Eurographics Symposium on Geometry Processing 2013
Yaron Lipman and Richard Hao Zhang
(Guest Editors)

Volume 32 (2013), Number 5

Animation-Aware Quadrangulation

Giorgio Marcias [1,2] Nico Pietroni [1] Daniele Panozzo [3] Enrico Puppo [4] Olga Sorkine-Hornung [3]

[1] ISTI - CNR, Pisa, Italy [2] University of Pisa, Italy
[3] ETH Zurich, Switzerland [4] University of Genova, Italy

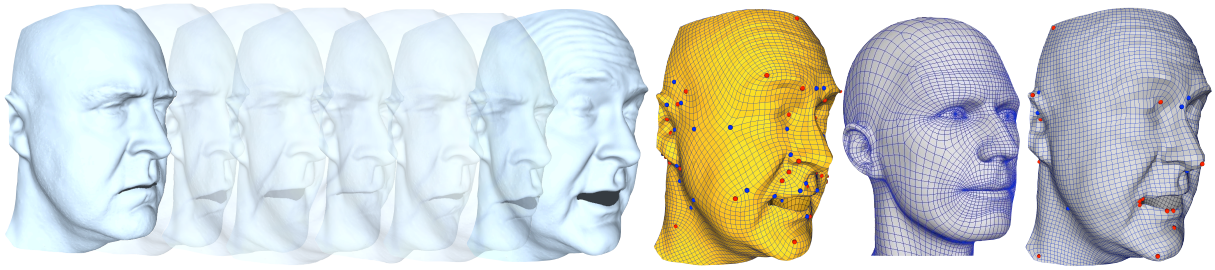

Figure 1: *Given a sequence of triangle meshes sampling an animation (left), we produce a quadrilateral mesh that can fit all animation frames well (center, yellow). Our mesh better represents natural shape structures, just like meshes designed by artists (center-right), while a quadrangular mesh computed from a single pose with a state-of-the-art method [BZK09] (right, grey) exhibits a much more artificial structure. Colored dots highlight singularities of the quad meshes.*

Abstract

Geometric meshes that model animated characters must be designed while taking into account the deformations that the shape will undergo during animation. We analyze an input sequence of meshes with point-to-point correspondence, and we automatically produce a quadrangular mesh that fits well the input animation. We first analyze the local deformation that the surface undergoes at each point, and we initialize a cross field that remains as aligned as possible to the principal directions of deformation throughout the sequence. We then smooth this cross field based on an energy that uses a weighted combination of the initial field and the local amount of stretch. Finally, we compute a field-aligned quadrangulation with an off-the-shelf method. Our technique is fast and very simple to implement, and it significantly improves the quality of the output quad mesh and its suitability for character animation, compared to creating the quad mesh based on a single pose. We present experimental results and comparisons with a state-of-the-art quadrangulation method, on both sequences from 3D scanning and synthetic sequences obtained by a rough animation of a triangulated model.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Most models used in character animation for game and film production are based on quadrilateral meshes, since they are easier to edit and animate than triangle meshes. Moreover, they can be used as control cages for the Catmull-Clark subdivision, which is the standard de facto in the industry to generate smooth surfaces. In a standard production pipeline, a mesh is designed *before* the definition of the skeletal and skinning structures for animation, and of the animation itself. However, an artist who generates the mesh must take into account *in advance* the deformation that the mesh will undergo during animation, in order to minimize artifacts. In order to enable high-quality deformations, the elements of the control mesh should remain nearly flat, rectangular and aligned with shape features and principal directions of curvature as much as possible throughout the entire animation.

The dynamic shape features of living bodies are aligned to bundles of muscles and Langer's lines, which form a virtual

frame field on the surface. Such lines intersect to form rather complicated configurations, which characterize the topology of this frame field. While it is often assumed that a good quad mesh should exhibit high regularity - i.e., it should contain a small number of singularities connected through separatrix lines to form a simple layout of patches [BLP*13] - in fact a certain amount of singularities is necessary to follow natural shape features properly. Meshes designed by artists, such as the one depicted on the center-right side of Figure 1, contain a relatively large amount of singularities, yet they result in natural and aesthetically-pleasing surfaces and are ready for animation.

In recent years, several methods have been proposed in the geometry processing literature, which try to automatically generate quadrilateral meshes with features similar to those that artists design by hand [BLP*13]. However, presently, no automatic method can produce a quad mesh with acceptable quality for direct subsequent use in production. We observe that all existing methods take a static shape as input, i.e., a single pose in the case of a model to be animated, and thus they do not take into account the information coming from animation of the shape. On the other hand, recent technologies allow for the generation of animated meshes, i.e., the real time capture of 3D characters in motion (see, e.g. [BHB*11, LLV*12]). Mesh sequences with point-to-point registration can be obtained, which provide a sampling of the real animation.

In this paper, we present the first method that exploits such dynamic data in designing a quad mesh, by mimicking, at least partially, the principles followed by artists. In particular, we develop a method that tries to create mesh elements that will stay as rectangular as possible throughout deformations induced by the animation. Our method is based on a guidance field to align mesh elements, and it uses existing technology for mesh quadrangulation, with proper modifications. Our main contribution is a simple and efficient method for producing an initial guidance field by local analysis of the shape throughout the animation: we analyze the local deformation that the shape undergoes at each point, and we set a cross field on a reference pose in the sequence, in such a way that the directions of this field will remain as orthogonal as possible during animation.

Our experimental results show that, besides producing a quadrangulation that adapts well to the animation, the resulting mesh alignment is also natural, even if no manually-specified directional constraints are given as input. For instance, in the example depicted in Figure 1, the mesh we obtain (shown in the center) contains a relatively higher number of singularities (depicted with blue and red dots), but it is much closer to natural orientation than a mesh obtained with a state-of-the-art quadrangulation algorithm (shown on the right).

2. Previous Work

Recent literature on mesh generation and processing is quite extensive, and its discussion is beyond the scope of this paper. We refer the interested reader to [BLP*13] for a recent survey, and here we briefly review the most relevant results for our purposes.

A few quadrangulation methods, which were designed primarily to obtain mesh elements with a given alignment, are based on a *guidance field*, i.e., a field of directions defined on the surface to be quadrangulated. Such a guidance field is either taken as input or computed from a given set of constraints, and it is meant to align to shape features and/or to principal curvature directions. Several approaches exist to automatically compute the guidance field: some methods use nonlinear formulations based on periodic functions [HZ00, PZ07, RVAL09], while others are based on an integer-valued representation [RVLL08, BZK09]. Quadrangulation is computed such that the edges of the quads are aligned to field directions. Methods that try to directly trace directions of the guidance field (e.g., [ACSD*03]) present some difficulties, and they only produce quad-dominant meshes. Methods based on global parametrization [RLL*06, KNP07, BZK09] currently exhibit the best results. Nevertheless, since optimization algorithms frequently get trapped in local minima, both during estimation of the guidance field and during parametrization, the results still suffer from several problems, e.g., in the proper positioning of singularities. Note that our contribution is orthogonal with respect to these quadrangulation algorithms: our method works at their early stage, namely the generation of a guidance field, and new optimization algorithms can be plugged into the pipeline as they become available.

A number of recent works deal with the generation of mesh sequences from the outcome of 3D scanning of moving characters. In [VBMP08], a system is proposed that, given multi-view synchronized video streams of a human performance and an articulated template of the performer, captures the motion of both the skeleton and the shape, and produces a sequence of meshes in full correspondence. More recent results [VPB*09] allow for high-resolution capture of human-sized moving 3D geometry up to a temporal resolution of 60 Hz, without requiring prior knowledge. Shape completion of sequences affected by occlusions and holes has been addressed in [LLV*12]. For the case of multiple poses without time coherence, non-rigid alignment has been addressed by several authors (see, e.g., [BHKH13]) and the reconstruction of a temporally coherent sequence has been addressed in [BHLW12], by also considering changes in topology. Face capture is a special case, handled in several recent works [BBB*10, BHPS10, BHB*11, GFT*11]. Some of the available systems are low-cost, yet able to capture very accurate (pore-scale) geometry at high frame rates. Industrial applications of similar technologies start emerging for both face and full-body scans [IR], yet the resulting models must be processed and remeshed by user-driven tools such as ZBrush [ZB] in order to obtain models that are suitable for subsequent production [TG].

Various recent works deal with processing of mesh sequences for the purpose of smoothing, simplification, editing, deformation transfer and more, but none handle deformation-aware quad remeshing. A common technique

relies on the analysis of the deformation gradient [SP04]; for instance, [JT05] converts a mesh sequence into a skinned model by clustering that undergo similar transformations. In our work, we also employ the deformation gradients to estimate the stretch of each mesh triangle.

3. Animation-aware quadrangulation

Our method, like several other quadrangulation methods, computes a quad mesh by means of a guidance field. Our main contribution is that the guidance field is computed by taking into account all deformations that the shape undergoes during animation.

Given an input sequence $M_0,\dots,M_k$ of triangle meshes with point-to-point correspondence, our method can be summarized as follows:

1. *Local analysis:* for each triangle t of M_0 independently, we analyze the deformation of t through the sequence, and we initialize a cross field at t, which is aligned with the principal directions of the average deformation; we also compute a stretch factor, which is proportional to the amount of stretch that t undergoes; this initial cross field on mesh M_0 is passed to the following phase;
2. *Smoothing and quadrangulation:* we follow a standard method with minor modifications:
   a. *Smoothing:* we modify the smoothing phase of the Mixed Integer Quadrangulation (MIQ) method [BZK09], analogously to [PLPZ12], by incorporating a term in the energy, which penalizes distance from the input field as a soft constraint; the method can optionally incorporate hard constraints;
   b. *Quadrangulation:* we use an off-the-shelf method for quadrangulation guided by a cross field - such as the MIQ itself - to produce a quadrangulation that fits M_0;
   c. *Animation of quad mesh:* we deform the quad mesh to fit it to all other frames.

Since the initial field computed in Phase 1 depends exclusively on deformation, it reveals shape features that originate from animation, while it disregards other features that may occasionally occur in specific poses. We adopt a similar approach to optionally incorporate hard constraints in Phase 2a: only sharp features that appear in *all* frames, i.e., remain stable throughout animation, are considered.

3.1. Local analysis

An animation in the continuum is a map $A : S \times [0,1] \to \mathbb{R}^3$, where S is a smooth 2-manifold, which assigns to each point of S at each instant of time a position in 3D space. Let p be a point of S, and U_p be a neighborhood of p. The evolution $A(U_p,t)$ through time represents variations in the shape of U_p, thus providing local information about the dynamic behavior of S at p, e.g., how and how much it is deformed.

A *frame field* on a smooth surface is defined point-wise as a pair of non-parallel vectors on the tangent plane of each point of the manifold.

Such two vectors, together with their opposite vectors, define a cross of directions on the tangent plane (see inset). A special case of a frame field is the 4-RoSy field, as described in [RVLL08], in which vectors are unit-length and mutually orthogonal.

Let us assume a 4-RoSy field $\mathcal{C}$ is defined on S. The Jacobian of animation A acts on the tangent bundle of S and it maps the 4-RoSy field $\mathcal{C}$ to a frame field $\mathcal{C}_t$ for any generic frame $S_t = A(S,t)$ of animation. In general, $\mathcal{C}_t$ will *not* be a 4-RoSy field, i.e., its two vectors at each point will not necessarily be orthogonal or unit-length. If we consider the first order approximation of A in the neighborhood U_p of p and we express A in local 2D coordinates centered at p, then the Jacobian of $A(p,t)$ is expressed by a 2×2 matrix $J_{p,t}$. The polar decomposition $J_{p,t} = R_{p,t}P_{p,t}$ isolates in $P_{p,t}$ the deformation of the neighborhood U_p (while $R_{p,t}$ is the nearest rotation to $J_{p,t}$). From the singular value decomposition $J_{p,t} = U\Sigma V^*$, we have:

$$R_{p,t} = UV^*, \quad P_{p,t} = V\Sigma V^*.$$

From the relation between SVD and principal component analysis, we also know that the columns of U are the principal directions of $J_{p,t}$ and, similarly, that the columns of V are the principal directions of $P_{p,t}$. Now, since $J_{p,t}V = U\Sigma$, it means that $J_{p,t}$ maps the directions of principal components of $P_{p,t}$ to the directions of principal components of $J_{p,t}$ itself. Therefore, since principal directions are always mutually orthogonal, if we align $\mathcal{C}$ at all points to the columns of V, its image $\mathcal{C}_t$ through J_t will remain orthogonal (not unit-length, though). In fact, this is the only possible orientation for the field $\mathcal{C}$ to remain orthogonal under transformation J_t.

In the discrete setting, let $\mathcal{M} = (M_0,\dots,M_k)$ be an input sequence of triangle meshes with point-to-point correspondence, i.e., all meshes have the same structure $(\mathcal{V},\mathcal{E},\mathcal{F})$ of vertices $\mathcal{V}$, edges $\mathcal{E}$ and faces $\mathcal{F}$, while each mesh M_i has a different mapping of vertices $R_i : \mathcal{V} \to \mathbb{R}^3$. Sequence $\mathcal{M}$ provides a sampling of animation A as described above.

Without loss of generality, we select mesh M_0 as a reference position (the sequence is not necessarily sorted with respect to time, so any other mesh could be selected). We analyze the deformations of each triangle of M_0 throughout the sequence, and we set an initial discrete 4-RoSy field $\mathcal{C}_0$ on M_0, such that the transformed fields $\mathcal{C}_i$ on the other frames $(i = 1,\dots k)$ remain as orthogonal as possible. Local analysis is performed triangle by triangle as follows:

1. Let $t^0 = (a,b,c)$ be a triangle of M_0 and let $t^i = (a',b',c')$ be its corresponding triangle of M_i. The *deformation gradient* J_t^i is the unique affine transformation that maps t^0 to t^i. We disregard the translational component, which is not relevant for our purposes, and we derive the linear component by first expressing both triangles in a common 2D reference frame:
   a. Triangle t^0 is represented in a frame that has its origin at a and the x-axis aligned with $b-a$, and similarly for t^i;

b. The deformation gradient J_t^i that maps t^0 to t^i is obtained by solving a 2×2 linear system.

2. We compute the SVD $J_t^i = U_t^i \Sigma_t^i V_t^i$:

- We back-project the columns of V_t^i to 3D, by using the reverse mapping to the local frame of t^0, and we save them as ideal directions for a 4-RoSy field on t^0 to be mapped to t^i;
- The singular values s_1 and s_2 in the diagonal matrix Σ_t^i represent the stretching induced by J_t^i along these directions. Assuming $s_1 > s_2$, we define the *stretch factor* associated with the transformation as $s_t^i = |s_1|/|s_2| - 1$.

In order to remove outliers, we clamp both singular values s_1, s_2 to an interval $[s_{\min}, s_{\max}]$ before computing the stretch factor. In all our experiments we obtained good results by setting $s_{\min} = 0.25$ and $s_{\max} = 4$, i.e., we assume that each triangle is neither stretched nor compressed more than four times its size.

After this analysis, for each triangle t and for each frame M_i we have an ideal orientation $\mathcal{C}_t^i$ for a 4-RoSy field on M_0, to maintain its orthogonality when mapped to M_i, and a stretch factor s_t^i providing a weight for the relative importance of M_i in "pulling" the cross field towards its desiderata.

In order to obtain a cross field that tries to adapt to all poses, we compute a field $\bar{\mathcal{C}}$ triangle by triangle, as the average modulo $\pi/2$ of the $\mathcal{C}_t^i$'s, each weighted by its stretch factor s_t^i. The field in triangle t is represented by the signed angle between a representative vector of the cross field and an arbitrary direction in the tangent plane [BZK09, RVLL08]. We compute the average angle modulo $\pi/2$ as follows:

$$\bar{\theta}_t = \frac{1}{4}\Phi\left(\frac{\sum_i s_t^i \Psi(4\theta_t^i)}{\sum_i s_t^i}\right)$$

where $\Psi(\theta) : \mathbb{R} \to \mathbb{R}^2 = (\cos\theta, \sin\theta)$ and $\Phi(x,y) : \mathbb{R}^2 \to \mathbb{R} = \text{atan2}(y,x)$. We also compute the average stretch at t as the weighted average

$$s_t = \frac{1}{\sum_{i=1}^k s_t^i} \sum_{i=1}^k s_t^i \cdot s_t^i$$

Note that in the formula above the s_t^i's are used both as quantities to be averaged and as weights for the average itself. In this way, we compensate for the fact that some parts of the surface might remain nearly rigid for a large part of the sequence and then undergo large deformations on relatively few frames.

All computations of this phase are done triangle by triangle independently and involve simple computations. If M_0 has n triangles and there are $k+1$ frames in input, the computational complexity is linear, i.e., $O(kn)$, and it admits a trivial parallelization.

3.2. Field smoothing and quadrangulation

We use the results of the previous step to define a smooth guidance field on mesh M_0 to feed into the quadrangulation phase. We adopt an approach analogous to the MIQ, except that we modify the energy for smoothing the cross field, in order to allow for soft constraints, following [PLPZ12]. Note that all the following computations, except for the final mapping of quadrangulation to the other frames, involve just the reference mesh M_0.

As outlined in the previous section, a discrete 4-RoSy field $\mathcal{C}$ is represented at triangle t by an angle θ with respect to a local frame [RVLL08]. Smoothness of the field across the mesh means that the fields of each pair of triangles t, t' adjacent along an edge $e_{t,t'}$ should be not too different. However, we cannot just measure the difference $|\theta - \theta'|$ because this angle representation is ambiguous: the local frames of different triangles are generally different, and also the angle in every triangle is unique only up to a rotation by multiples of $\pi/2$. Let $\kappa_{t,t'}$ be the (fixed) rotation angle between the local frames of t and t', and let $p_{t,t'}$ be an integer variable, called the *period jump*. Now the field in triangle t can be expressed relatively to a neighboring triangle t' as $\theta + \kappa_{t,t'} + p_{t,t'}\pi/2$.

Following [PLPZ12], we define the following energy, which combines a smoothness term based on the formula above, together with a term measuring faithfulness to the orientation $\bar{\theta}_t$ computed in the previous section (which is thus used as a soft constraint):

$$E = (1-\alpha) \sum_{e_{t,t'} \in \mathcal{E}} \left(\theta_t + \kappa_{t,t'} + \frac{\pi}{2} p_{t,t'} - \theta_{t'}\right)^2 + \alpha \sum_{t \in \mathcal{F}} s_t \left(\theta_t - \bar{\theta}_t\right)^2. \quad (1)$$

The parameter α is used explicitly to allow the user trading off between smoothness and faithfulness to the initial cross field. With $\alpha = 0$ the result will be identical to the standard MIQ computed on the reference mesh, while $\alpha = 1$ would return back our input field. Note that, while smoothness is computed just on the reference mesh, the initial cross field $\bar{\mathcal{C}}$ takes into account the whole animation sequence.

Optimization of the field $\mathcal{C}$ with respect to Energy (1) is a mixed-integer problem, which can be solved with a greedy optimization algorithm. Optionally, it is possible to introduce hard feature constraints given as a set of angles $\{\hat{\theta}_{i_1}, \ldots, \hat{\theta}_{i_k}\}$ on a subset of triangles, which remain fixed during optimization. Such constraints may refer to shape features of high curvature anisotropy. See [BZK09] for details on mixed-integer optimization and hard constraints based on curvature anisotropy. Differently from the static (single pose) case, we select a triangle as a hard constraint if and only if curvature shows high anisotropy *in all poses*. This simple criterion guarantees that we are constraining our mesh to be aligned with true sharp features of the object, not occasional shape features that arise just in a specific pose.

Note that, in general, hard constraints should be used with care. A single outlier direction used as a hard constraint may severely perturb the RoSy field and introduce several singularities. Additionally, numerical methods for estimating curvature directions suffer from instability in the pres-

Model	Face0	Face	Squat	Crane
Source	[BHPS10]	[BHB*11]	[VBMP08]	
Facets	50K	100K	20K	20K
Frames	76	287	41	50

Table 1: *Statistics on datasets.*

ence of noise. By contrast, soft constraints as in our energy have a more moderate behavior: an outlier direction would be smoothed out if it increases the smoothness part of the energy too much. However, since we assess the stability of hard constraints throughout the sequence, we drastically reduce the presence of outliers, thus obtaining a stable behavior even if we include them in the optimization.

We compute a quadrangular mesh on the reference position by feeding the parametrization/quadrangulation phase of the standard MIQ with cross field $\mathcal{C}$ computed with the smoothing scheme above. Note that our method is fully modular with respect to this phase: any other off-the-shelf algorithm for quadrangulation guided by a cross field could be used instead.

Finally, the quadrangular mesh is mapped to all other frames. Since the MIQ provides a parametrization of the input mesh M_0, we know the barycentric coordinates (a_q, b_q) of every vertex q of the quad mesh with respect to a triangle $t = (u, v, w)$ of M_0. The 3D position of q on the first pose is thus computed by combining the barycentric coordinates (a_q, b_q) with the coordinates $(R_0(u), R_0(v), R_0(w))$ of t on pose M_0. Analogously, the quad mesh corresponding to each other pose M_i is computed by using, for each vertex q, the same barycentric coordinates combined with the coordinates $(R_i(u), R_i(v), R_i(w))$ of t on pose M_i. In case a different quadrangulation method is adopted, which does not produce a parametrization, a similar result can be obtained by first computing barycentric coordinates of each vertex q of the initial quad mesh through projection of q onto reference mesh M_0.

4. Results

Our framework has been implemented in C++, by using the VCG library for mesh processing [VCG] and the Eigen library for numerical computation [Eig]. Experiments are performed on a Macbook pro with 2.3 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 RAM, by using a single core.

We present results on real data from 3D dynamic range scanning: two face sequences from [BHPS10] and [BHB*11], respectively, and two full body sequences from [VBMP08] containing large limb movements and garment draping. Statistics on input datasets are given in Table 1. The Face dataset has been decimated with respect to the original (5M triangles per mesh) to allow for easier processing during the MIQ Phase, while our Local analysis Phase could easily run on the full resolution dataset. Processing the largest sequence takes about 0.5 seconds for local analysis and about 10 seconds for smoothing and quadrangulation with the MIQ optimizer. All datasets were processed by using α in the interval $[0.3, 0.5]$ in energy 1.

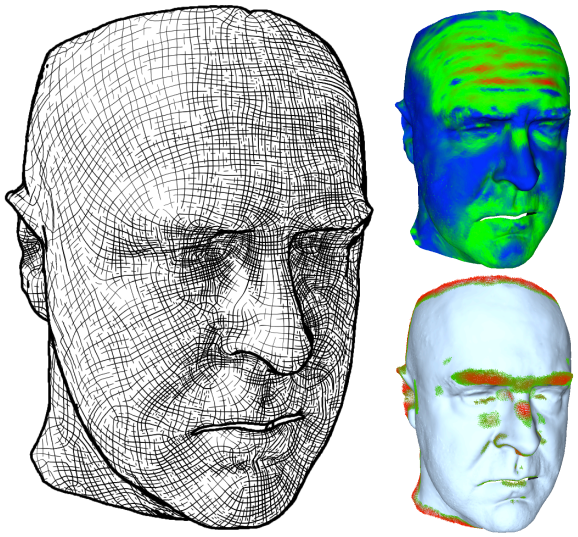

Figure 2: *Face dataset. The initial field (left) and the corresponding map of stretch (upper right) computed through local analysis: color scale goes from blue to green to red; hard constraints with curvature anisotropy > 0.6 on the first frame used for MIQ (lower right).*

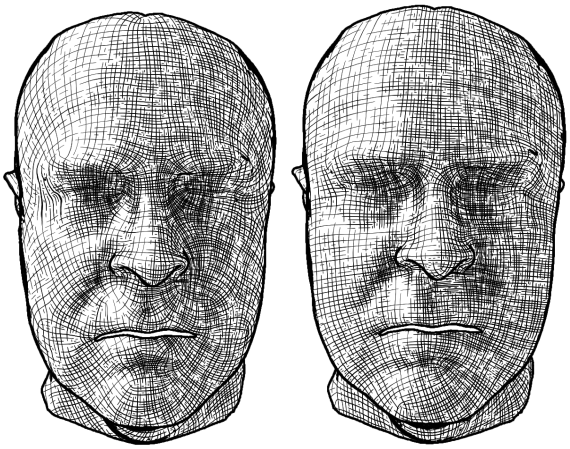

Figure 3: *Face dataset. The 4-RoSy field after smoothing, computed with our energy by using just soft constraints (left), and computed with standard MIQ energy with hard constraints (right).*

In Figure 2 we show the initial field on the Face dataset, as well as its corresponding map of stretch, computed through our local analysis. Note how wrinkles on the forehead, which appear just at the late stages of animation and produce major stretch, are detected. Note also how the overall alignment of the final mesh is already evident in this non-smooth field. We also show the set of hard constraints used for the stan-

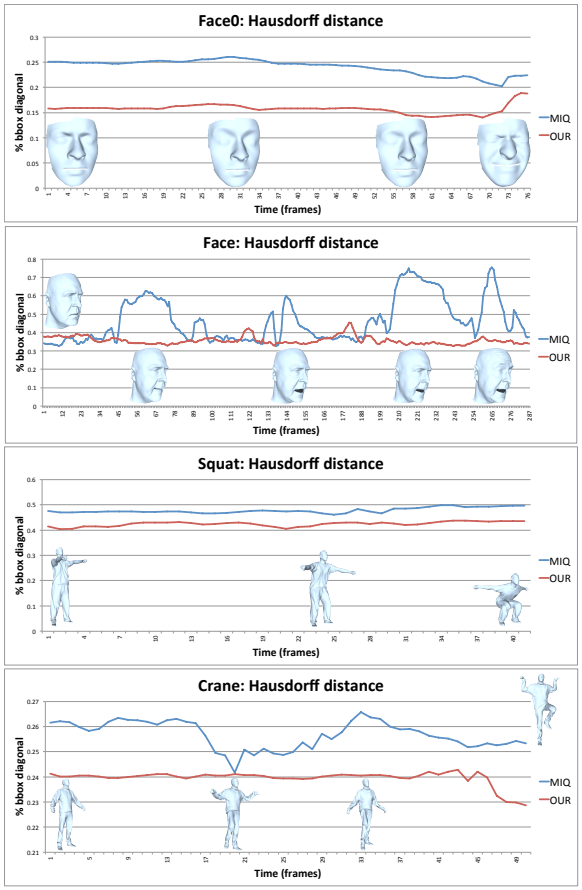

Figure 4: *Hausdorff distance between the input mesh and the quad mesh computed with our method (red) and MIQ (blue). Graphs show running measures along the sequence.*

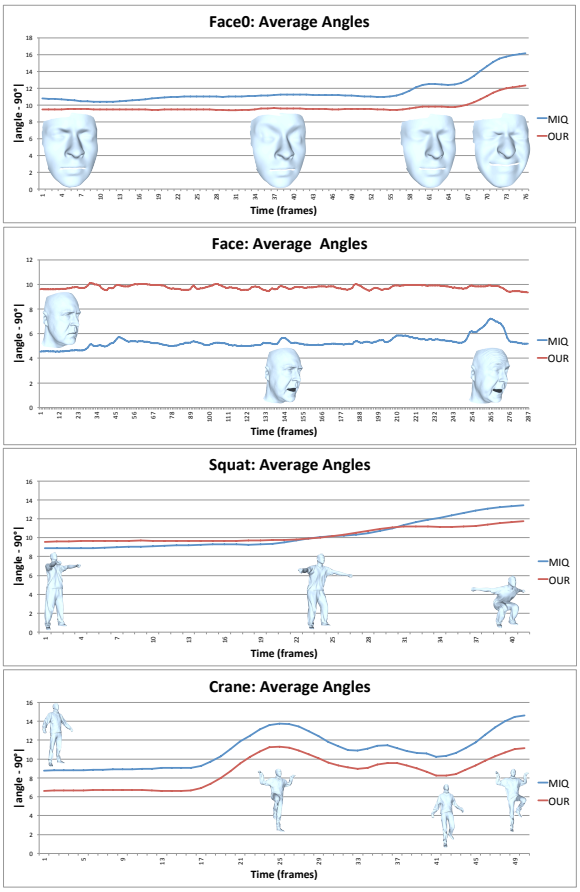

Figure 5: *Shape of elements in the quad mesh computed with our method (red) and MIQ (blue). The average difference in degrees between angles of quads and 90° is measured at each pose. Graphs show running measures along the sequence.*

dard MIQ, which have been computed by the curvature estimation method presented in [YLHP06] and the anisotropy formula given in [BZK09]. Stable constraints throughout the sequence are not shown, but they are mostly coincident with those ones in the figure, at the eyebrows and nose, while those at chin, lips, nasolabial fold, and tempe disappear.

In Figure 3 we show a comparison of the 4-RoSy fields obtained with our method and with plain MIQ, after smoothing, on the first pose of the same dataset. Note that in our case we did *not* use the hard constraints for this dataset. The corresponding quad meshes are shown in Figure 1 on a different pose in the sequence. While the field computed with MIQ is smoother, with fewer singularities, the corresponding quadrangulation results are quite artificial, especially in the zones of the cheek, between the upper lip and the nose, and around the eyebrows, where features involved in animation are not evident from the initial pose alone. On the contrary, our field follows facial features and muscle bundles much more naturally, getting closer to the structure of a manually designed mesh.

Figures 4 and 5 present statistics about our experiments on all datasets. We measured the Hausdorff distance, at each pose in the sequence, between the input mesh and the quad meshes computed with our method and with MIQ, respectively. The graphs in Figure 4 depict the running values along the sequence. In all cases, our method performs better than MIQ. On the Face dataset, MIQ meshes exhibit a slightly smaller error on the initial poses and in poses similar to the initial one - indeed the mesh computed with MIQ was optimized for smoothness in the first pose - but they are affected by much larger error in poses corresponding to large deformations, showing an error twice larger than ours in the worst cases. In the Face0, Squat and Crane sequences the two methods are quite stable, with ours showing a consistently smaller error for about 10%-40%. The graphs in Figure 5 evaluate the shape of elements of quad meshes, by measuring the absolute difference of angles of quads from 90 degrees. In the Face dataset, the MIQ performs consistently better than our method. This is no surprise, as MIQ imposes a highly regular grid over the face, while our method inserts

more singularities in order to better follow the flow of face features. Elements in the proximity of singularities are always affected by distortion in the parametrization, which induces deformation of quads. However, in the Face0 dataset, which contains less detail, our method also obtains a better alignment with a number of singularities comparable with MIQ mesh, and the resulting shape of quads is consistently better; in the last poses, which exhibit the largest mesh distortion, both methods show increasingly distorted angles. In the squat sequence MIQ performs better on the first poses, while our method improves and overcomes MIQ as deformation increases through the sequence. In the crane sequence both methods increase distortion with the largest deformations, with our method being always better than MIQ.

Figure 6 shows comparisons between quad meshes obtained with our method and with the standard MIQ computed on the initial pose of each sequence. Small figures depict the initial pose as well as some key frames of each sequence. On the Squat dataset, the top views show how the quad mesh obtained with MIQ, being computed on the first pose with forward-pointing arms, does not follow correctly the natural transition between shoulders and arms, thus causing a large torsion towards the end of the sequence. Our mesh, on the contrary, compensates for torsion in all poses of the sequence, thus resulting in the correct alignment. On the Crane dataset we have a similar situation. The meshes in the initial pose look quite similar, but upon large deformations, large torsions appear in the MIQ mesh in the inner side of legs (side view) and in the inner side of elbows (top view in the detail), while our mesh compensates very well deformation, always maintaining the correct alignment. More views of the Face and Face0 datasets shows how our meshing better captures the contenance of the jaw, eyebrows and protruding mouth, as well as other relatively static details such as earlobes.

5. Concluding remarks

We have presented a first method to automatically compute quad meshes for deforming shapes, which takes into account in advance the needs of animation. Our method produces quad meshes that are closer to those designed by artists, with respect to previous methods that compute quadrangulation from a single pose. This is especially evident for the case of soft objects without sharp features, like a human face, in which our method reveals the underlying muscle structure inferred from motion, while static methods just fit a rather artificial straight grid.

There are certainly a number of limitations of our work, most of which come from the standard technology we rely on in the late stages of our pipeline. Meshes designed by artists have several characteristics that cannot be obtained by standard quadrangulation methods: they may be largely anisotropic; they are adaptive, representing different parts at different resolutions, with a finer density of elements where the mesh undergoes more deformation; they may exploit shear and twist of elements to better adapt to certain deformations. All such requirements cannot be fulfilled if the quad mesh is required to follow an isotropic and orthogonal frame field. Recent results [LXW*11] have shown that using a more flexible framework based on conjugate fields can achieve better results in producing quad meshes for architectural design. We believe that an even more flexible approach could be pursued, by using a general frame field, which is just subject to a soft penalty to the amount of non-orthogonality in the optimization energy. Such a field can be initialized during the local analysis phase, by a non-linear optimization, to find the best pair of directions on M_0 to remain as orthogonal as possible throughout animation. Since only few variables and data are involved in local analysis, even non-linear optimization should work in reasonable time. Of course, the optimization framework used in the smoothing and quadrangulation phase must be modified, similarly to [LXW*11], in order to deal with general frame fields.

Further extensions are possible. A quadratic model could be used during the local analysis phase, such as in [MHTG05], in order to estimate also bending and twist, which may be useful to set a sizing field for the quadrangulation. Mesh symmetrization [PLPZ12] could also be combined seamlessly with our framework to improve results further, especially in the case of faces and human characters. Using the entire sequence to detect symmetry more robustly is also an interesting venue for future work.

Acknowledgments

We wish to thank the authors of [VBMP08] for making data of the Crane and Squat sequences available on the web (http://people.csail.mit.edu/drdaniel/mesh_animation/). We wish to thank the authors of [BHPS10] and [BHB*11] for kindly providing the two face sequences. The research leading to these results is partly funded by the EU Community's FP7 ICT under the V-MusT.net Project (Grant Agreement 270404), the SNF award 200021_137879, ERC grant iModel (StG-2012-306877) and a gift from Adobe Research.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. In *ACM SIGGRAPH 2003 Papers* (2003), SIGGRAPH '03, pp. 485–493. 2

[BBB*10] BEELER T., BICKEL B., BEARDSLEY P., SUMNER B., GROSS M.: High-quality single-shot capture of facial geometry. *ACM Trans. Graph. 29* (July 2010), 40:1–40:9. 2

[BHB*11] BEELER T., HAHN F., BRADLEY D., BICKEL B., BEARDSLEY P., GOTSMAN C., SUMNER R. W., GROSS M.: High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30* (August 2011), 75:1–75:10. 2, 5, 7

[BHKH13] BUDD C., HUANG P., KLAUDINY M., HILTON A.: Global non-rigid alignment of surface sequences. *Int. J. Comput. Vision 102*, 1-3 (Mar. 2013), 256–270. 2

[BHLW12] BOJSEN-HANSEN M., LI H., WOJTAN C.: Tracking surfaces with evolving topology. *ACM Trans. Graph. 31*, 4 (July 2012), 53:1–53:10. 2

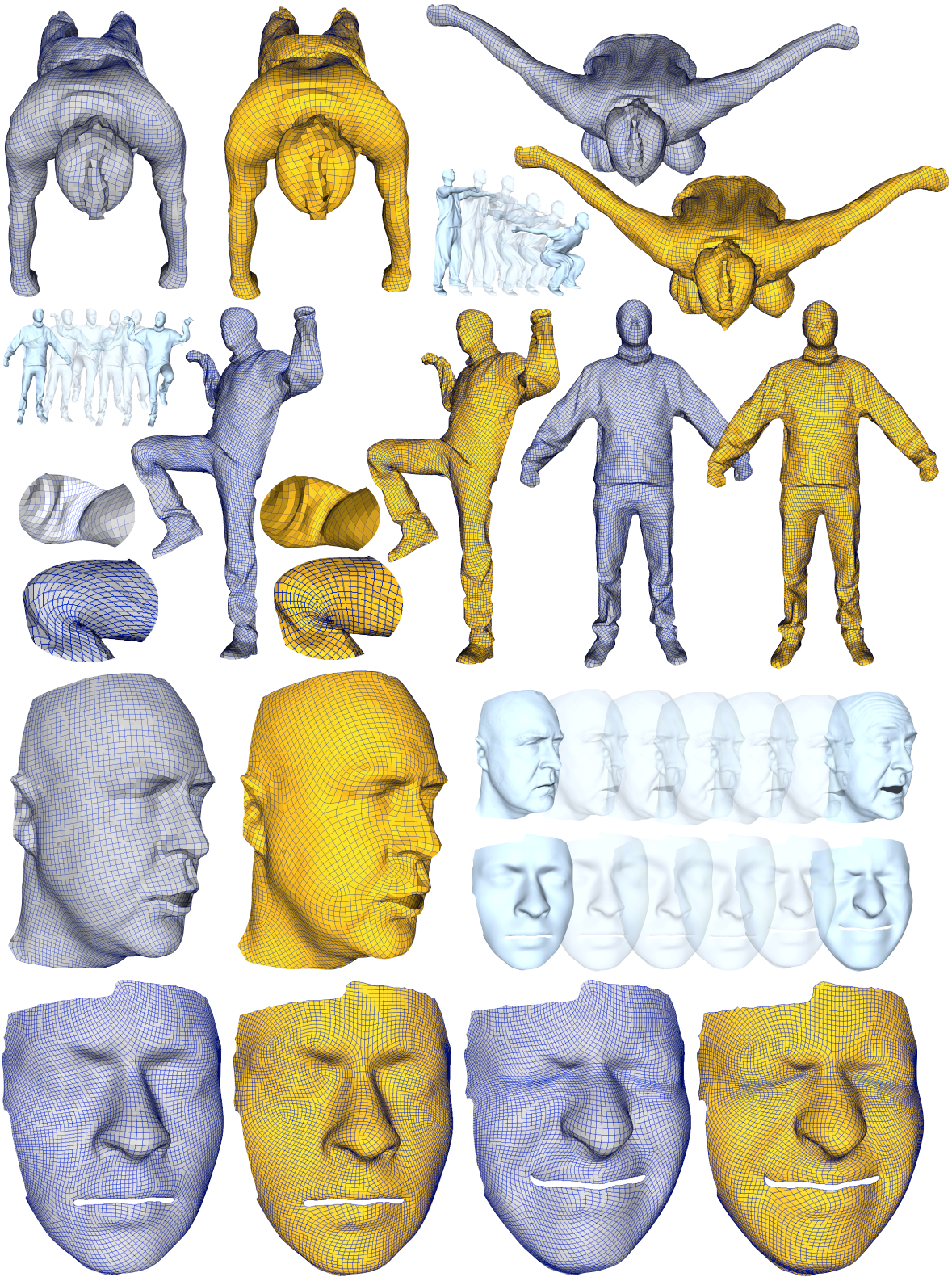

Figure 6: *Results on all datasets, comparison between our method (yellow) and plain MIQ (grey).*

[BHPS10] BRADLEY D., HEIDRICH W., POPA T., SHEFFER A.: High resolution passive facial performance capture. *ACM Trans. Graph. 29*, 4 (July 2010), 41:1–41:10. 2, 5, 7

[BLP*13] BOMMES D., LÉVY B., PIETRONI N., PUPPO E., SILVA C., TARINI M., ZORIN D.: Quad mesh generation and processing: A survey. *Computer Graphics Forum* (2013). To appear, published online. doi:10.1111/cgf.12014. 2

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (July 2009), 77:1–77:10. 1, 2, 3, 4, 6

[Eig] Eigen template library for linear algebra. URL: http://eigen.tixfamily.org. 5

[GFT*11] GHOSH A., FYFFE G., TUNWATTANAPONG B., BUSCH J., YU X., DEBEVEC P.: Multiview face capture using polarized spherical gradient illumination. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 129:1–129:10. 2

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), SIGGRAPH '00, pp. 517–526. 2

[IR] Infinite-realities. URL: http://ir-ltd.net. 2

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

[KNP07] KÄLBERER F., NIESER M., POLTHIER K.: Quadcover - surface parameterization using branched coverings. *Comput. Graph. Forum 26*, 3 (2007), 375–384. 2

[LLV*12] LI H., LUO L., VLASIC D., PEERS P., POPOVIĆ J., PAULY M., RUSINKIEWICZ S.: Temporally coherent completion of dynamic shapes. *ACM Trans. Graph. 31*, 1 (Feb. 2012), 2:1–2:11. 2

[LXW*11] LIU Y., XU W., WANG J., ZHU L., GUO B., CHEN F., WANG G.: General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 140:1–140:10. 7

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (July 2005), 471–478. 7

[PLPZ12] PANOZZO D., LIPMAN Y., PUPPO E., ZORIN D.: Fields on symmetric surfaces. *ACM Trans. Graph. 31*, 4 (July 2012), 111:1–111:12. 3, 4, 7

[PZ07] PALACIOS J., ZHANG E.: Rotational symmetry field design on surfaces. *ACM Trans. Graph. 26*, 3 (2007). 2

[RLL*06] RAY N., LI W. C., LÉVY B., SHEFFER A., ALLIEZ P.: Periodic global parameterization. *ACM Trans. Graph. 25* (October 2006), 1460–1485. 2

[RVAL09] RAY N., VALLET B., ALONSO L., LEVY B.: Geometry-aware direction field processing. *ACM Trans. Graph. 29* (December 2009), 1:1–1:11. 2

[RVLL08] RAY N., VALLET B., LI W. C., LÉVY B.: N-symmetry direction field design. *ACM Trans. Graph. 27*, 2 (May 2008), 10:1–10:13. 2, 3, 4

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

[TG] Triplegangers. URL: http://www.triplegangers.com. 2

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 97:1–97:9. 2, 5, 7

[VCG] The vcg library. URL: http://vcg.sourceforge.net. 5

[VPB*09] VLASIC D., PEERS P., BARAN I., DEBEVEC P., POPOVIĆ J., RUSINKIEWICZ S., MATUSIK W.: Dynamic shape capture using muN̆lti-view photometric stereo. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 174:1–174:11. 2

[YLHP06] YANG Y.-L., LAI Y.-K., HU S.-M., POTTMANN H.: Robust principal curvatures on multiple scales. In *SGP 2006: 4th Eurographics Symposium on Geometry processing* (2006), Polthier K., Sheffer A., (Eds.), Eurographics Association, pp. 223–226. 6

[ZB] Zbrush. URL: http://pixologic.com. 2