ICIC Express Letters
Part B: Applications
Volume **7**, Number **7**, July **2016**

ICIC International ©2016 ISSN 2185-2766
pp. 1415–1421

ANIMATION QUADRANGULATION ON SYMMETRIC SURFACES

Ping Du[1,2] and Changhe Tu[1]

[1]School of Computer Science and Technology
Shandong University
No. 1500, Shunhua Road, Jinan 250101, P. R. China
haidou2009@qq.com; chtu@sdu.edu.cn

[2]School of Computer Science and Technology
Shandong Institute of Business and Technology
No. 191, Binhai Road, Yantai 264005, P. R. China

Received November 2015; accepted February 2016

Abstract. *We study the problem of generating the quad meshes of symmetric surfaces from triangle mesh sequences with high quality. We present a novel method to optimize the cross field in the first stage of quadrangulation pipeline to strike a balance between symmetry and animation. Our method is semi-automatic, which needs a few input symmetry points by the user. The experimental results we obtained show better in Hausdorff distance and average angle compared to the state-of-the-art algorithms, while the singularity points are less and the visual aesthetics are improved better.*

Keywords: Mesh generation, Quadrangulation, Animation, Symmetric surfaces, Cross field

1. Introduction.

Quad mesh generation remains a topic of great interest and importance. This is especially true as the proliferation of dynamic geometry data acquisition equipment progresses. For a number of different poses from the same object, powerful geometric processing methods are required to refine them into high quality quad meshes which are suitable for further use. Usually, the majority of dynamic data are from human beings and animals, who naturally have symmetric property. Quad meshes with symmetry are visually preferable and reduce deformation artifacts. In a standard production pipeline, an artist who generates the mesh must take into account both animation and symmetry, which is useful for the following production.

Field-guided parametrization-based quad meshing methods for animation such as animation-aware quadrangulation [1] have proven to be powerful. The pipeline typically follows a three-stage approach as follows. Firstly, a cross field is constructed on the reference of input surfaces which defines guiding information for the quad element orientation as well as singularity placement. Secondly, the reference surface is parametrized into an integer grid map [2] so that the canonical integer grid in the parameter domain induces a quad mesh on the input geometry. Thirdly, quad meshes to all the other surfaces are deformed from the reference one. However, presently, no method can produce quad meshes with regard to both animation and symmetry. So, it is very meaningful to introduce the symmetry property into the animation-aware quadrangulation process.

2. Related Work.

Cross field is constituted by four coupled vector fields, which can explicitly show local properties of surface. Cross field is first mentioned in [3]. Cross field can be either designed manually or generated automatically. Cross field exhibits the same type of singularities with quad mesh, so the generation of a highly regular quad mesh is strongly related to the generation of cross field.

All the methods generate the smoothest cross field potentially subject to certain constraints. Sometimes the constraints are selected by the user [2]; sometimes they are computed by some heuristics [4]. [4] explored an algorithm for the robust computation of symmetry maps for surface. [1] explored an algorithm for the synthetic sequences for a rough animation of a triangulated model. Since none of the methods combine seamless with symmetry and animation to improve results further especially for the animation of human beings and animals.

Quadrangulation was discussed in the survey [5, 6]. Our solution applies to the class of field-guided parametrization-based methods, first proposed in [7], then improved in [2, 8, 9, 10]. Apart from the pure guidance point of view, the field guided methods decompose the quad mesh generation problem into two sub-problems, cross field generation and global parametrization. Such things provide convenience for processing the problem of mesh optimization.

For the quadrangulation of animation meshes, state-of-the-art methods also decompose the problem into two sub-problems, parametrization for reference frame and mapped to all the others. This class of methods consider the results of parametrization as texture of surfaces, which can be easily mapped to other frames and then get the final quad meshes with point-to-point correspondence. In this paper, the framework of our method is followed by the way above.

Symmetry detection methods was discussed in the survey [11]. Symmetry can be defined by considering self-isometrics of a surface with respect to a metric of the surface, either intrinsic or extrinsic. Intrinsic symmetry preserves geodesic distance, while extrinsic symmetry preserves Euclidean distance. Kim et al. discussed the problem of intrinsic symmetry map in [12, 13]. Animation causes the changes of pose, so it is not possible to have perfectly isometric extrinsic symmetries during animation. To the best of our knowledge, various recent works deal with symmetric detect or animation quad meshes generation, but none handle the optimization of the quality of the animation quad meshes with symmetric property.

In this paper, we present a novel method to optimize the cross field in the first stage of pipeline to strike a balance between symmetry and animation. Our experimental results show that the generated quad meshes adopt well to both symmetry and animation, which can significantly improve visual aesthetics and obtain less singularities. The results we obtained also show better in Hausdorff distance and average angle compared to the state-of-the-art generation algorithms.

3. Animation Quadrangulation on Symmetric Surfaces.

3.1. Algorithm outline.

Given an input sequence $M_0, \dots, M_k$ of triangle meshes with point-to-point correspondence, our method consists of the following steps (see Figure 1):

1. Construction of generalized bilateral intrinsic symmetry of the surface, stationary line and transport;

2. Analysing all the deformations of triangle meshes during animation;

3. Combining seamlessly with the results of Step 1 and Step 2 for the final cross field generation;

4. Parametrization and quad mesh extraction.

3.2. Symmetry detection.

Our method is semi-automatic symmetry detection algorithm. So, we have to give a small number of user-defined correspondences first. Different from the state-of-the-art methods just for static triangle mesh, our method allows user to choose point-to-point correspondences during animation, as the registration of the meshes. All the input of the correspondences are fit for any one of the meshes in the animation.

Without loss of generality, we select mesh M_0 as the reference mesh. $(p_i, q_i) \in (M_0, M_0)$, $i = 1, \dots, n$ are the input pairs of symmetric landmarks. Then we construct the generalized bilateral intrinsics symmetry similar to the algorithm of [4].

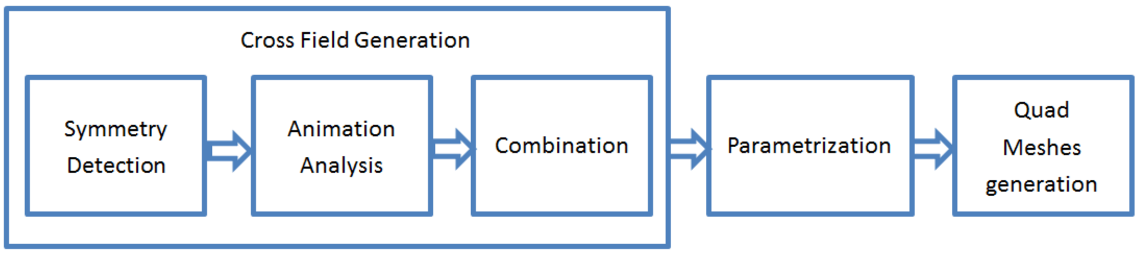

FIGURE 1. Algorithm flow chart

Firstly, we calculate the conformal map $\varphi : M \to \widehat{C}$ of the reference mesh to the plane, and then transform the landmarks to the extended complex plane $z_i = \varphi(p_i)$ and $w_i = \varphi(q_i)$, $i = 1, \ldots, n$.

Secondly, we find the anti-involution Möbius $\widehat{m}(z)$ minimizing the deviation of $\tilde{m}(z)$ from w_i, $i = 1, \ldots, n$. Here,

$$\widehat{m}(z) = \frac{a\bar{z} + b}{c\bar{z} - \bar{a}}, \ (b, c \in \mathbb{R}) \tag{1}$$

For a proof see [4], we can get that an anti-Möbius transformation $\widehat{m}(z) = \dfrac{a\bar{z} + b}{c\bar{z} + d}$ is an anti-involution iff the matrix of coefficients $\begin{pmatrix} c & d \\ -a & -b \end{pmatrix}$ is hermitian, that is $d = -\bar{a}$ and $b, c \in \mathbb{R}$.

The stationary circle $\mathbb{C}$ of the anti-involution can be defined by $\widehat{m}(z) = z$.

Thirdly, we transfer the circle $\mathbb{C}$ to the real axis though a Möbius transformation m. Here, the stationary circle of $\widehat{m}$ has the explicit equation as follows,

$$|z - z_0| = r^2, \ \left(z_0 = \frac{a}{c}, r^2 = \left|\frac{a}{c}\right| + \frac{b}{c} \right) \tag{2}$$

For there are three degrees of freedom for the Möbius transform mapping it to the real axis, so we just select three equal distant points t_i $(i = 1, 2, 3)$ on the circle $\mathbb{C}$ to find the Möbius transformation m.

$$at_i + b = (ct_i + d)x_i, \ (i = 1, 2, 3; x_i \in \mathbb{R}) \tag{3}$$

Fourthly, we map the landmarks z_i, w_i with m and extract optimal symmetric points in the least-squares sense and use smooth deformation ψ to move the landmarks to their optimal symmetric configurations. Here,

$$\psi(m(z_i)) = \overline{\psi(m(w_i))} \tag{4}$$

Finally, our final map is $g = \psi \circ m \circ \varphi$, the symmetry map in these new coordinates is $z \to \overline{z}$ and the transport and stationary line can be computed as follows:

Transport of Cross field R_g. To define cross field that fit the symmetry map g, we should compare the values of the field at symmetric points. The differential Dg defines a natural map from a given point p to $g(p)$. We compute the Singular Value Decomposition (SVD for short) of Dg

$$Dg = U\Sigma V^T \tag{5}$$

and get $R^g = UV^T$, which is the closest orthonormal transform to the differential Dg.

Stationary Line $\mathbb{C}$. Because of the symmetry, if v is a symmetric cross field, and p is a point on stationary line, we must get $R^g v(p) = v(p)$. So it must be one of the following situations: (1) p is a singularity point; (2) one of the directions of v is the stationary direction s_p of R^g_p; (3) one of the bisectors of angles formed by consecutive vectors of v is aligned with s_p.

3.3. **Animation analysis.** Our method is automatic animation analysis algorithm, which presented in [1] first. We use the selected mesh M_0 in Section 3.2 as the reference mesh. The steps are as follows:

Local deformation analysis. For every triangle $t = (a, b, c)$, the local x-axis is aligned with b-a, z-axis is aligned with the normal vector of triangle and the origin point at a.

The deformation gradient can be presented by the affine transformation from triangle of M_0 to its corresponding triangle of M_i one by one. We use J_i^j as the deformation gradient of t_i in M_j. For we consider the only component of deformation, we treat the triangles in a common XY 2D-coordinate axis. So J_i^j can be obtained by solving a 2×2 linear computation.

Then, we compute the SVD of J_i^j.

$$J_i^j = U_i^j \Sigma_i^j V_i^j \tag{6}$$

From the knowledge of SVD, we know that the columns of V are principal directions under transformation of J, which are always orthogonal. So we use θ and $\theta + \pi/2$ to present the columns of V relatived to the local 2D coordinate system of the triangle. The singular values s_1 and s_2 in the diagonal matrix Σ represent the stretching induced by J along the directions. We define the stretch factor s_i^j as $|s_1| / |s_2| - 1$.

Animation deformation analysis. In order to obtain the cross field that fit the animation, we compute the average $\bar{\theta}$ as follows:

$$\bar{\theta}_t = \frac{1}{4}\Phi\left(\frac{\Sigma_i s_t^i \Psi\left(4\theta_t^i\right)}{\Sigma_i s_t^i}\right) \tag{7}$$

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

$$\bar{s}_t = \frac{1}{\Sigma_{i=1}^k s_t^i} \sum_{i=1}^{k} s_t^i s_t^i \tag{8}$$

3.4. **Combination.** From the above steps, we get both symmetry and animation analysis of the meshes. Now we compute the cross field that strikes a balance of the two. Our method consists of the following steps:

1. Set hard constraints at the stationary line and set animation deformation as soft constraints, and then extend field to the rest of M_0 by running MIQ smoothing algorithm. We can express the energy of the cross field as follows:

$$E_{smooth} = (1-\alpha)\sum_{e_{i,j}\in E}\left(\theta_i + k_{i,j} + \frac{2\pi}{4}p_{i,j} - \theta_j\right)^2 + \alpha\sum_{t\in F}\bar{s}_t\left(\theta_t - \bar{\theta}_t\right)^2 \tag{9}$$

Here, $k_{i,j}$ is the fixed rotation angle between the local frames of t_i and t_j. $p_{i,j}$ is an integer variable, called the period jump. α is used to allow the user trading off between smoothness and soft constraints.

2. Use field transport R^g to symmetrize field v by averaging over orbits O, and the results are $\bar{v}$. The orbit of t is defined as the union of all triangles in its 1-ring and the 1-ring of the triangle containing $g(c)$. We define a weight $s_i(t)$ to every $t_i \in O(t)$ inversely proportional to its geodesic distance from $O(c) = \{c, g(c)\}$.

$$s_i(t) = \Phi\left(\min_{c'\in O(c)} \|c_i - c'\|\right) \tag{10}$$

Here, Φ is a Gaussian with standard deviation equals the maximum of the triangles' 1-ring diameter. Then $s_i(t)$ is normalized to have a unit sum for the final computation.

3. Repeat Step 1 to obtain the final cross fields v using $\bar{v}$ as soft constraints.

3.5. **Parametrization and quad mesh extraction.** We use state-of-the-art methods to complete the following steps. First, parametrization of the reference mesh can be computed by mixed-integer quadrangulation method [2]. Then, the quadrangular mesh is mapped to all the other meshes. Finally, we extract all the meshes with [14], producing the sequence of quad remeshings.

4. **Results and Comparison.** Our framework has been implemented in C++, by using the libigl library [15] and CoMISo library [16] for mesh processing, and the Eigen library [17] for numerical computation. At last, we use libQEx [14] to implement quad extraction.

In Figure 2, we show a comparison of bouncing dataset with our method and [1] method (AAQ method for short). Our result is on the left, and AAQ result is on the right. It is obvious that quad meshes computed with our method show better nature and symmetric properties during the animation. Not only that, the number of singularities is also obviously decreased. We think the results meet our expectations, for when not considering the situation of symmetry, we could inevitably take something just like the clothes fold as singularity. However, this situation can be optimized by our processing. In Table 1, we show the comparison of singularity numbers.

In Figure 3 and Figure 4, we show the comparisons of Hausdorff distance and average angle with our method and AAQ method at each pose in the sequence. In all the graphs, x-axis is frames, which means time. Y-axis is average angle for the left and Hausdorff distance for the right. Our method has an obvious improvement on average angle in most cases. The gaps of Hausdorff distance are not very clear all the time, but our method is still better than AAQ method on them.

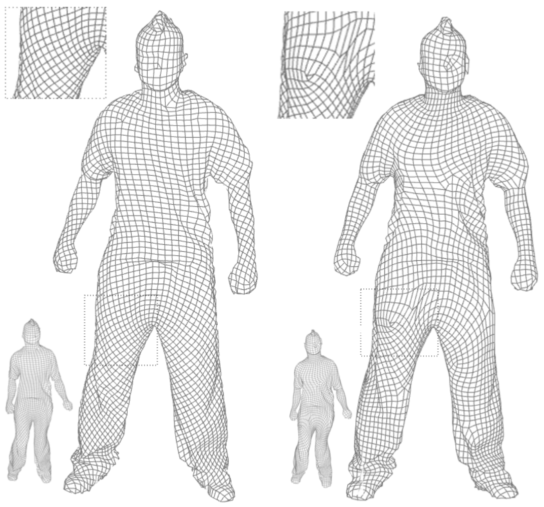

FIGURE 2. Quad meshes of bouncing datasets computed with our method (left) and AAQ method (right)

TABLE 1. Comparison of singularity numbers

	Our method			AAQ method		
Models	Total points	Singularity points	Proportion	Total points	Singularity points	Proportion
bouncing	5195	88	1.68%	5084	136	2.68%
handstand	5448	86	1.58%	4925	185	3.76%
squat1	5214	73	1.4%	4901	112	2.29%
squat2	5418	105	1.92%	5159	174	3.37%
swing	5367	81	1.51%	5009	139	2.78%

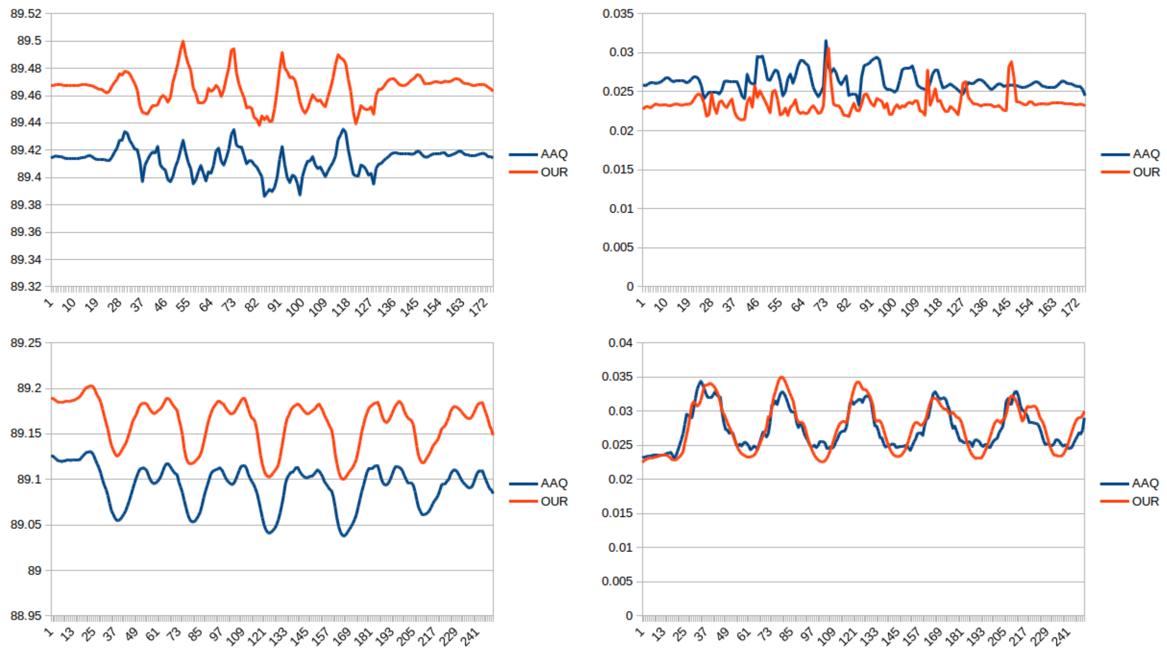

FIGURE 3. Average angle (left) and Hausdorff distance (right) difference between our method (red) and AAQ methed (blue) of the samba1 dataset (up) and squat1 dataset (down)

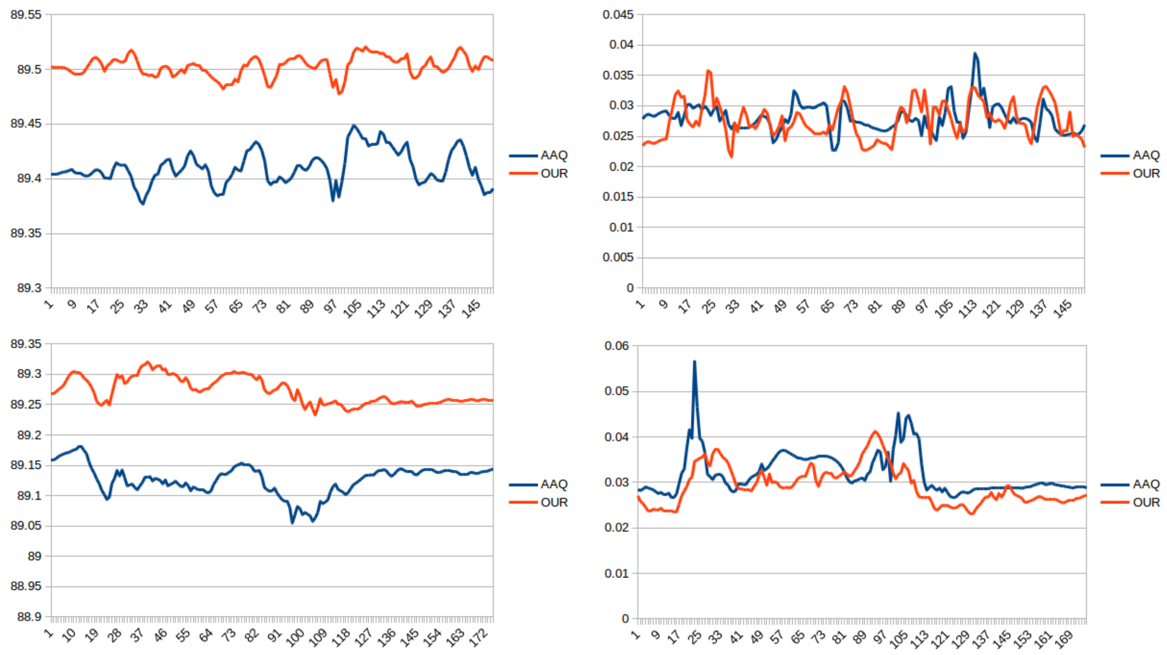

FIGURE 4. Average angle (left) and Hausdorff distance (right) between our method (red) and AAQ methed (blue) of the swing dataset (up) and squat2 dataset (down)

5. **Conclusion and Future Work.** We presented a novel method to generate the quad meshes of symmetric surfaces of animation. Our method is the improvement of the methods of [1, 4]. The key idea of our method is to strike a balance between symmetry and animation which makes the cross field with high quality. All the improvements are computed in the first stage of the pipeline. And the method is semi-automatic. The experimental results we get show better than state-of-the-art method not only in the visual

aesthetics but also in the data analysis, such as singularity numbers, Hausdorff distance and average angle.

Our method still has some limitations. It is not fully automatic, which needs a few user interactions. While we didn't make good use of the characteristics of the animation sequence to complete the symmetry judgement, which is one of our future exploration directions. On the other hand, the limitation is about the input data. The input needs to be a sequence of triangle meshes with point-to-point correspondence, and the next step we will try to introduce the improvement of the method just like we used in this symmetry judgement to the registration of any kind of input sequences for better robustness.

Acknowledgements. We would like to thank Daniele Panozzo and Alec Jacobson for the geometry processing framework libigl [15], the authors of [19] for making data of the sequences of triangle meshes on the website [18], David Bommes for providing his mixed-integer solver [2, 16, 20], Hans-Christin Ebke for providing the code of the libQEx quadrangulator [14].

This work was supported in part by the National Natural Science Foundation of China No. 61272242 and No. 61373079. This work was also supported by the Youth Scientific Research Foundation of Shandong Institute of Business and Technology No. 2015QN002 and Shandong Province Natural Science Foundation No. ZR2013FL018.

REFERENCES

[1] G. Marcias, N. Pietroni, D. Panozzo et al., Animation-aware quadrangulation, *Computer Graphics Forum*, vol.32, no.5, pp.167-175, 2013.
[2] D. Bommes, H, Zimmer, L. Kobbelt et al., Mixed-integer quadrangulation, *ACM Trans. Graphics*, pp.1-10, 2009.
[3] A. Hertzmann and D. Zorin, Illustrating smooth surfaces, *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pp.517-526, 2000.
[4] D. Panozzo, Y. Lipman, E. Puppo et al., Fields on symmetric surfaces, *ACM Trans. Graphics*, vol.31, no.4, p.111, 2012.
[5] D. Bommes, B. Lévy, N. Pietroni et al., Quad-mesh generation and processing: A survey, *Computer Graphics Forum*, vol.32, no.6, pp.51-76, 2013.
[6] D. Panozzo, Demystifying quadrilateral remeshing, *IEEE Computer Graphics & Applications*, vol.35, no.2, pp.88-95, 2015.
[7] N. Ray, W. C. Li, B. Lévy et al., Periodic global parameterization, *ACM Trans. Graphics*, vol.25, no.4, pp.1460-1485, 2006.
[8] N. Ray, B. Vallet, W. C. Li et al., N-symmetry direction field design, *ACM Trans. Graphics*, vol.27, no.2, p.10, 2008.
[9] N. Ray, B. Vallet, L. Alonso et al., Geometry-aware direction field processing, *ACM Trans. Graphics*, vol.29, no.1, 2009.
[10] F. Kaelberer, M. Nieser and K. Polthier, QuadCover – Surface parameterization using branched coverings, *Computer Graphics Forum*, vol.26, no.3, pp.375-384, 2007.
[11] N. J. Mitra, P. Mark, W. Michael et al., Symmetry in 3D geometry: Extraction and applications, *Computer Graphics Forum*, vol.32, no.6, pp.1-23, 2013.
[12] V. G. Kim, Y. Lipman, X. Chen et al., Möbius transformations for global intrinsic symmetry analysis, *Computer Graphics Forum*, vol.29, no.5, pp.1689-1700, 2010.
[13] V. G. Kim, Y. Lipman and T. Funkhouser, Blended intrinsic maps, *ACM SIGGRAPH Annual Conference*, pp.76-79, 2011.
[14] H. Ebke, D. Bommes, M. Campen et al., QEx: Robust quad mesh extraction, *ACM Trans. Graphics*, vol.32, no.6, pp.2504-2507, 2013.
[15] *The Libigl Library*, http://www.inf.ethz.ch/personal/dpanozzo/libigl_tutorial/tutorial.html.
[16] *CoMISo: Constrained Mixed-Integer Solver*, http://www.graphics.rwth-aachen.de/software/comiso.
[17] *Eigen Template Library for Linear Algebra*, http://eigen.tuxfamily.org.
[18] *http://people.csail.mit.edu/drdaniel/mesh_animation.*
[19] D. Vlasic, I. Baran, W. Matusik et al., Articulated mesh animation from multi-view silhouettes, *ACM Trans. Graphics*, vol.27, no.3, 2008.
[20] D. Bommes, H. Zimmer and L. Kobbelt, Practical mixed-integer optimization for geometry processing, *Curves and Surfaces*, pp.193-206, 2012.