pose (top). By construction, this situation cannot be reproduced by vector field integration. Instead, integration results in a partially corrected (●) shape. The remaining regions are copied from the original target (bottom).*

Guaranteeing absence of self-intersections is a main feature of our method. However, this can also impose a fundamental limitation. Figure 9 shows an example where the key frame interpolation generates a global self-intersection: the lion's tail moves through the leg. Generally, such self-intersections can occur in practice. In a sense the input to interpolation methods is ill-posed: these methods usually depend on local surface properties such as isometry or bending energy, they do not incorporate global (or even semantic) properties. Therefore, global self-intersection cannot be avoided automatically in key frame generation. By construction, our method is unable to reproduce self-intersections in space-time. So in the example, the lion's tail cannot move "through" the leg — it will always stay in front of the leg while both regions are deformed. In such cases, we ignore the result from surface integration for the affected region and replace it instead by the corresponding region of the input target mesh.

7. Conclusions

We presented a novel automatic approach to mesh repair, which is tailored to the correction of a sequence of meshes with same connectivity representing different poses of the same shape. Our aim is to keep the mesh connectivity fixed; this is in contrast to existing mesh repair methods. This sort of mesh repair is important not only to suppress visual artifacts but also to improve numerical stability of further mesh processing stages.

Our approach to mesh repair is based on continuous mesh deformation. The deformation is defined by space-time integration of a smooth vector field. This guarantees that the deformation of the reference surface does not introduce self-intersections in the desired target shape. Our approach is efficient thanks to first, an improved strategy to select basis functions, second, a novel blocked factorization update, and third exploiting the GPU. Experiments for a variety of examples confirm the effectiveness of the method.

The guaranteed avoidance of self-intersections is a main feature of a space-time surface integration. However, this imposes also a limitation as self-intersections cannot be reproduced. At present we apply local modifications of the mesh to handle such cases. A challenge for future research is to avoid "collisions" leading to global self-intersections during pose interpolation in the first place. Anther possible direction is extension of the approach to space-time coherent remeshing.

Acknowledgements

The lion, cat, and horse poses are courtesy of Robert Sumner. This work was supported by the German National Academic Foundation and by the German Research Foundation (Center of Excellence in "Cognitive Interaction Technology", CITEC).

References

[BDK98] BAREQUET G., DUNCAN C., KUMAR S.: Rsvp: a geometric toolkit for controlled repair of solid models. *IEEE TVCG 4*, 2 (1998), 162 –177. 2

[BK03] BOTSCH M., KOBBELT L.: Multiresolution surface representation based on displacement volumes. In *Comput. Graph. Forum* (2003), Eurographics Association. 2

[BK05a] BISCHOFF S., KOBBELT L.: Structure preserving cad model repair. *Comput. Graph. Forum 24*, 3 (2005), 527–536. 2

[BK05b] BOTSCH M., KOBBELT L.: Real-time shape editing using radial basis functions. *Comput. Graph. Forum 24*, 3 (2005), 611–622. 2, 4

[Boh93] BOHN J. H.: *Automatic CAD-model repair*. PhD thesis, Rensselaer. Polytechnic Institute, Troy, NY, USA, 1993. UMI Order No. GAX94-05656. 2

[BPK05] BISCHOFF S., PAVIC D., KOBBELT L.: Automatic restoration of polygon models. *ACM Trans. Graph. 24* (2005), 1332–1352. 2

[BWK03] BARAFF D., WITKIN A., KASS M.: Untangling cloth. In *Proc. SIGGRAPH* (2003), ACM, pp. 862–870. 2

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proc. SIGGRAPH* (2001), ACM, pp. 67–76. 2, 3, 4, 6, 7

[CGA09] CGAL: Computational Geometry Algorithms Library, 2009. http://www.cgal.org. 4

[CK10] CAMPEN M., KOBBELT L.: Exact and Robust (Self-)Intersections for Polygonal Meshes. *Comput. Graph. Forum 29*, 2 (2010), 397–406. 2, 3

[FB11] FRÖHLICH S., BOTSCH M.: Example-driven deformations based on discrete shells. *Comput. Graph. Forum* (2011), to appear. 1, 6

[FKY08] FENG W.-W., KIM B.-U., YU Y.: Real-time data-driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph. 27*, 3 (2008), 91:1–91:9. 1

[FYK10] FENG W.-W., YU Y., KIM B.-U.: A deformation transformer for real-time cloth animation. *ACM Trans. Graph. 29*, 4 (2010), 108:1–108:9. 1

[GW01] GUSKOV I., WOOD Z. J.: Topological noise removal. In *Proc. Graphics interface* (2001), pp. 19–26. 2

[Ju04] JU T.: Robust repair of polygonal models. In *Proc. SIGGRAPH* (2004), ACM, pp. 888–895. 2

[Ju09] JU T.: Fixing geometric errors on polygonal models: A survey. *J. Comput. Sci. Technol. 24* (2009), 19–29. 2

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Trans. Graph. 27* (May 2008), 12:1–12:13. 1, 3

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. *ACM Trans. Graph. 26*, 3 (2007). Proc. SIGGRAPH. 1, 2

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. SIGGRAPH* (2000), ACM Press/Addison-Wesley, pp. 165–172. 1

[LSLC05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3 (2005), 479–487. 1

[MRT10] MARTINEZ ESTURO J., RÖSSL C., THEISEL H.: Continuous deformations of implicit surfaces. In *Proc. VMV* (2010), Eurographics Association, pp. 219–226. 2

[NT03] NOORUDDIN F., TURK G.: Simplification and repair of polygonal models using volumetric techniques. *IEEE TVCG 9*, 2 (2003), 191 – 205. 2

[OBS05] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: 3d scattered data interpolation and approximation with multilevel compactly supported rbfs. *Graph. Models 67* (May 2005), 150–165. 2

[PJS06] POPA T., JULIUS D., SHEFFER A.: Material-aware mesh deformations. In *Proc. Shape Modeling International* (2006), IEEE Computer Society, pp. 141–152. 1

[PTVF07] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007. 5

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23* (2004), 399–405. 3

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. *ACM Trans. Graph. 24*, 3 (2005), 488–495. 1

[TWHS05] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Topological methods for 2D time-dependent vector fields based on stream lines and path lines. *IEEE TVCG 11*, 4 (July - August 2005), 383–394. 2, 3, 5

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. In *Proc. SIGGRAPH* (2006), pp. 1118–1125. 2, 6

[vFTS07] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Elastic secondary deformations by vector field integration. In *Proc. SGP* (2007), vol. 257, ACM, pp. 99–108. 2

[WDAH10] WINKLER T., DRIESEBERG J., ALEXA M., HORMANN K.: Multi-scale geometry interpolation. *Comput. Graph. Forum 29*, 2 (2010), 309–318. 1, 3

[WPP07] WANG R. Y., PULLI K., POPOVIĆ J.: Real-time enveloping with rotational regression. *ACM Trans. Graph. 26*, 3 (2007). 1

[WSLG07] WEBER O., SORKINE O., LIPMAN Y., GOTSMAN C.: Context-aware skeletal shape deformation. *Comput. Graph. Forum 26*, 3 (2007), 265–273. 1

[XZWB05] XU D., ZHANG H., WANG Q., BAO H.: Poisson shape interpolation. In *Proc. Symposium on Solid and Physical Modeling* (2005), ACM. 1, 3