Real-time Variable Rigidity Texture Mapping

Charalampos Koniaris
Disney Research
ckoniaris@disneyresearch.com

Kenny Mitchell
Disney Research
kenny.mitchell@disneyresearch.com

Darren Cosker
University of Bath
d.p.cosker@bath.ac.uk

ABSTRACT

Parameterisation of models is typically generated for a single pose, the *rest pose*. When a model deforms, its parameterisation characteristics change, leading to distortions in the appearance of texture-mapped mesostructure. Such distortions are undesirable when the represented surface detail is heterogeneous in terms of elasticity (e.g. texture with skin and bone) as the material looks "rubbery". In this paper we introduce a technique that preserves the appearance of heterogeneous elasticity textures mapped on deforming surfaces by calculating dense, content-aware parameterisation warps in real-time. We demonstrate the usefulness of our method in a variety of scenarios: from application to production-quality assets, to real-time modelling previews and digital acting.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*; Computer Graphics [**Three-Dimensional Graphics and Realism**]: [Color, shading, shadowing, and texture]

Keywords

texture mapping, parameterisation distortion, mesh deformation, position-based dynamics

1. INTRODUCTION

Texture mapping is the process of mapping detail (colour, bump or displacement) to a surface using a corresponding parameterisation – the most common case being a 2D parameterisation of a 3D surface [8]. Some representations have natural parameterisations (e.g. NURBS), while others, such as polygonal meshes, require non-trivial methods or manual input to obtain parameterisations. In the case of polygonal meshes, parameterisations are represented in the same way as vertices: as piecewise-linear approximations to continuous functions. Parameterisations are ideally isometric but that is rarely the case. As such, a standard metric for the quality of a parameterisation is the distortion introduced by the mapping [17].

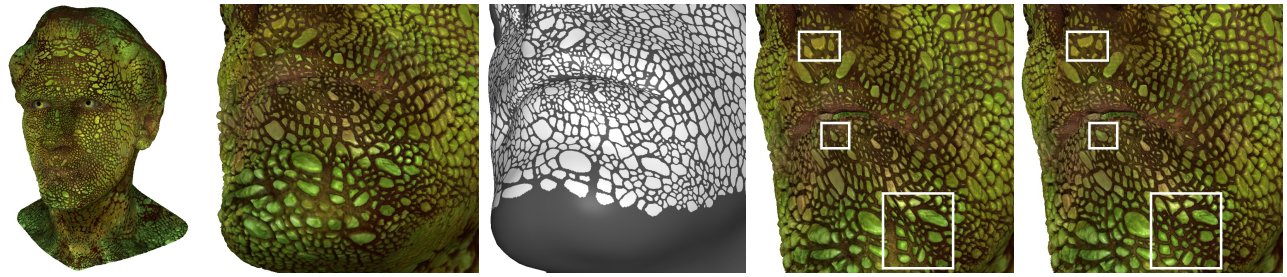
In a typical scenario, when creating a 3D model an artist will model low complexity geometry and paint texture, displacement or bump maps according to a specific mesh pose (e.g. a rest position) to simulate high complexity geometric detail. This data is stored using the 2D parameterisation. Given the piecewise-linear nature of the parameterisation, subsequent non-rigid deformation of such meshes will result in apparent distortions in the mapped detail (e.g. elastic stretching or squashing). While some surfaces may require this behavior (e.g. skin, rubber), it is preferable to be able to finely tune the degree of elasticity or restrict certain elements to remain entirely rigid (e.g. horns, armour elements and scales). While physical simulations may be employed to approximate this behavior, these are typically not appropriate for real-time animation and interactive editing and can be impractically slow for very detailed mesostructure. Even in production workflows where quality is prioritised over performance, this is still a difficult and relevant problem ([4, 7]) .Therefore, to reduce this type of distortion, either the parameterisation needs to be regenerated, or additional detail maps must be authored for key poses, or the mesh needs to be carefully edited so that deformation does not cause rigid texture areas to deform.

Contributions. In this paper, we introduce a novel method to reduce visual artifacts caused by the deformation of a parameterised surface in a user-controllable way in real-time. This allows a variety of texture mapped detail to be applied to an animated model without it undergoing visually undesirable behaviors.

Our main contribution is a real-time parameterisation distortion matching algorithm, which is guided by user-supplied rigidity maps and *warp masks* represented as an additional *scalar* texture and a bitmask respectively. The algorithm selectively matches the parameterisation distortion of a deformed frame to the one at the rest pose by warping the parameterisation domain. The effect is that textured areas selectively preserve their shape and size in rest pose at a varying degree, depending on their rigidity values. The warp is a grid deformation in UV space, mapping the domain bijectively onto itself. The grid dimensions directly affect the fidelity of the warp: our method supports any size that can be mapped to GPU memory (16384x16384 for modern graphics cards). For high-resolution production models that use several UV tiles (UDIM UV mapping), the charts in each tile can be individually processed.

The rest of this paper is organised as follows. We first describe previous work in related fields of research (sec. 2). We then present an overview of the method (sec. 3), followed by detailed descriptions of its components: simulation of texture-space warps (sec. 4) and rendering of the warped textures (sec. 5). We then present the results (sec. 6) and follow with implementation details (sec. 7) and conclude our main findings (sec. 8).

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235



(a) Model **(b)** Rest pose **(c)** Rigidity map **(d)** Deformed pose **(e)** Deformed pose, corrected texture-space detail

Figure 1: *Rigid features on an animated face model. Zoomed-out view of an animated frame (a), authored rigid features shown in (c). The parameterisation is constant, so deforming from the rest pose (b) to an animation frame (d) compresses the features. Using our parameterisation warps, the shape and scale of the features are preserved (d).*

2. RELATED WORK

In order to prevent parameterisation distortions caused by mesh deformations, the simplest approach is to control deformations; several recent shape-aware deformation or skinning methods can be applied to achieve this goal ([29, 27, 31, 26, 2, 10]). Such methods typically work with a coarse representation of the mesh and do not take into account fine-scale information. Therefore, they are complementary to our stated goal, which is to correct the appearance of fine-scale, texture-mapped heterogeneous data using a given deformation. Still, a number of techniques have been developed that approach mesh deformation from a content-sensitive point of view and as such, we briefly discuss them below.

Popa et al. [21] approach content-aware deformation by introducing local bending and shearing stiffnesses as factors in how a mesh deforms. Given such material information, and transformations for a number of anchor triangles, they calculate the deformation of the mesh as a weighted sum or blend of the anchor transformations. The material information is user- or data-driven, providing additional control on how parts of the mesh deform when editing it. As the anchor transformations are required to be a combination of rotations and uniform scales, the space of supported deformations is restricted.

Kraevoy et al. [14] focus on protecting vulnerable parts of a complex model under global non-uniform scaling. They define a vulnerability map on a volumetric grid that encloses the object, and transform the grid while respecting this map. While they estimate vulnerability based on *slippage* and *normal curvature*, the map can be user-driven. The technique focuses only on a very special deformation case (non-uniform scaling transform), so it's not applicable to more complex deformations.

Yang et al. [30] simulate skin sliding by remeshing the surface based on resampling of its parameter space. They use the *Force Density Method* (FDM) to construct embeddings of original and deformed patches into their parameter domains. As the technique deforms the actual geometry and force densities are specified on edges, so the deformed patch needs to be highly tessellated and the result is dependent on the triangulation, which reduces the flexibility and applicability of the method.

Müller and Chentanez [19] add wrinkles as fine surface detail to coarse, deforming surfaces in real-time. Their approach shares similarity to ours in terms of running a simulation on a high-resolution mesh overlaid on the coarse base mesh, but they only simulate wrinkling for smooth, homogeneous surfaces. In contrast, our method allows more complex deformations, and for non-homogeneous fine-scale detail.

Dekkers and Kobbelt [3] achieve content-aware mesh deformation by using seam carving on dense meshes, but with several limitations. The seam lines which determine the saliency of the mesh are restricted to ring shapes and require precomputation (no times are given). In contrast, our technique supports dynamic/animated saliency maps of any form. The method works with dense triangle meshes with no parameterisation and modifies the mesh and connectivity, in contrast to our technique that works with coarse quad meshes and retains the geometry but warps the parameterisation domain. Additionally, the technique is not well suited for animation as temporal coherence artifacts can be observed when applying deformation on the handles, while our simulation produces smoothly changing results.

If the deforming mesh cannot be modified or controlled, content-aware reduction of texture deformation can be achieved by modifying the parameterisation. The majority of existing parameterisation algorithms focus on generating a parameterisation by minimizing a global distortion metric (often independent of the data being mapped) at the rest pose of the mesh [16, 9, 25, 5]. While typically the distortion reduction can become content-aware by introducing spatially-varying weights, the per-vertex nature of the optimization cannot capture the fine-scale texture-mapped details and their variation in rigidity. More importantly, they do not address the problem of *distortion matching* as they address distortion minimization. Finally, they do not guarantee temporal coherence when applied to a series of smoothly deforming poses. Below we briefly discuss methods that take into account content, temporal coherence or distortion matching.

Falco and Driskill [4] reduce distortion on artist-defined areas over a mesh under deformation. They regenerate the texture by detecting and correcting the movements of points in relation to feature centers. However, the rigidity map is binary (not continuous) and compression is not handled, therefore the method is restricted in its applications.

Sander et al. [23] minimise a signal-stretch metric that allows reduction of distortions of any vector-valued function (the signal) defined over the domain. The metric is non-linear and the process requires a few minutes per model. While the technique is content-aware, it does not take into account temporal coherence of the parameterisation, which is important when considering a deforming mesh.

Sheffer and De Sturler [24] overlay a 2D uniform Cartesian grid on the texture parameterisation domain (as a 2D triangle mesh) and warp the grid so that the warped parameterisation minimises edge length distortions. While this technique uses an overlaid grid to warp the parameterisation, it is not content-aware and so ignores features which should remain rigid or fixed (i.e. non-sliding features).

Ptex, by Burley and Lacewell [1], eliminates the need for explicit parameterisation by using the natural one afforded by subdivision surface quad-faces and providing anisotropic filtering between

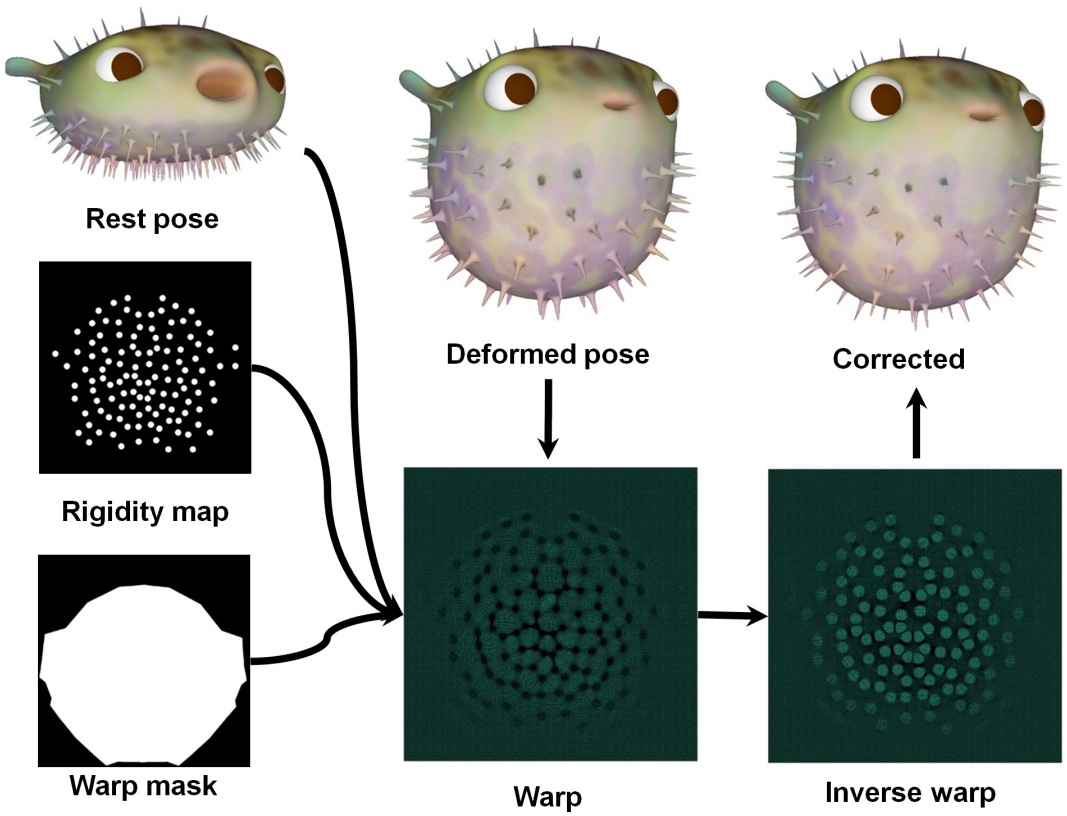


Figure 2: *Overview of the method. Given the mesh (rest pose and animation) and texture, we first artistically define the rigidity map and the warp mask. The rigidity map is used to generate a fine regular grid (at map resolution) that serves as the simulation mesh. Constraints are defined over this simulation mesh, whose stiffnesses are directly derived from the rigidity map (eq. 7), while the warp mask is used to select a subset of the grid constraints to simulate. The maps and mesh data are then used by our novel optimization process to hierarchically calculate the warped parameterisation domains (section 4). The warp is then inverted so that it represents new texture rather than geometry sampling coordinates. The resulting dense, deformed grids can be then used to render the dynamically warped texture (section 5).*

faces. While this eliminates many of the explicit parameterisation issues, such as distortions and seams, animated meshes still pose a problem, as the individual quads still deform and distortions are reintroduced.

Koniaris et al. [13] use distortion control maps on rectangular areas of a mesh and apply a nonlinear optimisation method over the rectangular domain. The technique can be used in real-time for simple distortion control maps, but scales poorly with map complexity.

Jin et al. [11] calculate content-aware parameterisations by minimizing a modified Least-Squares Conformal Map (LSCM) energy metric guided by importance maps. However the parameterisations take seconds to calculate and animated models are not considered.

Li et al. [15] reduce parameterisation distortions caused by deforming geometry using a thin hyperelastic skin simulation approach. The technique has interactive performance but does not take into account heterogeneous fine-scale detail.

Grabli et al. [7] re-parameterise UV charts driven by a feature map so that rigid features undergo a rigid transformation at a deformed frame, distributing the distortion to non-rigid areas using a technique based on Feature Aware Texturing by Gal et al. [6]. The technique can handle challenging cases and high-resolution textures, but is slow and therefore aimed towards production workflows. Temporal coherence is achieved by guiding the re-parameterisation using positional constraints.

Image retargeting techniques [22, 28, 12] focus on content-aware resizing of images. This is a very specialised case of our goal, as the surface and parameterisation of the image plane are trivial, and the deformation is typically a simple non-uniform scaling transform. Such techniques do not take into account sliding of features or temporal coherence, as these are not issues in the context of image retargeting.

Gal et al. [6] introduce a content-aware image warping technique that allows arbitrary 2D warps instead of just non-uniform scaling. As such, it can be used for 2D texture mapping with some limited uses in 3D. They use a binary feature mask to specify salient features and calculate a deformed mapping so that such features undergo a similarity transformation. However, our approach and goals differ in two main ways: (1) we are required to generalise to arbitrary surface deformations in 3D space, (2) we are required to account for sliding of features and temporal coherence.

In production and in practice, in order to reduce texture space distortion variations in sensitive regions of an animated mesh, artists need to manually add vertices, tightly bounding the rigid area and making sure that it does not distort under deformation. For example, for rigged models, the regions around joints are the most prone to distortions, so additional vertices may be placed. When the deformation is known, additional vertices can be placed appropriately so that deformation is spread to areas that do not contain any salient rigid features. The problem remains when the deformation is unknown or varying so much that adding and manually animating vertices becomes impractical. Procedurally generated detail, static or animated, provides an even greater challenge as the location of the additional vertices cannot be easily determined.

3. OVERVIEW

Our method effectively generates parameterisation warps in real-time given the geometry in the current and rest poses in addition to a user-specified rigidity map and a warp mask (see figure 2). The warps are calculated using a solver that is based on the Position-Based Dynamics (PBD) framework by Müller et al. [20]; we define a warp energy (sec. 3.1) as a scalar non-linear constraint function fit for minimization using PBD.

To aid clarity of exposition, we outline our notation here. The parameterisation is expressed as a bidirectional mapping between 3D and 2D space: f_O and f_D map the 2D unit square to rest and deformed pose 3D coordinates accordingly ($\mathbb{R}^2 \rightarrow \mathbb{R}^3$), while f_O^{-1} and f_D^{-1} perform the inverse mapping. W is a function that warps the unit square bijectively onto itself.

The rigidity map R stores continuous values in the range $[0,1]$: a value of 0 represents infinitely elastic material, whereas 1 represents an as-rigid-as-possible material. Indexing a grid X is written as $X_{i,j}$. Linear sampling of the domain of a function X is written as $X(s,t)$. If the function is represented as a grid of values, sampling interpolates the points linearly.

The warp mask M simply specifies areas that can or cannot warp (using values 1 and 0 respectively). For example, areas in texture space that are *on* or *outside* chart borders should not warp. Additionally, this mask can be used to control sliding by masking out curves that act as barriers, for example preventing sliding across creases or other high-curvature areas of a mesh.

3.1 Parameterisation distortion matching

Our goal is to match the parameterisation distortion in the deformed frame to the one in the rest pose, weighted by rigidity (rigid areas should match better than non-rigid). For an infitesimally short parameterisation segment p_0p_1, this "matching" energy is calculated as follows:

$$\begin{aligned} \ell_{\text{org}}(p_i,p_j) &= ||f_O(p_j) - f_O(p_i)|| \\ \ell_{\text{def}}(p_i,p_j) &= ||f_D(W(p_j)) - f_D(W(p_i))|| \\ E_{\text{match}}(p_i,p_j) &= \max_{p \in p_i,p_j} (R(p))(\ell_{\text{def}}(p_i,p_j) - \ell_{\text{org}}(p_i,p_j)) \end{aligned} \quad (1)$$

where f_O,f_D the parameterisations for the original and deformed poses $\mathbb{R}^2 \rightarrow \mathbb{R}^3$, $R(p)$ the scalar rigidity for a point p and W our warping function: $\mathbb{R}^2 \rightarrow \mathbb{R}^2$. The rigidity of a segment p_i,p_j is calculated as the maximum of the two endpoints.

We minimise this energy over the whole re-parameterisation domain as follows:

$$E_{\text{total}} = \sum_{p_ip_j \in V} E_{\text{match}}(p_i,p_j) \quad (2)$$

where V is the the set of segments of a 2D, 8-connected regular grid that covers the whole parameterisation domain.

The warp is represented by a regular grid of values in $\mathbb{R}^2$ and maps the unit square bijectively onto itself in a piecewise-linear manner.

4. SIMULATION: EXTENDED POSITION BASED DYNAMICS

We use the PBD framework to effectively solve an optimization problem, as it offers several characteristics beneficial to our goal. First, it provides *control over positions*, which is necessary in order to fix points on the deformation grid. This control over positions allows control of sliding at a fine-scale: we can trivially define parts of the domain to be more difficult to move than others. PBD also easily *handles non-linear energy functions*, which is again necessary as our distortion metric (eq. 1) is non-linear. Finally, PBD can be easily adapted to support level-of-detail using a hierarchical version of the algorithm. Its abstract, framework nature allows a variety of applications, such as the ones we develop in this paper.

The PBD simulation mesh is a 2D, 8-connected regular grid, finely tessellated so that it captures the rigidity variation on the surface (grid dimensions being equal to rigidity map resolution). The rest state of the grid covers the unit square and is used as the initial –identity– warp configuration. In the next section we overview the PBD process and describe how it is used to generate the warps.

4.1 PBD process

Similar to Müller et al. [20], a single PBD simulation iteration is composed of the following steps:

1. **Velocity damping:** Used to eliminate oscillations caused by the integration step.
2. **Calculation of estimated positions:** Uses an explicit Euler integration step to predict the new positions.
3. **Constraint projection:** Manipulates the estimated positions so that they satisfy given constraints using Gauss-Seidel type iterations.
4. **Regularization:** We introduce this step to adaptively regularise the estimated positions.
5. **Integration:** Moves positions to optimised estimates and updates velocities accordingly.

In our approach, we concentrate specifically on novelly adapting the constraints and constraint projection of the PBD algorithm and introduce a new *regularization* step immediately after the constraint projection. Finally, given the fine tessellation of the grid we use in PBD, we propose a hierarchical strategy for optimization that improves convergence and therefore overall performance of simulation.

4.2 Constraints and constraint projection

As described, one of our main technical novelties with respect to the PBD algorithm is adapting its constraints to our problem domain needs. PBD constraints are scalar functions. We only use a distance constraint, with cardinality of 2 and a type of equality. Therefore we define a constraint function that depends on two points and that has to be satisfied exactly; the constraint function is identical to equation 1:

$$C(p_i,p_j) = E_{\text{match}}(p_i,p_j) \quad (3)$$

where the constraint pairs p_i,p_j are all the edges in the 8-connected grid; the set V in equation 2. For an $N \times N$ grid, they are:

$$\begin{aligned} V_{\text{horz}} &= p_{i,j}, p_{i+1,j}\ , \forall i \in [1,N-1], j \in [1,N] \\ V_{\text{vert}} &= p_{i,j}, p_{i,j+1}\ , \forall i \in [1,N], j \in [1,N-1] \\ V_{\text{shear0}} &= p_{i,j}, p_{i+1,j+1}\ , \forall i \in [1,N-1], j \in [1,N-1] \\ V_{\text{shear1}} &= p_{i+1,j}, p_{i,j+1}\ , \forall i \in [1,N-1], j \in [1,N-1] \end{aligned}$$

We calculate the corrected estimated positions according to Müller et al. [20], but we add a few extra steps in the process.

First, before any calculations take place, we check if any of the constraints is immovable (by reading the warp mask), or if both are completely non-rigid. In any of these two cases, the constraint is not projected.

After (and if) the points are projected, we rotate the constraint segment around its center by a user-defined percentage towards its original orientation: one of 0° (V_{horz}), 45° (V_{shear0}), 90° (V_{vert}) or 135° (V_{shear1}). As the parameterisation domain is bounded, reducing rotations will prevent large angular distortions especially in non-rigid regions. We also use this "orientation preservation" parameter in cases of extreme compression, as it prevents the simulation mesh from folding.

Finally, to prevent constraints overshooting, we limit the maximum correction length by limiting each constraint point to the bounding box derived from their 8-neighborhood.

4.3 Regularization

As non-rigid constraints are not moved during the projection step, this can result in abrupt changes in non-rigid areas near boundaries to rigid areas. While we can always apply projection by setting a minimum rigidity value to non-rigid constraints, this will most likely result in the whole domain warping, rigid or not, which leads to undesirable global sliding.

To prevent this issue and locally improve the warp smoothness, we introduce a regularization step after the constraint projection where the estimates are blended with a low-pass filtered version of themselves using weights dependent on rigidity:

$$b_{i,j} = M_{i,j}((1 - R_{i,j}) + R_{i,j}s) \tag{4}$$

$$p'_{i,j} = (1 - b_{i,j})p_{i,j} + b_{i,j}K_G(p_{i,j}) \tag{5}$$

where i, j are grid coordinates, $R_{i,j}, M_{i,j}$ are the rigidity and mask values at those coordinates, s is a global smoothing factor for as-rigid-as-possible areas, $b_{i,j}$ is a blending factor for the low-pass filtered point and K_G is a 3×3 Gaussian smoothing kernel. The above equation has the effect of smoothing movable non-rigid points near boundaries to rigid areas to create a smooth transition between the original parameterisation and calculated warps.

4.4 Efficient Hierarchical PBD

Regular PBD on a finely tessellated grid converges slowly towards a good solution, as it takes many iterations for local effects to propagate. To counter this, a hierarchy can be used to calculate the warp in a multi-scale manner. Such a hierarchy allows convergence in real-time, as only a few iterations are needed. Müller suggested a hierarchical version of PBD [18], but it did not work well for our problem domain, as the restriction step destabilised the simulation. Below, we describe a simple new efficient hierarchical approach that exploits the regular structure of the simulation grid.

Given a simulation grid with $2^N \times 2^M$ cells, successively lower resolution grids can be constructed by reducing the dimension by half: $2^{N-k} \times 2^{M-k} \; \forall \; k < \min(N, M)$. As the grid has a power-of-two resolution, vertices of all grid levels share the same fine grid:

$$p_{i,j,\ell} = p_{2^{(k-\ell)}i, 2^{(k-\ell)}j, k} \tag{6}$$

where i, j are fine 2D grid coordinates and k, ℓ are hierarchy levels ($k > \ell$), higher being coarser. Constraints for the coarse levels are generated implicitly using equation 6.

The only step that differs from the non-hierarchical version is the constraint projection. At this step, we first simulate the coarsest grid. The results are propagated to the vertices of the next finer level via interpolation and the finer level vertices are smoothed. The process repeats (simulation, propagation, smoothing) until the finest grid has run the simulation.

While the 2D points of the grid are reused for all hierarchy levels, we explicitly construct hierarchies for the rigidity map and warp mask as a preprocess:

$$R_{i.j,k+1} = K_G(R_{i,j,k}) \tag{7}$$

$$M_{i,j,k+1} = \min_{x \in [-1,1], y \in [-1,1]} (M_{i+x,j+y,k}) \tag{8}$$

where $k, k+1$ are hierarchy levels of increasing coarseness, R is the rigidity map, M is the warp mask and K_G is a 3×3 Gaussian smoothing kernel.

One drawback of the hierarchy is that in coarser levels, constraint segments in the parameterisation do not usually map to object-space lines anymore, but curves. As such, we need to apply corrections more conservatively in coarser hierarchy levels.

5. RENDERING

The resulting warps are 2D dense grids that modify the 2D coordinates used to sample *geometry*, using the original coordinates to sample textures. As this is not a typical scenario, the warp typically needs to be inverted (W^{-1}) using any scattered interpolation method; We use a piecewise-linear method by rasterizing triangles on a dense grid using the warped coordinates as positions and carrying the original coordinates as rasterised data. The inversion always exists as the warp is bijective. After the inversion is calculated, the warped UV coordinates are calculated in the following way:

$$uv' = W^{-1}(uv) \tag{9}$$

where eq. 9 evaluates the warp using a "UV redirection" texture.

Example	Simulation Grid Dims	Preprocess	Warp
Face-0	2048^2	1.20	14.12
Face-1	4096^2	4.52	119.31
Face-2	2048^2	1.17	13.92
Face-3	2048^2	1.14	14.04
MugJug	2048^2	1.17	14.05
Blowfish	1024^2	0.37	3.87

Table 1: *Simulation times in milliseconds for various models. From left to right: (1) example model/animation name, (2) warp grid size, (3) Preprocessing time to generate rigity and warp mask hierarchies when any of the two changes and (4) time for simulation of the warp grid hierarchy. As it can be seen, preprocessing time depends purely on the size of the simulation grid. Also, it can be observed that above grid resolutions of 2048^2, performance drops significantly due to VRAM bottleneck.*

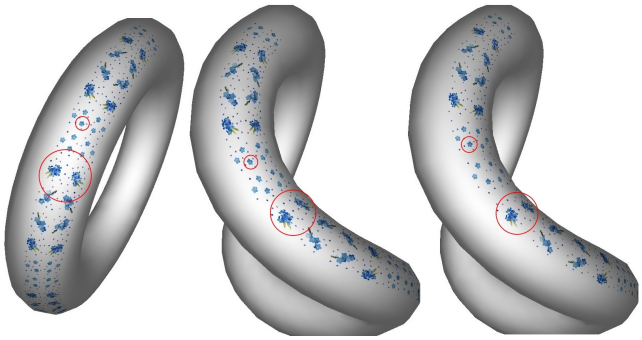


Figure 3: *Comparison of our method with Thin Skin Elastodynamics [15] (sliding variant). The rest pose is on the left, TSE is in the middle and our method is on the right. As it can be seen, our method reduces distortions in a content-aware way, compared to global reduction of distortion in TSE. Our approach is also more stable in terms of animation; a video comparison is provided as additional material.*

6. RESULTS AND ANALYSIS

In this section our algorithm is evaluated in a number of challenging cases. It demonstrates behaviour and performance on meshes of different complexities to highlight the effectiveness of our approach. We validated the algorithm on a number of models textured using a unique parameterisation. Experiments are carried out on an Intel Xeon X5550 CPU and an NVidia GFX Titan GPU. Figure 7 shows example deformations on our Face, Blowfish and T-Rex models. In all cases our approach highlights the preservation of texture shape and structure under deformation. In figure 6, our example highlights deformation on the MugJug model. In this case, successive deformations from an interactive editing procedure would usually distort the texture map. However, our method corrects this, removing the additional artist overhead of re-creating the texture map in the final approved model.

Figure 5 shows how different features, feature distribution and rigidity variation affect the warp calculation on a basic deformation. In extreme compression it can be observed that shape is not fully preserved. This happens because our method re-parameterises a bounded domain, therefore it is not possible to expand the features in texture space infinitely. Additionally, it can be observed that compression starts before all of the available non-rigid space has been used. This is a tradeoff of our method as it enforces stability and warp smoothness over using all available non-rigid space. In this example of severe compression, we apply a heuristic to determine the spatially-varying "orientation preservation" value over the domain. The heuristic assigns a high value in dense-rigidity areas (top of texture) and a low value in sparse-rigity areas (bottom of texture); this has the effect of of better preserving the shape in sparse areas while preventing compression problems in dense ones.

Example	Normal Rendering	Warp	
		Warp inversion	Rendering
MugJug	4.44	3.84	4.54
Blowfish	5.71	1.11	5.88
Face0	5.05	3.85	5.32
Face1	5.10	16.05	5.34

Table 2: *Rendering times (milliseconds). Columns 2 and 4 display the time required to render the scene, without and with applying the warp respectively. Column 3 shows the time required to invert the warp.*

The T-Rex example (fig. 7, bottom row) shows the application of our method in a production workflow. In such cases, several UV tiles can be used for a single model (e.g. UDIM tiles). As the chart boundaries are constant, we process and remap each tile independently, therefore scaling to any number of tiles. While our example uses as a proof of concept a total of six tiles, four out of which containing rigidity data ($2K$ resolution each), our method scales to any number of tiles and any hardware-supported tile size.

Our approach was also compared with other standard and state-of-the-art methods that match or minimise texture distortion under 3D model deformation.

In figure 4 we use the MugJug example to compare our approach against Koniaris et al. [13], Sheffer and De Sturler's method [24] and Mean Value coordinates. In all of these comparisons, our approach preserves local structure under artistic editing better. This is especially noticeable in regions undergoing large but local distortions, such as the center of the mug/jug.

In figure 3 we show a comparison against the leading method of Li et al. [15] using the deforming torus animation dataset. Our approach reduces distortions introduced by the deformation in salient regions better, and displays better temporal coherence.

Our accompanying video shows related deformation results for the Face, T-Rex, Blowfish, Torus and MugJug animated models. Corrected cases at selected deformation frames are highlighted to demonstrate the differences. From a qualitative view point, our animations visually reduce the introduced deformation in rigid areas, better preserving the intended structure and shape of the applied texture versus current state-of-the-art approaches.

Table 1 shows the dense warp calculation results for all the demonstrated models. As can be seen from the simulation times, the fast distortion correction time allows edits to be processed in real-time. The total simulation times for each model occur only when mesh vertices are modified simultaneously in a frame, and so they can be regarded as worst-case simulation times. Each simulation iteration involves running approximately 70 GPU shader passes, most passes being constraint projection of a subset of the constraints that can be safely run in parallel. As such, while using the hierarchical approach significantly increases the convergence of the algorithm, an increase of the dense grid resolution has a non-neglibible shader pass overhead, as can be seen by the performance difference of the two T-Rex entries. As the process is fully in GPU memory, including transfer of edge data, there is no CPU-GPU data transfer overhead during the simulation.

We compare rendering performance of a textured, deforming model with and without the use of the calculated warps. The measured rendering performance was averaged over several frames. The rendering involved a close-up of the model, using a basic lambert material using the (warped or not) rigidity map as the diffuse texture.

Table 2 shows the performance cost of evaluating the warps. While the cost of rendering the dense warps is small, an inversion of the warp is required first, adding to the total required time for rendering.

Figure 4: *Comparison of parameterisation methods using the MugJug model. The rest pose is the rightmost image; a mug. The deformed frames are shown from left to right: re-parameterisation using our method, Koniaris et al. [13], Sheffer and De Sturler[24], Mean Value Coordinates, and using the original UVs. Our method is the only one that preserves the features of the mug at the right scale. A video comparison is provided as additional material.*

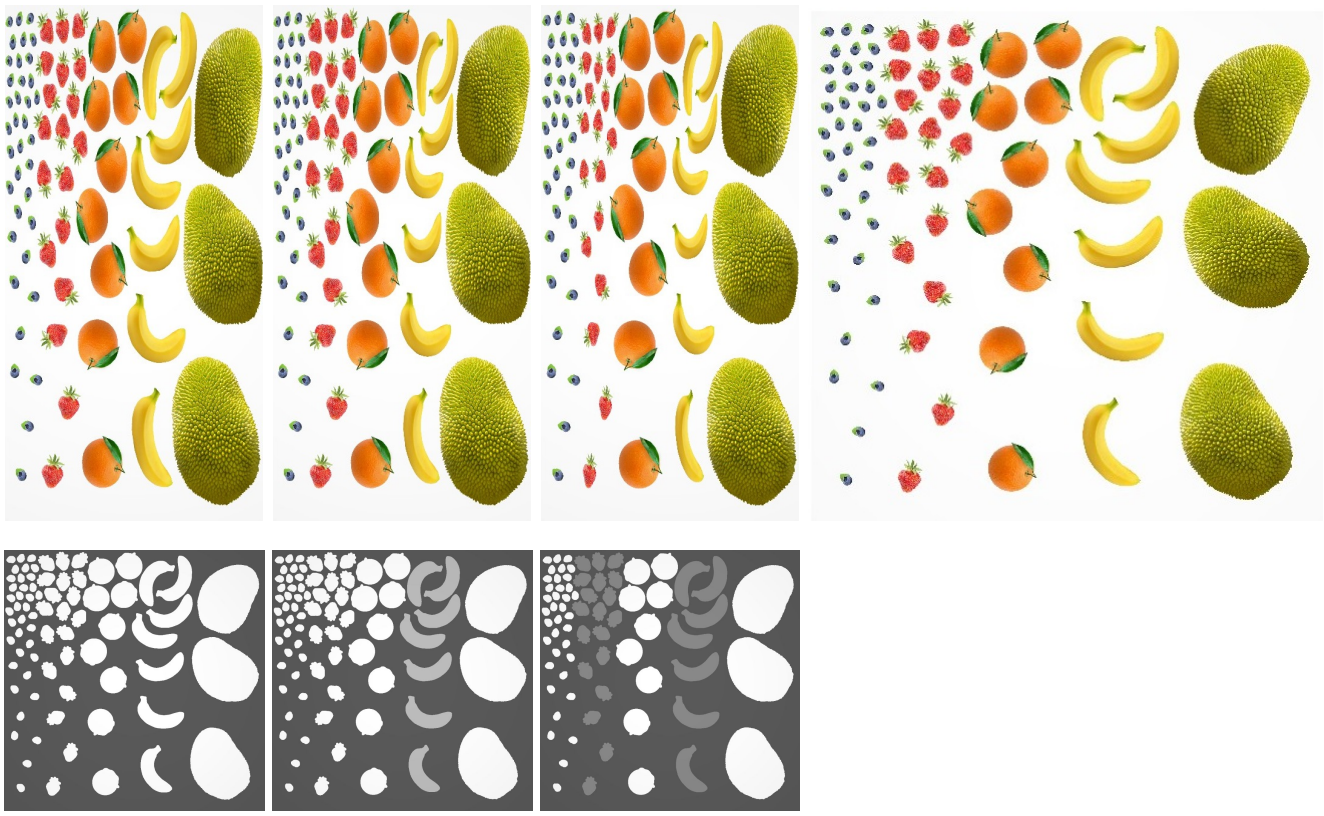


Figure 5: *A horizontally compressed square (rest-pose: right), demonstrating the effect of variable rigidity (bottom row) under deformation: 100% (left - all fruit), 50% (middle-left, bananas) and 25% (middle-right - bananas and strawberries) rigidity. As the compression is fairly strong and the rigid element distribution is dense, there is not enough non-rigid space in the parameterisation domain to warp in order to preserve the size of rigid features. In that case, rigid elements still compress, although more rigid elements compress less than less rigid ones. Additionally, our rigidity-density heuristic maintains the shape of features in sparse-rigidity areas (bottom of texture) better compared to more dense areas (top of texture).*

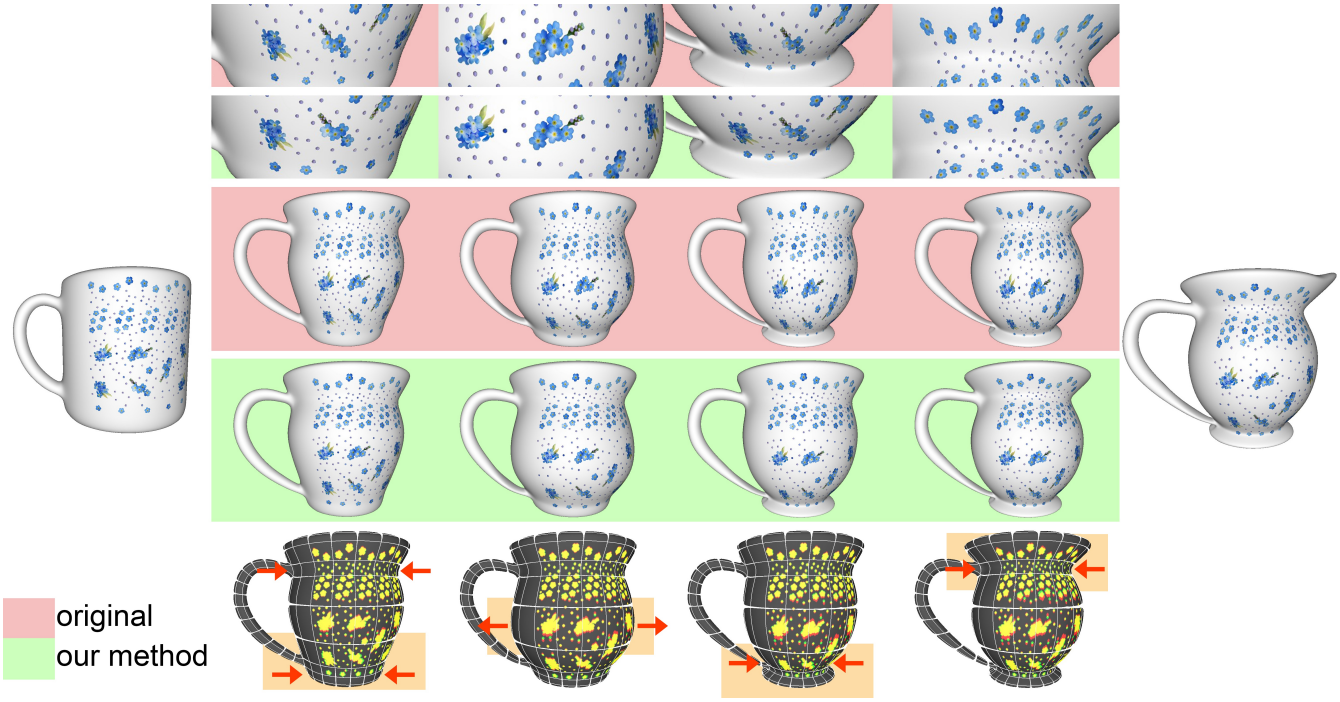


Figure 6: *In this example we highlight the benefits of our method in an artistic work flow. An initial model (left) is created and then successively edited by an artist (red arrows). At each edit (left to right), the original texture detail (middle row) is distorted, moving away from its initial desired shape (left). Our method (middle-bottom) preserves the textures original shape, saving the artist valuable time in manually creating a new undistorted texture after each edit. On the bottom, visualization are shown for rigid areas in the original (red) and warped (green) parameterisations, including areas where they overlap (yellow). The highlighted areas focus on specific parts of the texture under distortion and correction and shown more closely in the top and second from top rows. The final model corrected with our method preserves texture shape and size (right image).*

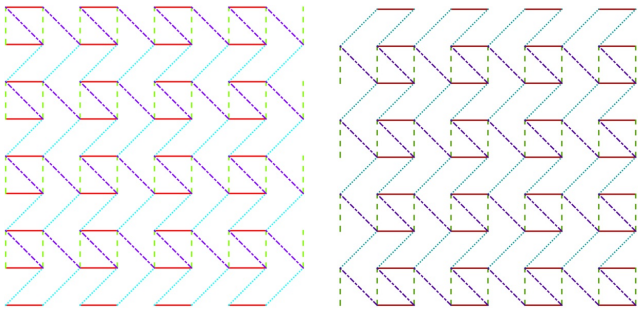


Figure 8: *Parallel sets of edge constraints (sec. 4.2) shown with unique combinations of line style and color. There are eight disjoint sets – we display four per figure for clarity.*

7. IMPLEMENTATION DETAILS

The PBD process was implemented in C++/DirectCompute, using the GPU for calculating the individual steps. Calculation of estimates, constraint projection, integration, as well as the in-between hierarchy levels interpolation and smoothing are all implemented as shader passes. At each deformation frame, the object-space positions are rendered to a texture, which is used by the simulation. The simulation results are rendered to a "UV redirection" texture using a single shader pass.

In constraint projection, the constraints are organised in parallel sets, where the vertices of a constraint in the set are not used by any other constraint in the same parallel set. The parallel sets are expressed programmatically in order to use minimum storage, and are shown in figure 8.

In addition to the per-point/constraint parallelization, we also partition the parameterisation domain into a 16×16 grid, where each of its cells is executed as a single threadgroup, thus improving the performance via better use of the GPU hardware.

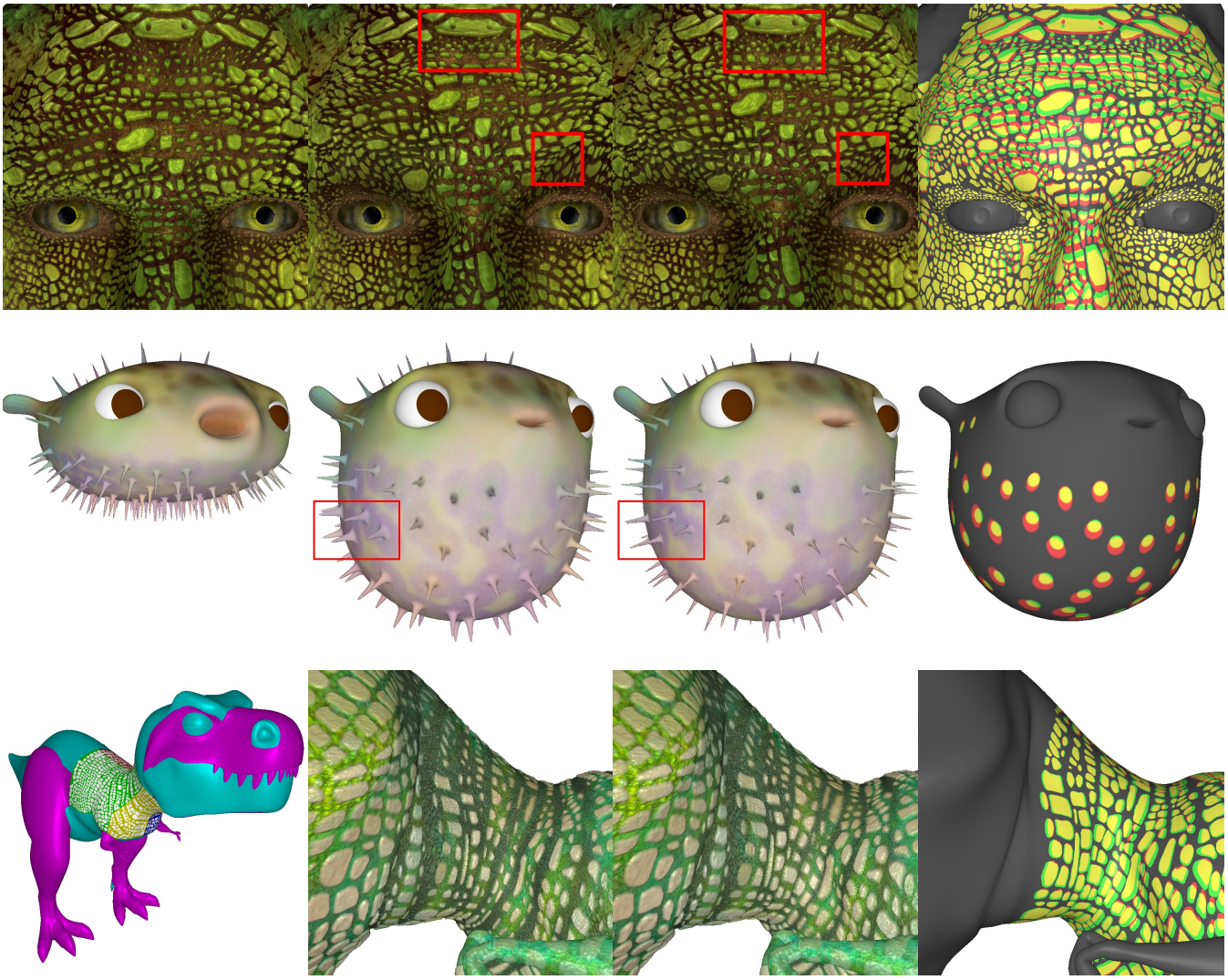
8. SUMMARY

In this paper we presented a technique to re-parameterise texture space of an animated model in real-time, such that important rigid features mapped on these regions are preserved when the surface deforms. The warp behavior is driven by a rigidity map and a warp mask, which can be modified dynamically (as the surface deforms) at a small additional cost.

Comparisons with existing state of the art methods also show our method to better preserve local texture deformation – minimising distortions and artefacts introduced by coarse surface deformation.

The technique can be applied to reduce elastic distortions in animated texture-mapped surfaces that represent materials with heterogeneous deformation properties. We have demonstrated such application to several use cases. In offline production, the technique can be used to warp the parameterisation of selected high-resolution fine-scale texture layers with heterogeneous elastic properties (e.g. leather cracks or scales). In modeling, artists can deform and morph object preserving the shape and size of selected texture layers. In real-time digital acting, a facial performance of an actor can be retargetted to a creature with heterogeneous texture detail, different to the source actor. In video games, different animated characters can share the base model and animations while warping parameterisations according to their unique textures.

We would like to extend the constraints of the simulation, so that we can apply the algorithm for different use cases that would benefit from post-deformation re-parameterisation, e.g. bidirectional texture functions or polynomial texture maps. We would also like to extend the algorithm to perform content-aware warp of the vol-



(a) Rest pose **(b)** Original **(c)** Corrected **(d)** Difference

Figure 7: *Example deformations for various datasets (Face2, Blowfish, T-Rex). The face and T-Rex textures contain stone/bone-like bumps which we wish to remain rigid. The blowfish has spikes mapped as displacement, which are marked as rigid. Uncorrected (red) and corrected (green) mapping of features are shown in (d). Corrected and uncorrected rigid features that overlap are displayed in yellow.*

umetric space (thick shell over the surface), so that detail of any complexity can be preserved under deformation.

Acknowledgements

We would like to thank Malgorzata Kosek for textures and the Blowfish and MugJug models, Martin Klaudiny for the face model, Stephen Grabli and Jose A. Iglesias-Guitian for insightful feedback and comments. T-Rex model is courtesy of Walt Disney Animation Studios. This work was funded by Innovate UK, project 101587.

9. REFERENCES

[1] B. Burley and D. Lacewell. Ptex: Per-face texture mapping for production rendering. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR '08, pages 1155–1164, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[2] I. Chao, U. Pinkall, P. Sanan, and P. Schröder. A simple geometric model for elastic deformations. *ACM Trans. Graph.*, 29(4):38:1–38:6, July 2010.

[3] E. Dekkers and L. Kobbelt. Geometry seam carving. *Comput. Aided Des.*, 46:120–128, Jan. 2014.

[4] R. Falco and H. Driskill. Inverse texture warping. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, pages 206–206, New York, NY, USA, 2002. ACM.

[5] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling*, pages 157–186. Springer, 2005.

[6] R. Gal, O. Sorkine, and D. Cohen-Or. Feature-aware texturing. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR '06, pages 297–303, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[7] S. Grabli, K. Sprout, and Y. Ye. Feature-based texture stretch compensation for 3d meshes. In *ACM SIGGRAPH 2015 Talks*, pages 71:1–71:1, New York, NY, USA, 2015. ACM.

[8] P. Heckbert. Survey of texture mapping. *Computer Graphics and Applications, IEEE*, 6(11):56–67, Nov 1986.

[9] K. Hormann, K. Polthier, and A. Sheffer. Mesh

parameterization: Theory and practice. In *ACM SIGGRAPH ASIA 2008 Courses*, SIGGRAPH Asia '08, pages 12:1–12:87, New York, NY, USA, 2008. ACM.

[10] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine. Fast automatic skinning transformations. *ACM Trans. Graph.*, 31(4):77:1–77:10, July 2012.

[11] Y. Jin, Z. Shi, J. Sun, J. Huang, and R. Tong. Content-aware texture mapping. *Graphical Models*, 76(3):152 – 161, 2014. Computational Visual Media Conference 2013Second Computational Visual Media Conference (CVM).

[12] P. Kaufmann, O. Wang, A. Sorkine-Hornung, O. Sorkine-Hornung, A. Smolic, and M. Gross. Finite element image warping. *Computer Graphics Forum*, 32(2pt1):31–39, 2013.

[13] C. Koniaris, D. Cosker, X. Yang, K. Mitchell, and I. Matthews. Real-time content-aware texturing for deformable surfaces. In *Proceedings of the 10th European Conference on Visual Media Production*, CVMP '13, pages 11:1–11:10, New York, NY, USA, 2013. ACM.

[14] V. Kraevoy, A. Sheffer, A. Shamir, and D. Cohen-Or. Non-homogeneous resizing of complex models. *ACM Trans. Graph.*, 27(5):111:1–111:9, Dec. 2008.

[15] D. Li, S. Sueda, D. R. Neog, and D. K. Pai. Thin skin elastodynamics. *ACM Trans. Graph.*, 32(4):49:1–49:10, July 2013.

[16] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1495–1504, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[17] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 27–34, New York, NY, USA, 1993. ACM.

[18] M. Müller. In F. Faure and M. Teschner, editors, *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)*. The Eurographics Association, 2008.

[19] M. Müller and N. Chentanez. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 85–92, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[20] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, Apr. 2007.

[21] T. Popa, D. Julius, and A. Sheffer. Interactive and linear material aware deformations. *International Journal of Shape modeling*, 2007.

[22] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir. A comparative study of image retargeting. *ACM Trans. Graph.*, 29(6):160:1–160:10, Dec. 2010.

[23] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering*, EGRW '02, pages 87–98, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[24] A. Sheffer and E. de Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Trans. Graph.*, 21(4):874–890, Oct. 2002.

[25] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, Jan. 2006.

[26] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[27] R. Vaillant, G. Guennebaud, L. Barthe, B. Wyvill, and M.-P. Cani. Robust iso-surface tracking for interactive character skinning. *ACM Trans. Graph.*, 33(6):189:1–189:11, Nov. 2014.

[28] D. Vaquero, M. Turk, K. Pulli, M. Tico, and N. Gelfand. A survey of image retargeting techniques. volume 7798, pages 779814–779814–15, 2010.

[29] Y. Wang, A. Jacobson, J. Barbic, and L. Kavan. Linear subspace design for real-time shape deformation. *ACM Trans. Graph.*, 34(4), 2015.

[30] X. Yang, R. Southern, and J. J. Zhang. Fast simulation of skin sliding. *Computer Animation and Virtual Worlds*, 20(2-3):333–342, 2009.

[31] J. Zhang, B. Deng, Z. Liu, G. Patanè, S. Bouaziz, K. Hormann, and L. Liu. Local barycentric coordinates. *ACM Trans. Graph.*, 33(6):188:1–188:12, Nov. 2014.