STAG: Smart Tools and Applications in Graphics (2018)
M. Livesu, G. Pintore and A. Signoroni (Editors)

Fast centroidal deformation for large mesh models

A. Morsucci, M. Centin and A. Signoroni[†]

Information Engineering Dept., University of Brescia – Brescia, Italy

Abstract

We present an algorithm that allows fast non-linear deformation editing on high-quality meshes. The proposed Fast Centroidal Deformation (FCD) method is based on a multi-resolution framework, where a centroidal deformation graph is built over the mesh in order to allow fast non-linear optimization at a coarse scale. The resulting deformation is then propagated to the initial dense mesh by exploiting the relationship between the constructed deformation graph and the input mesh through a mapping function that unifies local rotations and global translations without the need of solving a system composed by a number of linear equations of the same magnitude of the number of vertices of the mesh. A number of flexible user constraints can be imposed in the deformation through a handle-based metaphor where the user can redefine the position and orientation of single control points or entire portions of the input model. The proposed method addresses the obstacle of non-linear deformation on meshes composed by millions of vertices and is compared with the reference deformation techniques, showing significant improvements in terms of computational efficiency without renouncing to the quality of the results given by non-linear methods.

CCS Concepts
•***Computing methodologies*** $\rightarrow$ ***Mesh models;***

1. Introduction

Mesh deformation is a relevant, long-time investigated, and still active research field in Geometry Processing. Many applications require, as critical component, means to deform existing or acquired 3D models, often represented by mesh data structures. Creative editing and 3D characters animation are among the most obvious and popular application fields [JT05, NMK*06, BP07], but other at least as important domains call for high quality deformation tools in action, for example in medical procedures [CDA99, KR15] or in industrial design and digital manufacturing processes [MYF07, YLL*16, SGA*16]. This is especially true where adjustments or modifications are required on 3D models coming from some kind of 3D digitization/scanning of real scenes, objects or subjects. Also the growing metric performance of modern 3D scanners de facto raises the bar of the application requirements that must be fulfilled and require the development of advanced techniques capable of robustly and efficiently handle deformations on large and detailed models without spoiling their quality.
Most of the mesh deformation methods are based on the usage of structures (skeleton or cages) external to the mesh surface. They are referred to as *space-based* techniques, where the construction of guiding structures usually requires time and effort. This can be well justified in contexts like 3D animation, where models (e.g. human or animal bodies and faces) have to be deformed in many different poses and defined motion schemes (e.g. human skeleton driven movements). In these specific application contexts it is good to have a system to describe and control the motion. However, this typically requires a specific and time-consuming preliminary work by expert users. In this work we are instead interested in generic methods that usually find solutions in what is referred to as *surface-based* techniques, that directly work on the mesh surface and that can be driven by simple user interaction paradigms. Moreover, we especially need fast execution times of the deformation algorithms in order to preserve a good degree of user interactivity, an aspect that represents a critical and open issue for surface-based techniques, while for space-based techniques is tackled by exploiting the priory knowledge provided by the external structures.
In this work, we propose a general purpose mesh deformation technique that exploits a guiding structure that does not require a design phase, since it is automatically created with the sole purpose (typical of the surface-based methods) to drive a coarse deformation phase followed by a proper handling of the finer details.
Another important distinction among deformation methods is related to computational approaches, where *linear* techniques are faster but suffering from some limitations especially in their capability to correctly handle large deformation [BS08]. Instead, *non-linear methods*, i.e. requiring the solution of non-linear equation systems, typically coming from non-convex optimization problems, provide more realistic and physically plausible results, but at the price of a significantly increased computational cost, that easily becomes prohibitive when mesh dimension grows.

[†] Corresponding Author

DOI: 10.2312/stag.20181302

In this work, we want to manage large deformations, therefore we need a non-linear method, but at the same time we want to achieve computational performance compatible with a fluid user-interaction and an on-the-fly definition of the deformation we want to provide even on large mesh models. This is obtained by applying the deformation only to the automatically generated control structure and, in a final step, by transferring the deformation over the finer details by a suitable mapping.

1.1. Related Work

Early mesh deformation techniques, such as the one of Sederberg and Parry [SP86], can be considered space-based, with the advantage of being able to keep computational complexity under control by working on simplified structures. The method presented by Botsch et al. [BPWG07], based on rigid cell, is an example of a more recent technique that produces high quality space deformations. Another approach by Tao et al. [JSW05] efficiently maps the external structure to the mesh by using a generalization of Mean Value Coordinates. One main problem of these techniques is the need to build a lattice around the mesh in order to enable fine detail preservation. High detailed mesh can require cumbersome user interaction to achieve good results. Other early techniques were based on structures (skeletons) inside the model which gave birth to the *skinning* methods (see Magnenat-Thalmann et al. [MTLT88]). This type of deformation evolved into what are the most diffuse techniques in animation applications, giving realistic results [JT05]. Unfortunately, the main problem of this technique is the difficulty to automatize the process of skeleton construction and vertex weight estimation. Surface mesh deformation techniques introduced a new way to face this problem: regions or points over the mesh are set to be constraints to which a transformation is applied. The *region of influence* (ROI) defined on the mesh is deformed starting from these constraints. Methods using this approach can be subdivided in two groups: one that uses interpolation functions or *Radial Basis Functions* (RBF) (M. Botsch and L. Kobbelt [BK05]; de Boer et al. [dBvdSB07]), the other that minimizes energy defined on the mesh, computing the whole deformation starting from the fixed regions (Botsch and Kobbelt [BK04], Sorkine et al. [SCOL*04], Botsch et al. [BPK*07]. Surface mesh deformation has the great advantage to work directly on the mesh, preserving every detail, but with an increased computational cost.
For models coming from 3D scanning systems, especially those producing large and high quality meshes, deformation with detail preservation is a primary goal. Multiresolution approaches (Forsey and Bartels [FB88], Zorin et al. [ZSS97], Kobbelt et al. [KCVS98], Guskov et al. [GSS99], Lipman et al. [Lip04]) tackled the deformation problem separating high frequency details from the low frequency shape. Other techniques, instead, apply a deformation transfer from the coarse mesh to the fine one, preserving all the small details (Sumner and Popović [SP04], Botsch et al. [BSPG06], Moser [MCR16]). In the first case, the methods behave very well if the details we want to preserve can be expressed as a height map of the coarse mesh; if this assumption does not hold, the algorithm can lead to unsatisfying results. In the second case, transferring deformation form a mesh to another achieves great results avoiding many artifacts, due to the fact that deformation is applied directly on the detailed mesh. Deformation transfer is done solving a sparse linear system, and this can be done quite efficiently by using modern sparse solvers (Botsch et al. [BBK05]). Unfortunately, this efficiency decays when mesh dimension starts growing: even if the solution can be achieved through only one step of factorization, the computation cost of this step is too expensive for large datasets.
Moreover, even if real-time deformation could be achieved with multiresolution techniques or deformation transfer, linear methods cannot provide the best results from the point of view of realism. In fact, high quality large deformation can be obtained only by using non-linear deformation techniques. Many non-linear methods have been proposed providing high quality results. In the PriMo method, proposed by Botsch et al. [BPGK06], a prism-based structure over the mesh is constructed, where each prism is connected with its neighborhood by elastic forces. The transformation of every prism is computed by minimizing the elastic energy and then the deformation is transferred to the mesh surface. As-rigid-as-possible (ARAP) deformation, based on non-linear energy minimization, was proposed by Sorkine and Alexa [SA07] with the aim to provide robust and physically plausible deformation results. This technique fails to achieve satisfactory results for certain type of deformations and has been recently improved by Levi and Gotsman [LG15] with a technique called Smooth Rotation enhanced ARAP (SR-ARAP), where the deformation energy is extended with a further factor that penalizes too different rotations of close vertices, significantly increasing the quality of the final result in many situations. In Huang et al. [HSL*06] another non-linear energy term is defined that takes into account volume, projection and skeleton constraints and the final result is obtained through a least-square minimization of this energy. Other interesting approaches are the one proposed by Shi et al. [SZT*07], where a different way to approach skinning deformation is proposed, and by Chao et al. [CPSS10], where an elastic energy is exploited to generate physical plausible deformations.
The main disadvantage of non-linear methods is the high computational cost, that increases drastically when the number of vertices grows. To overcome this problem, many techniques based on spatial reduction have been presented. These methods provide tools to compute subspace of the mesh deformation problem, giving a direct relationship between the full space and the subspace. The designed subspace can be a coarser version of the original mesh or a subset of the possible deformations (Wang et al. [WJBK15], Barbič and James [BJ05], Hildebrandt et al. [HSTP11], von Radziewsky et al. [vRESH16]). Even if these methods are based on a very solid mathematical basis and provide high quality results, they require a lot of pre-computation to actually be fast during the deformation session. This computational complexity is a bottleneck of most of these applications when large meshes (e.g. over 1M vertices) must be handled. Even techniques that require just one pre-factorization of a sparse matrix (such as [BBK05] or [WJBK15]) are not fully suitable for the purpose. Sieger et al. [SGA*16] exploits a Moving Least Square (MLS) technique in order to reduce the dimensionality of the deformation problem, computing a basis for each control point; unfortunately however, this technique becomes impracticable for large meshes.
Sumner et al. [SSP07] define an embedded deformation, in order to link different data types (meshes, point clouds, particles, ...) to theirs coarse versions. This allows directly transferring of the deformation from a coarse graph to the fine mesh. This work, unfortunately, does not provide a reliable mesh deformation technique,

introducing artifacts for large deformations, such as pinching phenomena. The idea of using a coarser scale graph of patches over large meshes to propagate deformations in a smooth way was recently used also in [PBGC18] in a non-rigid deformation context within a technique similar to [BSB14].

1.2. Contribution

In this paper we present a non-linear space-based deformation technique. The construction of the control structure, called *deformation graph* is completely automatized (Sec. 2.1), with no need of interaction by the user: this procedure overcomes one of the most important disadvantages of space deformation techniques. Where many other methods fail to do so, our method can apply a non-linear deformation to mesh of millions of vertices in a very small amount of time, since it avoids requiring matrix factorization for solving sparse linear system with dimensions of the order of the number of vertices. This boosts the performance of both pre-processing and actual mesh deformation (Sec. 2.2). Thanks to the presence of the structure, our method does not require to directly act on finer local details on mesh surface, but, thanks to the relationship between graph and mesh, it can map coherently a global deformation applied to the graph to the mesh surface: it does not try to separate different *frequencies*, but can express and map mesh detail as a weighted function of the deformation over the graph control points (Sec. 2.3).

2. Proposed method

The description of the proposed method based on Fast Centroidal Deformation (FCD) follows the workflow depicted in Fig. 1 where we show the various phases described in the following subsections along with a demonstrative result of application of the method. The first and second steps (described in Sec. 2.1) allow the automatic construction of the reference deformation graph. The third step (Sec. 2.2) is responsible for a non-linear deformation applied the reference graph. The fourth, and last, step (Sec. 2.3) performs the mapping of the original mesh on the deformed graph by producing the resulting deformed mesh. The deformation works according to a handle-based interaction, where handles can be one single graph point or a set of points selected by the user, which has also the task of providing a rigid transformation of the handles. This is typically accomplished with the help of proper interaction tools for the definition of translation, rotation around an axis and their combinations. The user can even choose the region of influence of the deformation, deciding which vertices of the original mesh can be moved. Thanks to our FCD based on an automated reference structure (graph) generation, we enable highly interactive and responsive editing sessions, with users having the possibility to ignore any disturbing underlying complexity.

2.1. Deformation graph generation

We aim to the automated generation of a graph derived from the original mesh. This should act as a guiding structure (in this sense our method can be classified space-based) by providing control points from which the deformation can be computed and then transferred to the mesh. This graph structure is made by undirected edges and can be non-manifold and made of multiple connected components. The only requirement needed for our method to work is the knowledge of graph connectivity: this allows exploiting surface-based mesh deformation techniques that require the knowledge of vertices' neighborhood. Moreover, we want the graph generation to be effective and fast even on large mesh models and we also want to have the control on the simplification level we want to obtain (which can be linked to a metric of the graph).
The generation and exploitation of a graph in the context of embedded (space-based) mesh deformation was proposed by Sumner *et al.* [SSP07]. In their work a uniform mesh subsampling is operated where the deformation graph connectivity is defined on the sampled points according to how control points influence mesh vertexes in the final deformation mapping. However, this can easily generate topological changes of the graph with respect to the original mesh. This is why we here adopt a topology aware graph generation able to properly propagate connectivity information within the mesh simplification process. The construction of our deformation graph is similar to the one proposed by Cohen-Steiner *et al.* [CSAD04] where we also build on a k-means formulation [Mac67,Llo82] while differing by the use of euclidean distance metric for surface partitioning, by allowing the obtained graph to be non-manifold and by introducing a mechanism to adapt the number of generated graph nodes based on a user-defined metric parameter.

Given a subset $\mathcal{K}_0$ of a predefined number k_0 of points randomly selected from the set $\mathcal{M}$ comprising all the vertices of the input mesh, we start considering the points in $\mathcal{K}_0$ as centers of the patches, and we associate every point in $\mathcal{M}$ to a patch according to a point-to-point *proximity distance* criteria. In our case, we consider the Euclidean distance between the mesh point and its candidate patch-center. In concrete, patches grow considering mesh connectivity with a prioritized queue (to prevent creation of interferences and not connected regions). Then an iterative patch splitting and adjustment starts which is driven by a user defined distance D (reflecting a simplification strength parameter). This parameter is used to split a patch if one point is found to be at a distance greater than D from its center. This allows the user to metrically guide the subsampling without having to be constrained to define a fixed and more abstract patch number k_0. Moreover by defining a value of D lower than what can be expected to be the average length of the graph induced by the random subsampling, i.e. $D_0 \approx d_{\text{avg}} * \sqrt{(m/k_0)}$ (where m is the number vertices and d_{avg} the average edge length of the original mesh), we can produce a more uniform subsampling with respect to the purely random one, and this is beneficial to our method. The rule that governs patch splitting related to the introduction of the metric bound D is that every patch can be possibly split only once at each iteration. This avoids generating centers too close to each other. The algorithm iterates by updating all centers to the patch barycenter (assigning as new center its closest mesh vertex). At iteration n the updated set of centers $\mathcal{K}_n$ undergo the described split and adjust process until convergence is reached with a final number of patches k_N.
During the above shape simplification process, the graph is simultaneously extracted thanks to the intrinsic possibility to keep tracked the connectivity relations among the different patches subdividing our mesh. More specifically the adjacency matrix $\mathcal{A}$ is com-

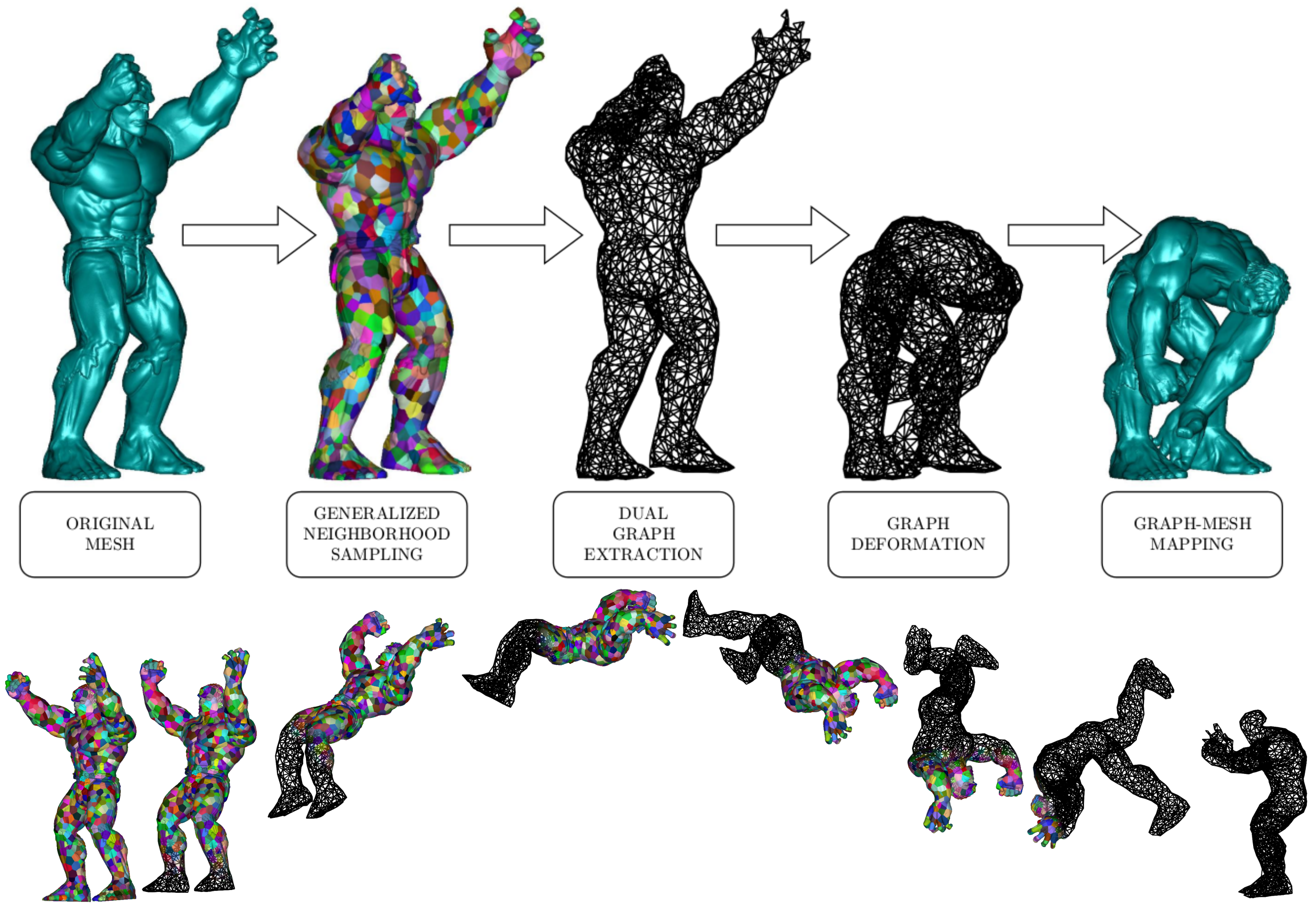


Figure 1: *Workflow and application of our method: Hulk (1.4M vertices) needs some stretching (top raw), before doing gym (bottom raw).*

puted and updated at every iteration and the deformation graph $\mathcal{G}$, the one used in the deformation process, is defined as the *dual graph* of the sampled mesh. This way, every center of each patch (the sampled vertices) is a node of the graph and the edges of the graph are the connections between adjacent centers. The edges are undirected and this type of construction can lead to non-manifold graphs, against which our method is robust.
In Figure 2 examples of our sampling method are shown, where every patch has been colored with a different color and where the main parameters related to the graph generation are specified along with timings. Our method allows a rapid convergence to a mesh partitioning reflecting the desired patch dimension, expressed by the parameter D, even starting from large and dense meshes. Convergence behavior is similar to the typical one of moving means clustering since, once the patch splitting induced by D is completed, the method evolves as a pure k-means, therefore converging to a local minima in a finite number of steps by minimizing a cost function E of the following nature:

$$E = \sum_{c \in \mathcal{K}} \sum_{p \in \mathcal{P}_c} ||p - c||^2 \qquad (1)$$

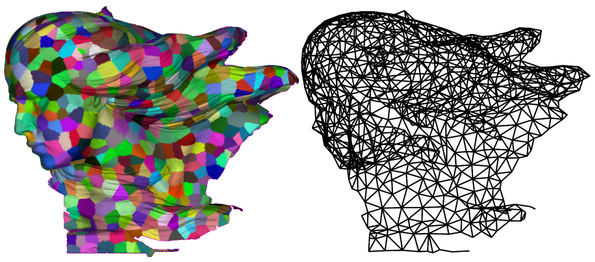
with $\mathcal{P}_c$ is the patch associated to a center c.

As demonstrated in Figure 2, our method is able to reach convergence in very few steps while allowing to obtain a near-uniform subsampling according to a desired level of simplification and reflecting the mesh structure according to the desired level of detail.

2.2. Non-Linear deformation

Among mesh deformation methods able to work starting from the definition of positional constraints (deformation handles), the as-rigid-as-possible (ARAP) approach by Sorkine and Alexa [SA07] gained popularity. The original ARAP formulation is based on the minimization of an energy term that takes into account non-rigid distortions in the 1-ring of the mesh vertices. An evolution of this method has been proposed by Gotsman and Levi [LG15], where a smooth rotation SR-ARAP solution was aiming to solve both computational and deformation quality drawbacks of the original ARAP surface deformation approach. However, also in this case, efficiency issues remains if the method is applied to large mesh models. In this work, we adopt an evolution of SR-ARAP deformation which is adapted to be used on generic graphs, as the *deformation graphs* described in Sec. 2.1. The deformation energy is

	Description	Hurricane	Dragon
m	number of vertices	214627	3609455
$d_{avg}[mm]$	mesh average edge length	1.16	0.11
k_0	number of initial seeds	300	300
$t_0[ms]$	initialization time	14	441
$D[mm]$	target patch radius	19.13	16.34
k_1	n. of seeds at 1st iteration	548	481
$t_1[ms]$	time for 1st iteration	2203	5505
N	total number of iterations	5	6
k_{N+1}	final number of seeds	779	547
$t_{N+1}[ms]$	time for last iteration	270	6198



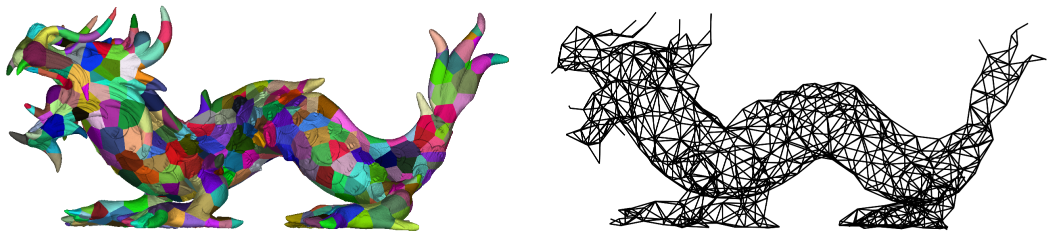


Figure 2: *Examples of our sampling algorithm, generated non-manifold deformation graphs and main parameters summarized in the table.*

defined as a sum of weighted rotated edge differences plus a penalty coefficient used to promote smooth rotations.

Given $\mathcal{K}$ the set of the k control points on the graph $\mathcal{G}$ and defining e_{ij} as the edge connecting the control points c_i and c_j on the original graph and e'_{ij} as the edge connecting the control points c'_i and c'_j on the deformed graph, the energy function defined on the graph is the following

$$E(\mathcal{G}) = \sum_{i\in\mathcal{K}} \sum_{j\in\mathcal{N}(i)} w_{ij}||e'_{ij} - R_i e_{ij}||^2 + \alpha A||R_i - R_j||_F^2 \qquad (2)$$

with R_i and $\mathcal{N}(i)$ respectively the local rotation matrix and the neighborhood (1-ring) associated to the i-th control point, w_{ij} the Laplacian weight of the ij-th edge of the graph and $||\cdot||_F$ the *Frobenious Norm* (which was chosen in [LG15] to derive a proper discretization of the continuous SR-ARAP formulation and to consistently promote small rotations). The coefficient α is a penalization weight that is scaled by the total mesh area A, in order to prevent scale variance. In the original definition of SR-ARAP the weights w_{ij} are the Laplacian coefficients computed on the mesh surfaces. Our method applies this algorithm on the deformation graph, that can be a generic graph, even not manifold neither watertight. To be able to apply the SR-ARAP algorithm on a generic graph we derive the w_{ij} weights from its Laplacian matrix. Given the adjacency matrix $\mathcal{A}$, that tracks graph connectivity, defined as

$$\mathcal{A}_{ij} = \begin{cases} 1 & \text{if } c_i \text{ is adjacent to } c_j \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

the Laplacian matrix $\mathcal{L}$ of the graph is computed as $\mathcal{L} = \mathcal{D} - \mathcal{A}$, where $\mathcal{D}$ is the degree matrix of the graph, computed as

$$\mathcal{D}_{ij} = \begin{cases} deg(i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

where $deg(i)$ is the number of edges incident to the i-th vertex. Using this particular type of weights, we don't rely any more on the surface characteristics, allowing our deformation technique to be completely transparent to any defects on the mesh.

In order to find the deformation graph $\mathcal{G}$ that minimizes the energy in (2), we need to minimize the local rotations of all edges, trying to achieve the most rigid transformation possible. To do so, the energy must be minimized according to both its degrees of freedom: the new vertex positions and the local rotation matrices. The process of solving this non linear problem is composed by an iterative procedure subdivided in two steps.
In the first step, called *local step*, the energy is minimized with respect to the local matrix rotations, keeping the new vertices positions fixed. Defining the matrix

$$S_i = -2\left(\sum_{j\in\mathcal{N}(i)} w_{ij}(e_{ij}e'^T_{ij} + \alpha A R_j)\right) \qquad (5)$$

and its SVD decomposition $S_i = U_i \Sigma V_i^T$, for each control point c_i its minimal local rotation matrix R_i is computed as $R_i = V_i U_i^T$. On a generic matrix, SVD is an expensive operation: luckily in this case S_i are 3x3 matrices and the decomposition can be efficiently computed.
In the second step, called *global step*, the energy is minimized with respect to new vertices positions, keeping fixed the rotations computed in the previous step. This minimization ends up with solving a sparse linear system with $k - k_c$ equations and $k - k_c$ unknowns (with k_c number of control points which positions have been determined by the user, i.e. handle and fixed nodes). The system to solve is then

$$\sum_{j\in\mathcal{N}(i)} w_{ij}(c'_i - c'_j) = \sum_{j\in\mathcal{N}(i)} \frac{w_{ij}}{2}(R_i + R_j)(c_i - c_j) \qquad (6)$$

Considered $\forall i \in [1,k]$, it can be expressed in matricial form as $\mathcal{L}c' = b$. The positional constraints imposed by the user have to be inserted in this equation, deleting the respective rows and columns from $\mathcal{L}$ and subtracting the known elements from the right-hand side. Therefore, the system matrix is a sub-matrix extracted from

the Laplacian matrix of the graph. This matrix is the same at every iteration, what changes is the right hand side of the system, i.e. the vector b. In order to speed up the computation time required by this minimization, the system matrix can be pre-factorized, only once before the first iteration, and re-used at each iteration to compute the new solutions. Being the dimension of the matrix $k-k_c$ in the order of many hundreds, but sparse, the factorization can be done very efficiently and therefore is fast.
The iteration of these two steps guarantees the minimization of the total energy converging after a certain number of iterations. To reduce the number of iterations needed to reach convergence, the initial guess should be as close as possible to the final position: the naïve bi-Laplacian over nodes displacement has turned out to be a good starting point.

2.3. Graph-mesh mapping

The deformation obtained on the graph has to be transferred to the initial mesh. For this step we still aim to avoid any expensive computation without renouncing to the preservation of finer mesh details. Based on the work done by Sumner et al. [SSP07], we implemented a simple and intuitive way to propagate global deformations where every point of the mesh p_i is reconstructed through a weighted sum of the transformations of control points c_j:

$$p_i' = \sum_{j \in \mathcal{K}} w_{ij}[R_j(p_i - c_j) + c_j + t_j] \tag{7}$$

where R_j is the local rotation matrix of the j-th control point and t_j is its displacement, such that $c_j' = c_j + t_j$. These parameters are available from the deformation energy minimization (Sec.2.2) where every transformations is defined as a local rotation and a translation.
Weight definition in (7) is a delicate matter since spreading over all mesh surface the computation of the rotation and translation for every vertex allows aesthetically pleasant results but risks to be computationally critical and can represent a bottleneck when the number of vertices and control points starts growing. On the other hand relying only on closest control points is computationally effective but can introduce unpleasant discontinuities to the mapping caused by significant changes of the set and relative influence of involved control points when crossing the patch borders. To avoid the above drawback, the weights w_{ij} are chosen to be a function of the distance of each vertex from every control point. This does solve the problem of the computational cost, because we can impose a decreasing function that goes to zero after a certain distance $d_{\max}$.

The sampling algorithm previously described allows obtaining a nearly uniform sampling on the mesh surface. Unfortunately, the sampling algorithm is not truly uniform, with the possibility to create patches with different sizes. Due to this differences, using the same identical mapping function for each patch could introduce some artifacts on the final result. In order to avoid these defects of the deformation algorithm, we introduce a modulation coefficient that takes care of the not perfect uniform distribution of the nodes on mesh surface. Analytically, the weights are defined as

$$w_{ij} = \begin{cases} (1 - \frac{r_{ij}}{d_j})^2 & \text{if } r_{ij} < d_j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

with $r_{ij} = ||p_i - c_j||$ and d_j maximum distance between the j-th center and all points inside the j-th patch.
Transferring both rotations and translation of each graph node to the mesh, allows preserving rotations and stretching introduced by the non-linear deformation algorithm, smoothly interpolating every finer local detail on the mesh surface.

3. Results

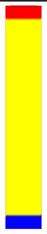
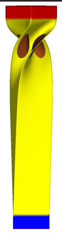
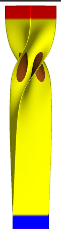
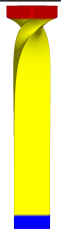
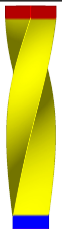
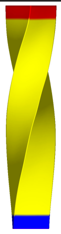
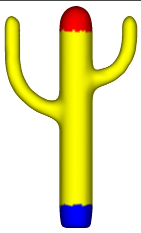
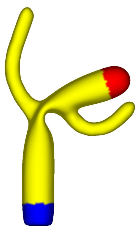
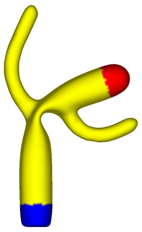
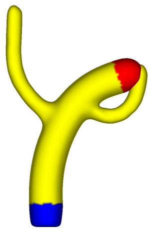
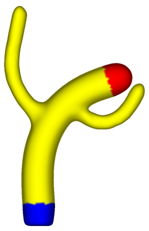
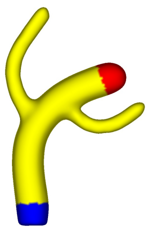
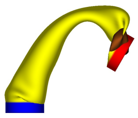
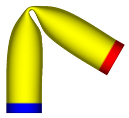
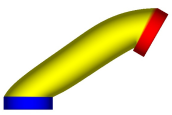
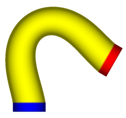
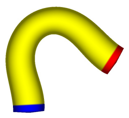
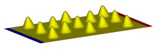
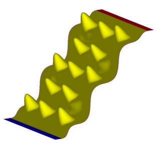
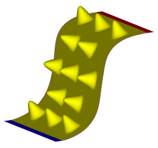
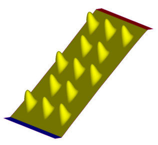
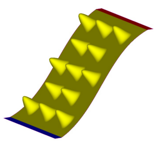
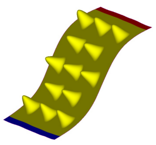
The results obtained with the proposed technique have been compared with the reference non-linear deformation methods ARAP [SA07] and SR-ARAP [LG15] both implemented in CGAL libraries [LSHXY17]. To provide faithful and accurate comparison between these methods we arranged our code implementation such that every algorithm can import a file where two things are specified: a tagging for every vertex and a transformation matrix for every tag. This file can be easily produced in a separate user-interaction phase and this allows to use the same handles and positional constraints for the different deformation techniques. Thanks to the tagging, vertices can be easily subdivided in regions and their final position computed thanks to the transformation matrices. We analyze and compare results in terms of visual quality of the deformation and computation times. Some of the meshes used to generate these results are courtesy of Stanford Computer Graphics Laboratory[†].

3.1. Deformation benchmarks

To verify that our mesh deformation technique behaves similarly according to other techniques, we first compare our system on the benchmark set introduced by Botsch and Sorkine in [BS08]. Four different examples are proposed: 135° twist (bar), 70° bend (cactus), 120° bend (cylinder), pure translation (knubbel). The deformed meshes obtained using the different methods are shown in Table 1, where colors are associated to handles (red), fixed regions (blue) and deformable regions (yellow). We represent both ARAP and SR-ARAP in two different instants: the first one, called *snapshot*, is the the result at the iteration step at which our algorithm reaches convergence (60 iterations for the *Bar*, 400 for the *Cactus*, 40 for the *Cylinder* and 100 for the *Plane*); the second one, instead, shows the results when they reach convergence.
From a visual quality point of view, ARAP behaves poorly in every case except for the Plane one. In the other cases its main disadvantage is that the deformation is done without minimally taking into account the volume of the object, i.e. meshes are treated as empty objects and this provides unsatisfying results [LG15]. SR-ARAP, instead, generates results very similarly to our method, but requires more iteration to converge to a suitable result. The corresponding computational time comparison is provided in Table 2. Even in these simpler cases, where involved meshes are not yet large, the difference between our method and the others is already significant. What most clearly emerge is the speed at which our method reaches convergence: it as at least one order of magnitude faster, but it can reach (in the *Plane* case) a speed up of two orders of magnitude.

† The Stanford 3D Scanning: http://graphics.stanford.edu/data/3Dscanrep/

Table 1: *Benchmark model deformations. From top to bottom: Bar (twisted), Cactus (bent), Cylinder (bent) and Knubbel (side translated).*

	ARAP		SR-ARAP		FCD
ORIGINAL	SNAPSHOT	CONVERGENCE	SNAPSHOT	CONVERGENCE	CONVERGENCE
					
					
					
					

3.2. Large mesh examples

We consider the main target of our work, which is providing a fast method able to produce high quality deformations even on large meshes (of the order of 1M vertices and more), with timings compatible with fluid user interactions. In this case, due to the high computational cost for the deformation techniques we compare with, only the pre-processing time and the time for one iteration has been compared. The pre-processing for ARAP and SR-ARAP consists in the computation and Cholesky factorization of the bi-Laplacian matrix while, in our case, it is the computation of the deformation graph. When applied to meshes having a high number of vertices, the difference between our method and the others becomes more defined. Examples of resulting meshes deformed with our method are shown in Figure 3 where it was not feasible to produce deformed meshes with ARAP and SR-ARAP with the hardware in use (a PC with an Intel Core i5 and 8 GB of RAM). Meshes tested in this session have not been chosen casually, they provide some specific examples in order to show the potential of our method. The David mesh, has a *dirty* topology, causing corruptions on the Laplacian matrix, making it impossible to pass the pre-process stage for both ARAP and SR-ARAP. On the contrary, being the neighborhood the only mesh information exploited by our algorithm during pre-processing, we could have failures only if isolated vertices are present, which is not the case in most cases. Dragon is too big for other methods to be pre-processed: the hardware used for testing froze most of the times, otherwise taking high computation times. Hulk, instead, is an example of a mesh for which our method proceeds flawlessly, while for other methods is already heavy to process. In all examples even in presence of imposed large deformations we observe pleasantly deformed objects. What is even more significant is what is shown in Table 3 where computational performance of our solution are compared and where it is evident how much faster and less expensive is our algorithm with respect to the others. Deformation graph construction convergence can be achieved in less than 20s for a mesh with more than 1 Million vertices. The other techniques require at least one minute only for pre-processing and ten minutes to compute 100 iterations of the methods, while still being far away from the desired final result.

3.3. Limitations and future works

One aspect that seems to introduce an undue computational overload is to adopt a large neighborhood for the graph to mesh deformation mapping. However, a suitably wide influence area for transform mapping is necessary to avoid artifacts generated by change of the set of neighboring patches. In case of too small neighborhood this might introduce discontinuity effects such as those shown in Fig.4 that depicts an example of the usage of different radii, highlighting the defects created by the smaller one.
Another limitation, somehow related to the former one, is due to the approximation we make when considering the Euclidean distance in the mapping. As already noted by Sumner et al. [SSP07], this could in some cases generate unwanted influences from seemingly close but erroneously included control points which actually are at higher geodesic distance. Unfortunately, the calculation of geodesic distance would weigh down the method too much. Therefore a possible advancement could be to find heuristics to reliably enough replacing the geodesic distance estimates in the area of weighting influence, so as to reduce most critical manifestations of the problem. Many other literature solutions could have been considered in principle to complete our comparisons, however we excluded any method we found to require either a relevant overload for the user or already order of seconds computation time for relatively small models (up to 100k points).
In our future works we want to explore new methods to map the deformation applied to the graph to the whole mesh surface, exploiting the information derived from the non-linear deformation algorithm and from graph generation: the mapping problems are intrinsically avoided by the deformation algorithm, that knows the connectivity. Exploiting this information, distant regions on mesh surface should be able to ignore each other. The directions to explore can be different, based on what we want to change: from one point of view, changing the quantity to interpolate can bring to more satisfying results; from another point of view, instead, modifying the weights used for interpolation can completely remove all the artifacts generated by the method. In this latter case, the desired result would be a function that, for every control point, propagates smoothly over the surface, avoiding any inconsistency introduced by the difference between geodesic and Euclidean distances.

Table 3: *Comparison of computational times (High dimensionality dataset)*

		ARAP	SR-ARAP	FCD
David (507k)	Pre-Process	FAIL	FAIL	3.6s
	Iteration 100	-	-	1.7s
Hulk (1.4M)	Pre-Process	50s	60s	13s
	Iteration 100	∼670s	∼600s	2.2s
Dragon (3.6M)	Pre-Process	128s	120s	36s
	Iteration 100	∼9800s	∼8000s	5s

Table 2: *Comparison of computational times (Benchmark examples)*

		ARAP	SR-ARAP	FCD
Bar (6k)	Pre-Process	33ms	33ms	33ms
	Iteration 60	770ms	630ms	138ms
	Convergence	5.2s	14.6s	138ms
Cactus (5k)	Pre-Process	25ms	26ms	27ms
	Iteration 400	3.1s	3.9s	691ms
	Convergence	4.7s	10.0s	691ms
Cylinder (5k)	Pre-Process	24ms	26ms	30ms
	Iteration 40	279ms	317ms	129ms
	Convergence	5.2s	5.9s	91ms
Plane (40k)	Pre-Process	388ms	377ms	246ms
	Iteration 100	10.1s	7.9s	793ms
	Convergence	216s	196s	793ms

4. Conclusion

Our method introduces a fast deformation technique able to exploit non-linear deformations to meshes with millions vertices. This is done avoiding any pre-computation (details extraction, subspace creation,...) that would require the resolution of a sparse linear system or harmonic subspaces. Our solution evolves from the work of Sumner et al. [SSP07] where we use an adaptation of a non-linear SR-ARAP [LG15] to an automatically generated centroidal deformation graph. However, this combination and adaptation yield very much more than the sum of the parts, since we have overcome the limitations of both originating approaches, obtaining the possibility of making large and free deformations on large meshes in times compatible with a free interaction of the user. As far as we know this had not yet been obtained before. This can also be seen as a starting point for further developments that are to be done to overcome the residual limitations and to bring this technique exploited in professional fields.

References

[BBK05] Botsch M., Bommes D., Kobbelt L.: Efficient linear system solvers for mesh processing. In *Mathematics of Surfaces XI* (2005), pp. 62–83. 2

[BJ05] Barbič J., James D. L.: Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph. 24*, 3 (July 2005), 982–990. 2

[BK04] Botsch M., Kobbelt L.: An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 630–634. 2

[BK05] Botsch M., Kobbelt L.: Real-time shape editing using radial basis functions. *Computer Graphics Forum 24*, 3 (2005), 611–621. 2

[BP07] Baran I., Popović J.: Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3 (July 2007). 1

[BPGK06] Botsch M., Pauly M., Gross M., Kobbelt L.: PriMo:

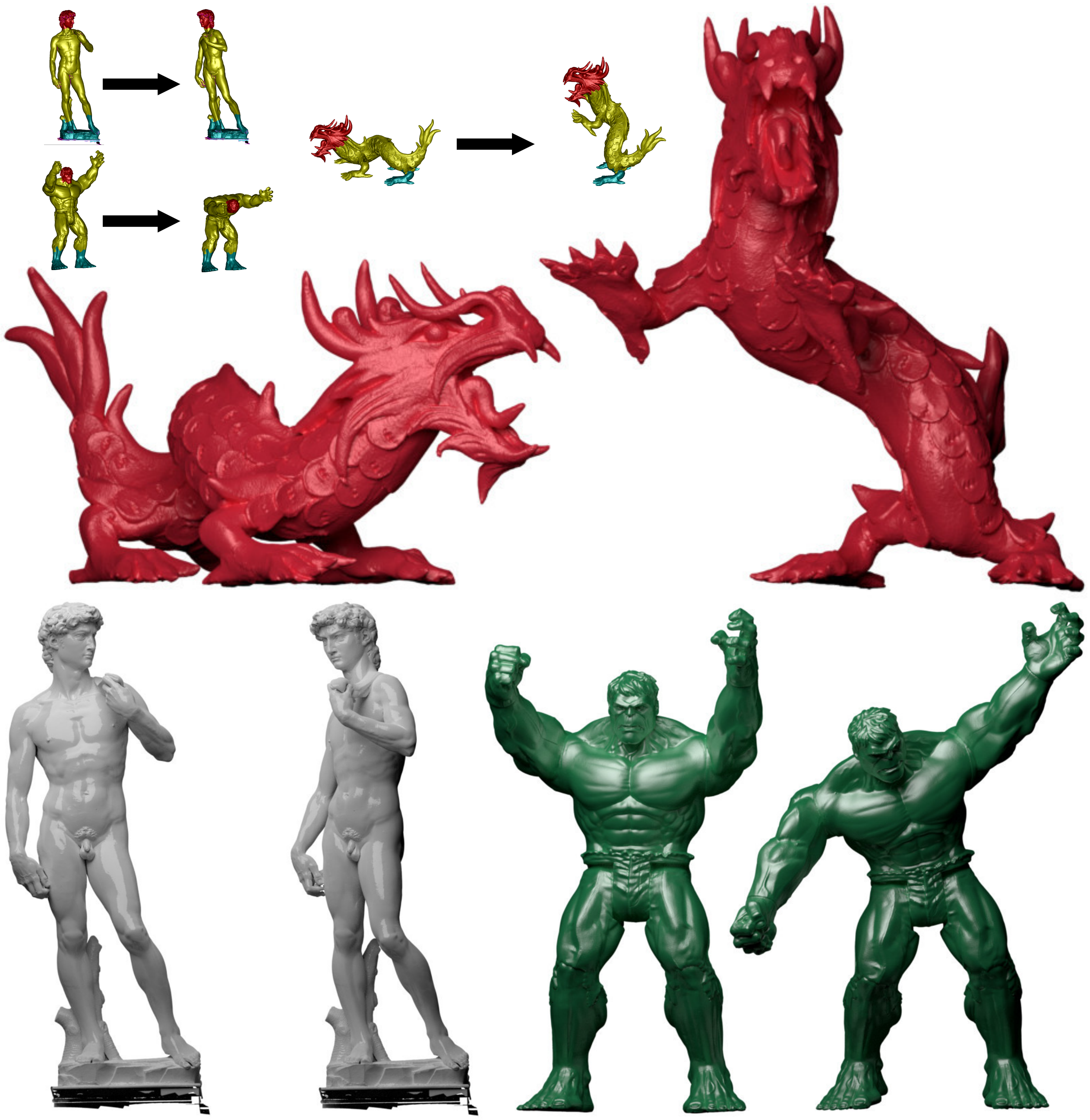


Figure 3: *Results on large models: Dragon (red), David (white) and Hulk (green). Handles are evidenced on the top-left corner.*

Coupled Prisms for Intuitive Surface Modeling. In *Symposium on Geometry Processing* (2006), Sheffer A., Polthier K., (Eds.), The Eurographics Association. 2

[BPK*07] BOTSCH M., PAULY M., KOBBELT L., ALLIEZ P., LÉVY B., BISCHOFF S., RÖSSL C.: Geometric modeling based on polygonal meshes. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 2

[BPWG07] BOTSCH M., PAULY M., WICKE M., GROSS M.: Adaptive space deformations based on rigid cells. *Computer Graphics Forum 26*, 3 (2007), 339–347. 2

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (Jan. 2008), 213–230. 1, 6

[BSB14] BONARRIGO F., SIGNORONI A., BOTSCH M.: Deformable registration using patch-wise shape matching. *Graph. Models 76*, 5 (Sept. 2014), 554–565. 3

[BSPG06] BOTSCH M., SUMNER R. W., PAULY M., GROSS M.: Deformation transfer for detail-preserving surface editing. In *Vision, Modeling, and Visualization 2006* (2006), Akademische Verlagsgesellschaft AKA, pp. 357 – 364. 2

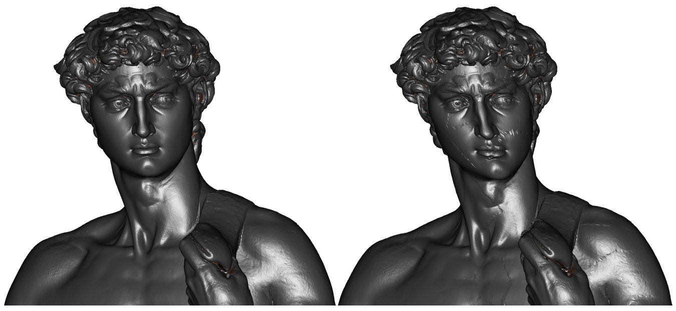


Figure 4: *Correct mapping (left) and patching artifacts (right) generated by a wrong mapping configuration.*

[CDA99] COTIN S., DELINGETTE H., AYACHE N.: Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (Jan 1999), 62–73. 1

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Trans. Graph. 29*, 4 (July 2010), 38:1–38:6. 2

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 905–914. 3

[dBvdSB07] DE BOER A., VAN DER SCHOOT M. S., BIJL H.: Mesh deformation based on radial basis function interpolation. *Comput. Struct. 85*, 11-14 (June 2007), 784–795. 2

[FB88] FORSEY D. R., BARTELS R. H.: Hierarchical b-spline refinement. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 205–212. 2

[GSS99] GUSKOV I., SWELDENS W., SCHRÖDER P.: Multiresolution signal processing for meshes. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, pp. 325–334. 2

[HSL*06] HUANG J., SHI X., LIU X., ZHOU K., WEI L.-Y., TENG S.-H., BAO H., GUO B., SHUM H.-Y.: Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3 (July 2006), 1126–1134. 2

[HSTP11] HILDEBRANDT K., SCHULZ C., TYCOWICZ C. V., POLTHIER K.: Interactive surface modeling using modal analysis. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 119:1–119:11. 2

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3 (July 2005), 561–566. 2

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Trans. Graph. 24*, 3 (July 2005), 399–407. 1, 2

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, pp. 105–114. 2

[KR15] KARADE V., RAVI B.: 3d femur model reconstruction from biplane x-ray images: a novel method based on laplacian surface deformation. *International Journal of Computer Assisted Radiology and Surgery 10*, 4 (Apr 2015), 473–485. 1

[LG15] LEVI Z., GOTSMAN C.: Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE Transactions on Visualization and Computer Graphics 21*, 2 (feb 2015), 264–277. 2, 4, 5, 6, 8

[Lip04] Differential coordinates for interactive mesh editing. In *Proceedings of the Shape Modeling International 2004* (2004), SMI '04. 2

[Llo82] LLOYD S.: Least squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137. 3

[LSHXY17] LORIOT S., SORKINE-HORNUNG O., XU Y., YAZ I. O.: Triangulated surface mesh deformation. In *CGAL User and Reference Manual*, 4.11 ed. CGAL Editorial Board, 2017. 6

[Mac67] MACQUEEN J. B.: Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* (1967), 281 Ű– 297. 3

[MCR16] MOSER L., CORRAL D., ROBLE D.: As-rigid-as-possible deformation transfer for facial animation. In *ACM SIGGRAPH 2016 Talks* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 29:1–29:2. 2

[MTLT88] MAGNENAT-THALMANN N., LAPERRIÈRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88* (Toronto, Ont., Canada, Canada, 1988), Canadian Information Processing Society, pp. 26–33. 2

[MYF07] MASUDA H., YOSHIOKA Y., FURUKAWA Y.: Preserving form features in interactive mesh deformation. *Computer-Aided Design 39*, 5 (2007), 361 – 368. 1

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836. 1

[PBGC18] PALMA G., BOUBEKEUR T., GANOVELLI F., CIGNONI P.: Scalable non-rigid registration for multi-view stereo data. *ISPRS Journal of Photogrammetry and Remote Sensing 142* (2018), 328–341. 3

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), SGP '07, Eurographics Association, pp. 109–116. 2, 4, 6

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (New York, NY, USA, 2004), SGP '04, ACM, pp. 175–184. 2

[SGA*16] SIEGER D., GAULIK S., ACHENBACH J., MENZEL S., BOTSCH M.: Constrained space deformation techniques for design optimization. *Comput. Aided Des. 72*, C (Mar. 2016), 40–51. 1, 2

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 151–160. 2

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 399–405. 2

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3 (July 2007). 2, 3, 6, 8

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[vRESH16] VON RADZIEWSKY P., EISEMANN E., SEIDEL H.-P., HILDEBRANDT K.: Optimized subspaces for deformation-based modeling and shape interpolation. *Comput. Graph. 58*, C (Aug. 2016), 128–138. 2

[WJBK15] WANG Y., JACOBSON A., BARBIČ J., KAVAN L.: Linear subspace design for real-time shape deformation. *ACM Trans. Graph. 34*, 4 (July 2015), 57:1–57:11. 2

[YLL*16] YANG L., LI B., LV Z., HOU W., HU P.: Finite element mesh deformation with the skeleton-section template. *Computer-Aided Design 73* (2016), 11 – 25. 1

[ZSS97] ZORIN D., SCHRÖDER P., SWELDENS W.: Interactive multiresolution mesh editing. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 259–268. 2