# Analysis of 3D Mesh Correspondences Concerning Foldovers

Johannes Merz
TU Darmstadt
Darmstadt, Germany
johannes.merz@gris.tu-darmstadt.de

Roman Getto
TU Darmstadt
Darmstadt, Germany
roman.getto@gris.tu-darmstadt.de

Tatiana von Landesberger
TU Darmstadt
Darmstadt, Germany
tatiana.von.landesberger@gris.tu-darmstadt.de

Dieter W. Fellner
TU Darmstadt & Fraunhofer IGD,
Darmstadt, Germany
dieter.fellner@gris.tu-darmstadt.de

## ABSTRACT

Foldovers (i.e., folding of triangles in a 3D mesh) are artifacts that cause problems for morphing. Mesh morphing uses vertex correspondences among the source and the target mesh to define the morphing path. Although there exist techniques for making a foldover-free mesh morphing, identification and correction of foldovers in existing correspondences is still an unsolved issue.
This paper proposes a new technique for the identification and resolution of foldovers for mesh morphing using predefined 3D mesh correspondences. The technique is evaluated on several different meshes with given correspondences. The mesh examples comprise both real medical data and synthetically deformed meshes. We also present various possible usage scenarios of the new algorithm, showing its benefit for the analysis and comparison of mesh correspondences with respect to foldover problems.

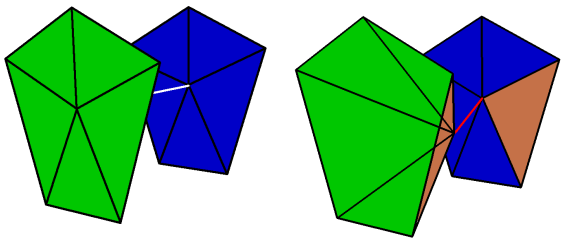
## Keywords

Foldover, Correspondence, Morphing, Mesh Comparison

## 1 INTRODUCTION

Mesh morphing is commonly used in various computer graphic applications for interactively showing animated correspondence between two meshes – the source mesh and the target mesh. Mesh morphing can be described as the continuous deformation from a graphical object to another one [Gom99]. During the deformation the points from the first mesh move to their corresponding points on the second mesh along the correspondence path.

Morphing methods using linear paths between mesh correspondences can show problems if the source triangle folds over during the morphing process. It means that a triangle $t$ with a positively-oriented area is mapped to a triangle $f(t)$ with a negatively-oriented area [Ebk+13] (see Figure 1). Foldovers often occur, for example, when morphing between complex meshes like an automatically segmented and expert-segmented liver from 3D medical images. Foldovers can produce unnaturally looking morphing sequences. Moreover, the foldover problem can lead to undesirable blending of neighboring textures when using texture mapping algorithms. Therefore, we need to identify and to resolve such foldovers.



(a) No foldover. (b) Foldover.

Figure 1: **(a):** no foldover. During the morphing from the source mesh (blue) to the target mesh (green) the center point does not leave the neighboring region. However in **(b)** this is the case, causing the positive area of the brown triangle to become negative: a foldover.

Mesh morphing may rely on point-to-point mesh correspondences, which are used for defining morphing paths. There are various methods for defining mesh correspondences [Tam+13; Tam+14; van+11]. One possibility is to use the correspondences determined by local mesh distance measures such as the surface distance or the extended surface distance [CRS98; GJC01; GKL15] . The (Extended) Surface Distance relates one

point of the first mesh to one point of the second mesh and thus creates correspondences. However, not all distance measures can be used for determining correspondences. For example, the Fréchet Distance [VH01] is based on a parameterization and thus requires preknown correspondences.

Depending on the method of determining the correspondences, the points on the meshes that are associated with each other may differ and often the quality is application-dependent. The differences between sets of correspondences can lead to qualitatively different morphing results. Thus, we also need to analyze and to compare correspondences with regard to foldover problems.

We would like to emphasize, that we focus on identification and solution of foldovers for predefined correspondences. Although there are techniques that can produce foldover-free mesh morphing [MZX14] they cannot use the existing correspondences. The usage of correspondences, however, is required in many cases such as the assessment of segmentation quality or mesh registration quality in 3D medical image segementation applications. As current foldover identification methods are not suitable for such 3D mesh correspondence-based problems (see Section 2), the identification and correction of foldovers for a given set of correspondences remains an open research question.

## Contribution & Application Benefit

We propose a new technique for identifying foldovers given a mesh and correspondences to another mesh (see Section 3). This enables the evaluation of correspondences concerning foldovers as well as the comparison of different sets of correspondences for a given mesh. Moreover, we automatically determine corrections to the foldover errors. The hereby obtained correction can be used to draw inferences about the used correspondence estimation method.

Our method can be applied to the analysis of 3D mesh correspondences in the evaluation of 3D medical image segmentation results, to the evaluation of surface distances creating correspondences, to the assessment of mesh deformations or to enable texture-enabled morphing using predefined correspondences. In the evaluation section, we show the benefit of our method for the first two usage scenarios (see Section 4).

# 2 RELATED WORK

This section discusses related work in the area of foldover handling. Foldover handling is most present in areas such as mesh morphing, texture mapping or mesh merging.

Regarding mesh morphing, Fujimura and Makarov [FM98] developed a technique for foldover-free image warping. They propose an approach, that uses a time-dependent triangulation of the image. The triangulation of the image is changed, when the foldover occurs. This happens, when a point leaves its surrounding polygon during warping. At this point in time the structure of the triangulation changes by removing unneeded edges and inserting new ones. Figure 2 shows an example of their approach. The moments of the foldover events can be precalculated by computing the intersection of the correspondence path. Although it produces foldover-free warping despite of erroneous correspondences, it does not represent a method to detect and correct foldovers in existing correspondences. This merely avoids foldovers. Furthermore it is developed for 2D images, not 3D meshes.

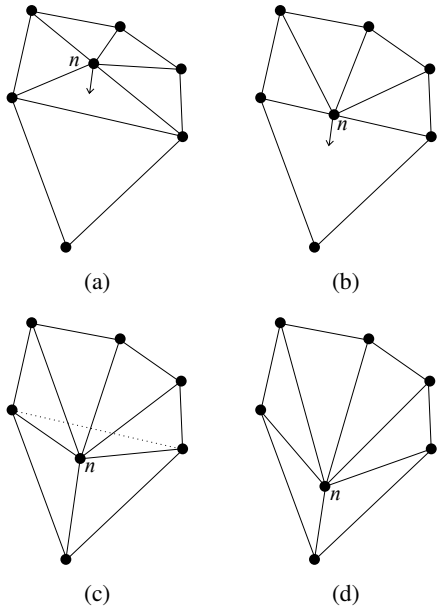



Figure 2: During the image warping the point $n$ moves along its correspondence path and leaves its surrounding polygon. A foldover occurs.

Lee et al. [LYY08] used a similar approach for addressing the foldover problem in texture mapping. To resolve foldovers they also use edge swaps at precalculatable moments.

Also working on foldover-free image warping, Ma et al. [MZX14] presented a technique, that maintains the triangulation of the source mesh. Instead of precalculating the foldover events along the correspondence path, their technique uses the help of radial basis functions (RBF). The first main step is to prepare the trajectories (correspondence paths) $\mathbf{C}_i$ for the warping. The $\mathbf{C}_i$ are piecewise linear polylines that are constructed iteratively out of the nodes $C_i^j, j = 0,1,\ldots,m$, where $C_i^0$ is the source point and $C_i^m$ is the corresponding point: $\{C_i^0\} \to \{C_i^1\} \to \cdots \to \{C_i^m\}$. Moreover they may not intersect. Following, the interpolation is executed over the span $\{C_i^j\} \to \{C_i^{j+1}\}$ with a stepsize

$\delta^{j,l}$. To ensure local bijectivity, the stepsize has to be a compromise of the RBF stepsize and the maximum stepsize until a foldover occurs. Other than in [FM98] this method does not adapt the triangulation to avoid foldovers, but the correspondence paths. Thus it also avoids foldovers during the warping process and is no method to analyze the data and perform corrections. Furthermore it cannot be precalculated.

Mocanu et al. [MTT13] developed a technique for mesh deformations that also makes use of radial basis functions. They observed, that an increased number of intermediate steps reduces the amount of foldovers. Similar to the previously mentioned techniques it aims at avoiding foldover rather than correcting it as well.

Looking into mesh merging, Alexa addressed the foldover problem [Ale00; Ale02]. He proposed an algorithm to merge genus 0 (without holes) meshes. While merging meshes, it is possible to introduce foldovers in the resulting mesh. The following function is used for the mapping:

$$f(x) = \begin{cases} \mathbf{x} + c(d - \|\mathbf{x}-\mathbf{v}\|)(\mathbf{w}-\mathbf{v}) & \|\mathbf{x}-\mathbf{v}\| < d \\ \mathbf{x} & \|\mathbf{x}-\mathbf{v}\| \geq d \end{cases} .$$

The function describes a point $\mathbf{x}$ moving from $\mathbf{v}$ to $\mathbf{w}$. If the mapping results in foldovers, the factor $c = 0.5$ can be adapted to modify the influence of both meshes on the result. Altering $d$ changes the influence radius, resulting in smaller steps. Although the proposed algorithm works with 3D meshes, it again aims at avoiding foldovers during the merging process.

The discussed work proposes multiple ways of handling the foldover problem, but none focuses on identifying existing foldovers and suggesting a corrected correspondence or works on 3D meshes.

# 3 APPROACH

## 3.1 Overview

The main idea of our approach is to extend the algorithm of [FM98] for detecting error points (i.e. foldovers) in three-dimensional space. We extend the algorithm of Fujimura as it is a one-step technique. However, its extension to 3D is not trivial. We need to solve several problems, such as possible rotations, skew lines or non-planarity of quads.

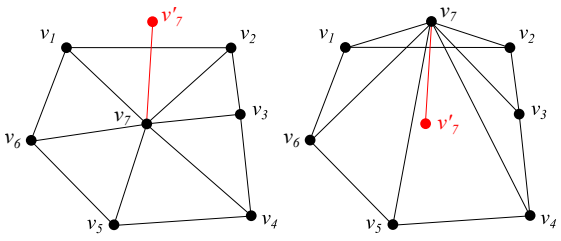
The core of our approach is the detection of error points leading to foldovers and their correction. This core approach is described in Section 3.2. As some error points detected by this algorithms can be false positives (e.g., due to rotations or non-planarity), we need to avoid them. For this purpose, we present several algorithms detecting these problems, thus extending our core approach (see Section 3.3). This also includes an extended error correction using weighting of errors.

## 3.2 Core Approach

In this section, we describe our core algorithms for detecting and correcting foldover error points.

### 3.2.1 *Detection of Error Points*

Fujimura and Makarov described the detection of foldover events in two-dimensional meshes in [FM98]. They state that the problem appears when a point leaves the neighboring polygon while being moved along its correspondence path (see Figure 3).




(a) Source mesh. (b) Target mesh.

Figure 3: We show a two-dimensional polygon with a translation of the point $v_7$. From **(a)** to **(b)** $v_7$ moves along its correspondence path outside of the polygon. Hence, the correspondence intersects the edge $v_1 - v_2$ of the polygon.

We use the same concept in 3D. For a foldover to be detected, the correspondence path no longer needs to intersect an edge of the points neighboring region. The correspondence path $cp(t)$ rather has to intersect one of the sides $S$ of the area between the source and the target area (see Figure 4). In this case, the area of the affected triangle becomes negative:

$$\exists s \in S : (cp(t) - \overrightarrow{v_s}) \cdot \overrightarrow{n_s} = 0 \quad , t \in [0,1].$$

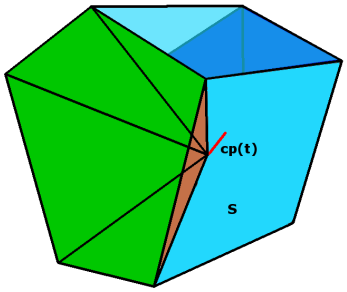
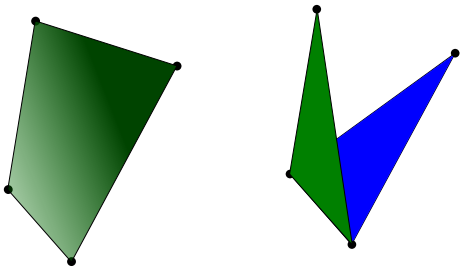



Figure 4: In the case of a foldover, the correspondence path $cp(t)$ intersects one of the sides $S$ (light blue) of the area between the source mesh (blue) and the target mesh (green).

Each side is put up by two source points and two target points. A surface that contains four points in three-dimensional space does not need to be planar, which makes it hard to calculate intersections. Therefore, the sides are approximated by two planes, each using the

```
for all v ∈ V do
    foldover ← false
    for all s ∈ S do
        if cp(t) intersects s1 or s2 then
            foldover ← true
    if foldover then
        errors.append(v)
```

Algorithm 1: Detect errors



(a) Non-planar quad.

(b) Approximation using two planes.

Figure 5: A quad that is build around four points does not need to be planar **(a)**. However with the help of two planes that are made out of three points each, this side can be approximated **(b)**.

two points from the source mesh $\mathcal{M}$ and one of the points from the target mesh $\widetilde{\mathcal{M}}$ (see Fig. 5):

$$s_1 : \overrightarrow{x} = (v_2 - v_1) + t\widetilde{v}_1$$
$$s_2 : \overrightarrow{x} = (v_2 - v_1) + t\widetilde{v}_2.$$

The core error detection algorithm is in Algorithm 1. The prerequisite for this technique is that the source mesh is 2-manifold. Otherwise the correspondence path may not intersect with any side even though a foldover exists. Furthermore the source mesh has to be foldover free, which should be ensured by a suitable parameterization. Holes in the mesh on the other hand do not interfere with the functionality, as the technique only depends on the direct neighbors of the current point and also works if the point is on the border of the mesh.

### 3.2.2 *Correction of Error Points*

With the detection Algorithm 1, it is possible to find vertices with correspondences that lead to foldovers. These problems are corrected by moving the target point to a valid area, so that the correspondence path does not leave the three-dimensional volume that is spanned by the neighboring region of the source and the target point and the area of the triangles that became negative turn positive again.

To satisfy this condition, various possible new positions can be used. Our algorithm moves the target point to

```
for all v ∈ errors do
    Find neighboring region r on the target mesh M̃
    correctedPoint ← r.ComputeCentroid()
    ṽ ← correctedPoint
```

Algorithm 2: Correct errors

the centroid $c$ of its neighboring region with the points $[v_0, \ldots, v_n]$:

$$c = \frac{\sum_{i=0}^{n} v_i}{n}$$

Using the centroid as the corrected point is not only easy to compute, it has the advantage that it has the optimal distance to the neighboring points. Thus the probability of a neighboring error affecting the current point is minimal. Algorithm 2 illustrates the sequences of steps for the correction.

## 3.3 Extension: Avoiding False Positives

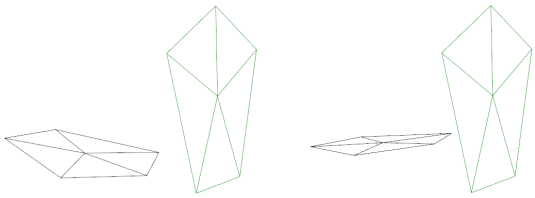
We extend our core error detection algorithm, as it may lead to false positives. For example rotation, mesh overlap or degenerated triangles can result in false positive error detections. In the following, these false positive errors are systematically filtered from the detected errors, leaving only relevant errors. Finally, the filtered errors are corrected using an extended correction algorithm, which orders the error correction according to the errors' severity.

### 3.3.1 *Detecting Rotations*

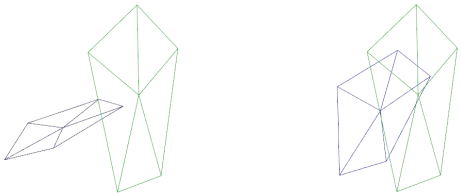
When comparing differently shaped meshes, it can happen that the corresponding regions on the source and the target mesh are rotated to one another by 90° or more. While morphing, the source region approaches the target region as shown in Figure 6.

Theoretically, the conditions for the detection of a foldover in rotated regions are the same as in the standard scenario described above. Nevertheless, the detection algorithm may mark a correct point as an error, because an intersection with one of the sides is identified. Figure 7a shows a scenario, where the detection algorithm incorrectly finds an error in a correct region, because the correspondence path crosses a side plane. In contrast, Figure 7c is an example for a correct error identification, despite of rotation. Hence both results are possible. To filter the false positive results, we first rotate the triangles before running the detection algorithm. Figures 7b and 7d show that the false positively detected error is resolved, whereas the correctly detected error remains. We perform the following three steps for this purpose:

*1. Approximate the surface normals of the source and target normals:*
To determine the rotation matrix we need to calculate

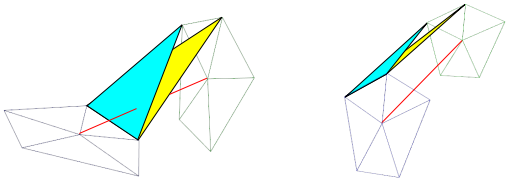


(a) Step 1: start (b) Step 2



(c) Step 3 (d) Step 4: finish

Figure 6: The example shows how a region (blue) can rotate from the source mesh to a target mesh. In such a case a foldover does not have to occur.



(a) Original (b) After rotation: no foldover

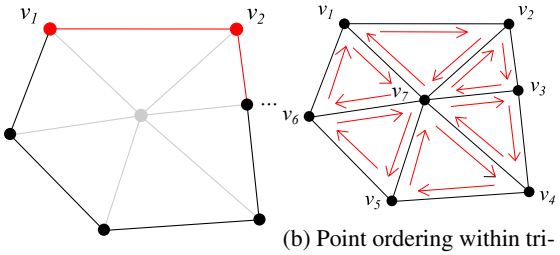
(c) Original (d) After rotation: foldover

Figure 7: Errors with and without rotation before the morphing. **(a)** and **(b)** represent a false positive error, whereas **(c)** and **(d)** show a real error.

the normals of both regions. With their help the rotation between two vectors can be computed. As the region that surrounds the current point is not a two-dimensional polygon, the surface consists of several triangles with different rotations in three-dimensional space. To determine one surface normal for the region, the surface normals of all triangles that are contained in the region are calculated and averaged. In our case, the surface normals were weighted by the area of the triangles. To ensure a correct addition of the single surface normals, they need to have the same orientation. This can be reached by ordering the points of the region in a constant manner. For this, we first discard all edges that contain the current point and then choose an arbitrary point. From this point, we iterate along the remaining edges to the next point and gain a sorted list of points (see Figure 8a). With the help of this list, we can now ensure a consistent orientation of all triangles. Each triangle starts at the center point and uses the previous and the next point in the sorted list (see Figure 8b). The surface normal $N$ can now be calculated as follows:

$$\forall \triangle_i(v_1,v_2,v_3) : N_i = (v_2 - v_1) \times (v_3 - v_1)$$
$$\Rightarrow N_{Region} = \frac{\sum_{i=0}^{n} N_i}{|\sum_{i=0}^{n} N_i|}.$$

*2. Calculate the rotation matrix between normals:*




(a) Overall point ordering (b) Point ordering within triangles.

Figure 8: To ensure that all surface normals have the same orientation, the points are ordered along the edges of the region **(a)**. Figure **(b)** shows the order of the points per triangle.

Now that we are able to calculate the surface normals of the source and the target region, it is possible to calculate the rotation matrix for a transformation from the source to the target normal. In three-dimensional space the rotation between two vectors can be described by three rotation angles $\alpha, \beta, \gamma$. A way to retrieve them is to use quaternions, which are generalized complex numbers:

$$q = s + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = (\mathbf{v}, s).$$

Using unit quaternions $|\hat{\mathbf{q}}| = (\sin(\theta)r,\ \cos(\theta))$ it is possible to calculate the rotation matrix as:

$$\mathrm{R}(\theta, r) = \hat{\mathbf{q}}\hat{\mathbf{p}}\hat{\mathbf{q}}^{-1},$$

where $\hat{\mathbf{p}}$ is the unit quaternion of the point $p$ that is to be rotated by $\theta$ around the axis $r$. According to Tomas et al. [AMHH08] this can be reduced to

$$R(n_1,n_2) = \begin{pmatrix} e+hv_x^2 & hv_xv_y - v_z & hv_xv_z + v_y & 0 \\ hv_xv_y + v_z & e+hv_y^2 & hv_yv_z - v_x & 0 \\ hv_xv_z - v_y & hv_yv_z + v_x & e+hv_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where $v = n_1 \times n_2$, $e = n_1 \cdot n_2$ and $h = \frac{1}{1+e}$.

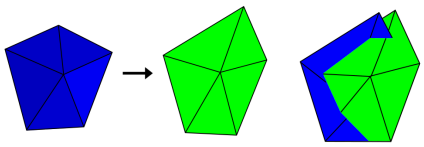
*3. Execute detection Algorithm with transformed source region:*

We now perform the detection algorithm 1 on the transformed region.

 ISBN 978-80-86943-66-4

### 3.3.2 *Detection of Overlaps, Sideward Movements and Degenerated Triangles*

Similar to the problems with rotated regions, mesh overlap or sideward movement as seen in Figure 9 can lead to false diagnosis. Because of the side movement the angles of the side planes to the mesh regions are extremely sharp. Moreover the mesh overlap results in correspondence paths, that point in different directions. We use a similar technique as with the rotations. During the handling of the rotation we already calculated the normal vectors. Thus it is possible to translate all points of the source region in the direction of the target normal. The result is that the movement is not sidewards anymore, which prevents overlap during the movement.



(a) Before (b) After

Figure 9: These images illustrate a scenario, where a source region (blue) moves sidewards towards the target region (green). As a result of the three-dimensional structure of the meshes, they can overlap.

When creating mesh-correspondences not every source point is associated with a distinct target point. To the contrary, correspondences can be surjective. Such a many-to-one correspondence can create degenerated triangles if two points have the same correspondence. Furthermore a triangle can degenerate, when one point moves onto the egde between the other two points. Our algorithm tests, whether a triangle degenerates and discards an error that resulted from these conditions. This is done by calculating the distance between the correspondence points and between the points and the edges, respectively.

### 3.3.3 *Correction using Error Weighting*

Up to now, only isolated errors of one point were considered. The detection algorithm uses the neighboring points to span the side planes. If one of these neighboring points is itself faulty, the side planes lead to miscalculations of the detection algorithm. Thus a correctly detected error can imply false positive detections at the neighboring points as Figure 10 shows.
In order to distinguish the error categories, the influence of a detected error on the involved points is calculated. We define the influence of an error as the proportion the error has on the Euclidean distance $d(x,y)$ between the original correspondence point and the corrected correspondence point. This is possible, because a corrected correspondence point can even be calculated for false positive errors. Regarding Figure 10 error **C** has direct influence on the Euclidean distance between the original correspondence point and the corrected correspondence point, because it is the actual error. Both **A** and **B** are neighboring errors that result from error **C**. However they do not have a direct influence on the error, because the neighboring points that span the region around the error only have an influence of $\frac{1}{N}$, where $N$ is the amount of neighboring points. Hence, it is possible to order the detected errors by their influence, also called error weight. Table 1 shows weights for Fig. 10.

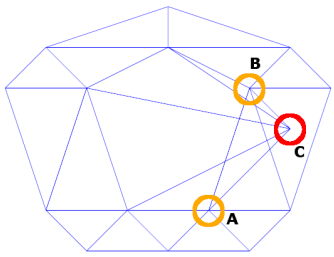



Figure 10: On the depicted target mesh, the point that results in a foldover is marked red. Due to this error correct neighboring points are mistakenly identified as errors. They are marked orange.

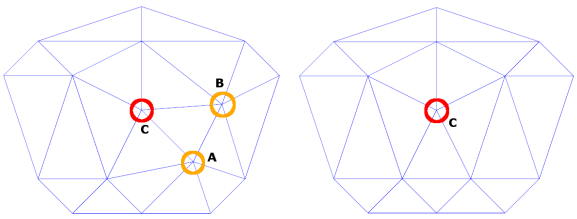
*errors.sort*(*DESCENDING*)
**for all** $v \in errors$ **do**
  **if** v.isStillError() **then**
    Find neighboring region $R$ on the target mesh $\widetilde{\mathcal{M}}$
    $correctPoint \leftarrow R.ComputeCentroid()$
    $\widetilde{v} \leftarrow correctPoint$
  **else**
    *errors.remove*($v$)

Algorithm 3: Correct errors in the order of their severity (i.e. error weight).

| Error | Error Weight |
|---|---|
| C | 22.50 |
| B | 2.01 |
| A | 1.25 |

Table 1: Errors from Figure 10 sorted in descending order by error weight. The actual error **C** is at the top of the list, before the resulting false positive errors.

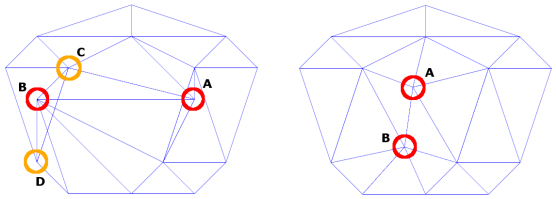
Then the correction algorithm corrects all errors. It starts with the first error in the list. The possibility for it to be before all its resulting false positive errors is very high. Hence, repeating the test for errors should yield no more false positive results. The correction algorithm can thus be extended to retest each error prior to the correction as described in Algorithm 3. The effect of the error ordering by error severity (error weights) can be seen in Figure 11.




(a) Without weighting. (b) With weighting.

Figure 11: Figure **(a)** shows the result of the correction without error weighting. All errors are corrected, regardless of whether they were actual errors or false positives. With weight-based ordering, **(b)** only the correspondence point that actually leads to a foldover is corrected.

The previous example showed false positive errors that result from neighboring real errors. However, the correction using error weighting also works with neighboring real errors. Figure 12 shows the correction of neighboring real errors. The resulting false positive errors are hereby filtered by the error weighting.




(a) Before the correction. (b) Weighted correction.

Figure 12: In **(a)** two neighboring real errors are marked red. The resulting false positive errors are marked yellow. **(b)** shows the example after the correction with error weighting. Only the real errors were corrected.

## 4 EVALUATION

With the help of several example meshes, we evaluate the analysis of mesh correspondences with regard to foldovers. We also discuss the advantages and demonstrate possible applications of the analysis as well as the limitations.

We present foldover analysis of two sets of data. First, correspondences of automatic and expert 3D medical image segmentations, and, second, correspondences between original and deformed 3D meshes. Note that we used simple not foldover-free correspondence estimation methods for the evaluation of our approach.

The first case analyzes foldovers in the correspondences resulting from the comparison of two meshes: an automatic 3D medical image segmentation and a segmentation performed by an expert. As medical image segmentation experts are interested in detecting segmentation errors (i.e., large distances between the automatic and expert segmentations), we use local distances for defining mesh correspondences. In particular, we employ and compare two distance measures: a commonly used Surface Distance [GJC01] and its recent improvement – the Extended Surface Distance [GKL15]. We analyzed the correspondences of segmentations of six livers and one carotid artery. The data stems from real images.

Second use case analyzes foldovers in the correspondences between original and deformed banchmark datasets: The Stanford Bunny (http://graphics.stanford.edu/data/3Dscanrep/), the Frog and the Buste (Both http://www.aimatshape.net/).

Table 2 shows the amount of vertices and triangles of the data sets.

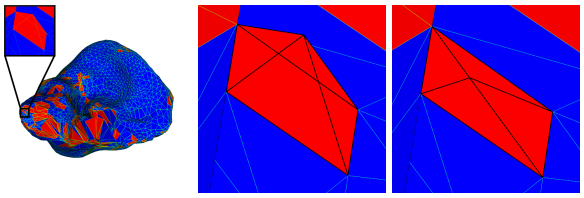
| Example | # Vertices | # Triangles |
|---|---|---|
| Liver 1 | 2562 | 5120 |
| Liver 2 | 6002 | 12000 |
| Liver 3 | 3996 | 8000 |
| Liver 4 | 4000 | 8000 |
| Liver 5 | 4002 | 8000 |
| Liver 6 | 3996 | 8000 |
| Carotid artery | 30674 | 61344 |
| Bunny | 2533 | 5062 |
| Frog | 5002 | 10000 |
| Buste | 5002 | 10000 |

Table 2: The table shows the amount of vertices and triangles of the organ segmentation examples.

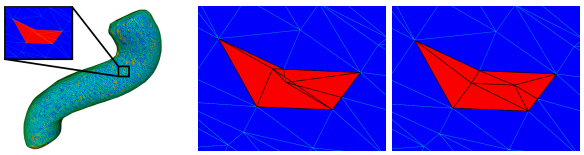
The presented foldover analysis and correction technique aims at detecting existing correspondences that result in foldovers and correcting them by moving the correspondence to a foldover free position.

To evaluate the technique, we first look at the identified foldover errors in the data visualizations. Figures 13a - 13c show the foldover result of a liver mesh comparison using SD. An isolated error is highlighted by marking the neighboring region red. Figure 13b clearly shows that, prior to correction, the correspondence point lies outside of the triangulation of the neighboring points, thus results in a foldover. After the correction, the erroneous correspondence point is moved inside the valid area (see Figure 13c).
Analogously, Figures 13d - 13f display the same scenario on the artery mesh using SD.

For deeper evaluation, Figure 14 displays two cases with large areas, where many errors exist. Both examples show the area before and after the correction is executed.
Figure 15 shows a comparison of a morphing sequence with foldover correction to one without foldover correction using ESD. In the uncorrected morphing it is visi-



(a) Liver SD. (b) Foldover. (c) Correction.



(d) Artery SD. (e) Foldover. (f) Correction.

Figure 13: After the analysis of the SD of a liver mesh **(a)** and an artery mesh **(d)**, the errors are marked red. Pictures **(b)** and **(e)** show obvious foldovers. In **c)** and **(f)** they are corrected.

ble, that during the sequence foldovers occur. The corrected morphing sequence stays foldover-free, without harming the global structure of the correspondences.
Alongside the sole correction of foldover problems, the analysis was also developed to be able to assess different sets of correspondences concerning their degree of foldover or characteristic foldover regions and compare them with others. In the following example (see Figure 16) a mesh of an automatic liver segmentation is shown. Marked with yellow is the region with the most detail of the mesh. It represents the porta hepatis (entry of artery and vein) of the liver. It is noticeable that the analysis detects several errors forming a ring around the area. This indicates, that the correspondences are prone to foldovers in high detail areas.

By now, we have used the foldover analysis to evaluate a given correspondence. Now, we look at different sets of correspondences and compare them with the help of the foldover analysis. Table 3 shows the results of the foldover comparison of SD and ESD on five different liver segmentation data. We report the amount of errors detected as well as their percentage in all triangles. This evaluation provides us with the insight, that the ESD produces more foldover problems than the SD. However the result is only valid for liver meshes, as different correspondences perform differently on varying mesh structures.

## 5 LIMITATIONS

Our approach corrects the found errors by moving the faulty correspondence points and thus resolving the foldover. This results in a slight change of the position of the corresponding point in the mesh. Because of the fact that the target mesh is constructed out of points on the second mesh of the comparison with the triangulation of the first mesh, the corrected correspondences do

| | **Surface Distance** | | **Extended S. Distance** | |
|---|---|---|---|---|
| Liver | # Errors | % | # Errors | % |
| 1 | 60 | 2,342% | 122 | 4,762% |
| 2 | 162 | 2,7% | 187 | 3,12% |
| 3 | 122 | 3,053% | 118 | 2,953% |
| 4 | 145 | 3,363% | 134 | 3,35% |
| 5 | 122 | 3,048% | 194 | 4,85% |
| **ø** | | **2,9012%** | | **3,807%** |

Table 3: Based on the five examples, the probability of a foldover in the category of liver segmentations of SD compared to ESD is evaluated.

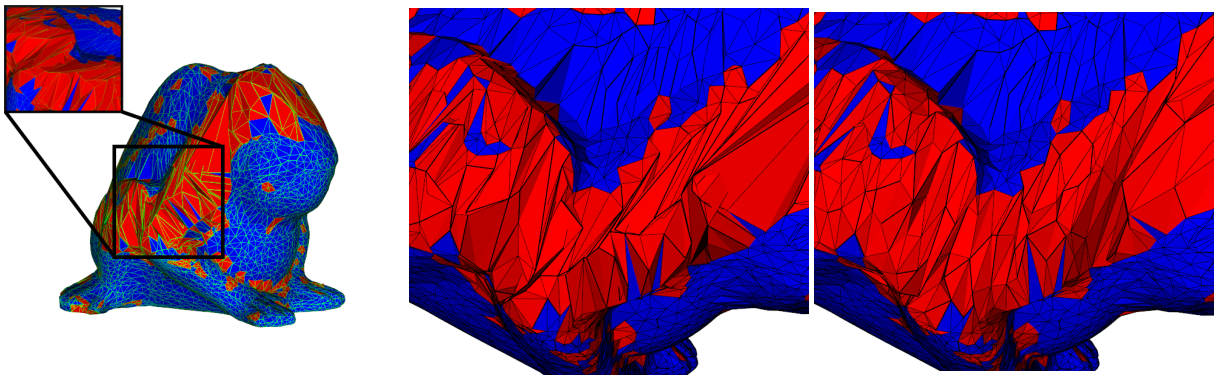
| | # Errors | |
|---|---|---|
| $\varepsilon$ | $\mathcal{Z}$ | $\mathcal{Z}_{\mathcal{K}}$ |
| $10^{-2}$ | 27 | 21 |
| $10^{-3}$ | 59 | 53 |
| $10^{-4}$ | 62 | 59 |

Table 4: Depending on the chosen $\varepsilon$-precision the amount of leftover false positive errors changes. By repeating the algorithm it can be minimized. $\mathcal{Z}$ is the result of the first and $\mathcal{Z}_{\mathcal{K}}$ the output of the second correction.

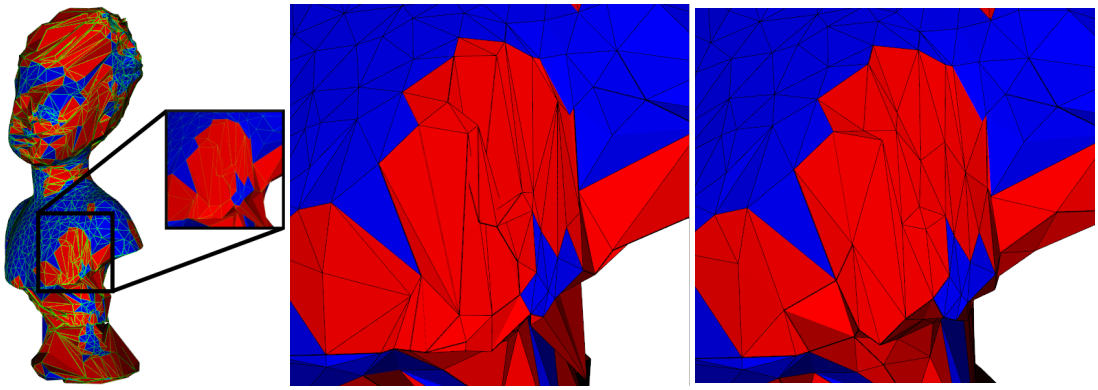
not necessarily lie on the second mesh. However, the correction is designed to draw inferences about the correspondence estimation method. It could thus be used as an indicator for the method. Moreover, the correspondence estimation can easily be repeated after the correction, to iteratively find foldover-free correspondences that completely lie on the second mesh.
Furthermore it must be noted that the correction algorithm only corrects foldover errors. If the underlying correspondences are not expressing a meaningful relation between the two meshes, the algorithm cannot improve that, but merely make it foldover-free.
The algorithm was also not designed to detect foldovers that were introduced through self collision of the mesh. Even if some parts of the mesh touch during morphing this is not a foldover, as the triangulation does not define the overlapping points as neighboring.
Due to inaccuracies in some parts of the algorithm a small amount of false positive errors can still remain. For example the approximation of the surface normal can be inaccurate, if one triangle of the region is very different from the rest. Additionally the computation of the intersections and the test for degenerated triangles works with an $\varepsilon$-precision. Depending on the chosen $\varepsilon$ the amount of remaining false positive errors can be adjusted. To reduce this amount the algorithm can be repeated by reanalyzing the corrected correspondences. Table 4 illustrates this issue.

## 6 CONCLUSION & FUTURE WORK

We proposed a new method to identify foldovers in 3D mesh correspondences. The method also offers a cor-

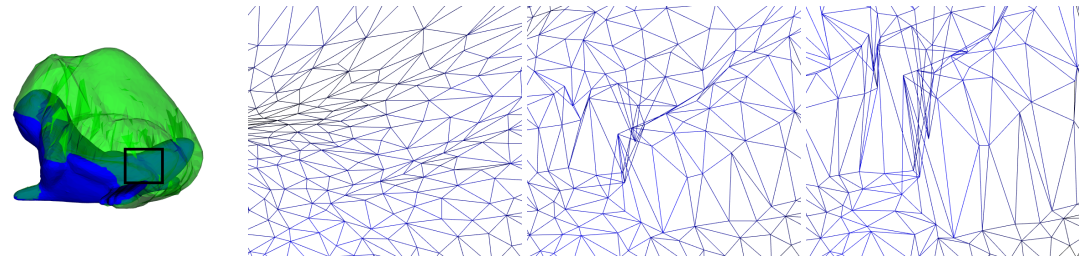


(a) Frog ESD. (b) Neighboring errors. (c) Correction.

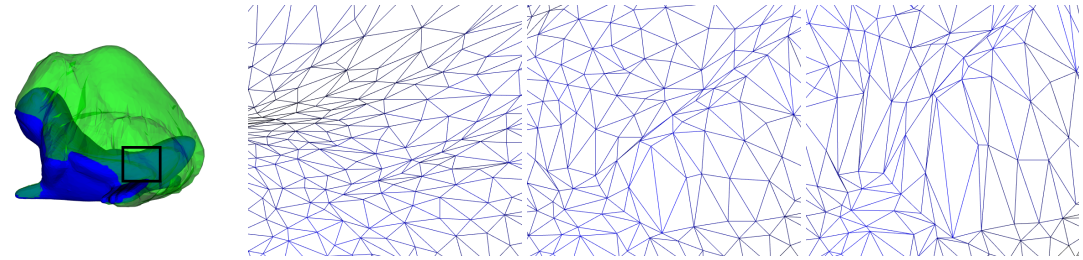


(d) Buste ESD. (e) Neighboring errors. (f) Correction.

Figure 14: ESD of the mesh of the frog **(a)** and the buste **(d)** are analyzed and corrected.



(a) Uncorrected ESD. (b) Morphing at 0%. (c) Morphing at 50%. (d) Morphing at 100%.



(e) Corrected ESD. (f) Morphing at 0%. (g) Morphing at 50%. (h) Morphing at 100%.

Figure 15: This figure compares a morphing sequence without foldover correction (**(a)** - **(d)**) to one with foldover correction (**(e)** - **(h)**) using ESD. The global structure of the correspondences is not harmed by the correction. For better visibility only the morphed mesh is shown in the steps, leaving out the source and target mesh.

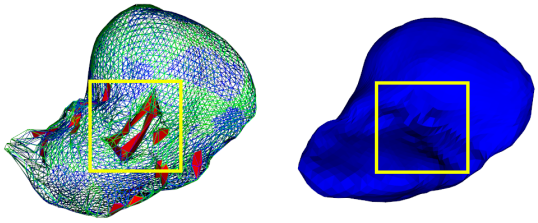


Figure 16: Around the area with the most detail of the mesh, an increased number of foldovers occur.

rection of the correspondences. It extends the work of Fujimura and Makarov [FM98] to 3D meshes and avoids false positive error detections.
The proposed algorithm is able to detect single and multiple foldover errors. Furthermore the correction provides a meaningful suggestion for the correspondences, provided the correspondences are of sufficient quality. With the obtained information it is possible to evaluate the given correspondences and compare them to others. The detection of false positive errors could be further improved with more reliable surface normals, as the surface normal calculation is not stable to outliers.

In the future, we will include the analysis into the development of a new distance measure, to ensure foldover-free correspondences during their generation. Moreover the amount of approximation regarding the side planes and the rotation will be reduced.

## 7 ACKNOWLEDGMENTS

The authors would like to thank Meike Becker, Matthias Kirschner and Georgios Sakas at TU Darmstadt as well as Stefan Wesarg at Fraunhofer IGD for providing us with the 3D medical image segmentation data and helping with the evaluation. The work was supported by DFG within a SPP 1335 project.

## 8 REFERENCES

[Ale00] Marc Alexa. “Merging polyhedral shapes with scattered features”. English. In: *The Visual Computer* 16.1 (2000), pp. 26–37.

[Ale02] Marc Alexa. “Recent advances in mesh morphing”. In: *Computer Graphics Forum*. Vol. 21. 2. Wiley Online Library. 2002, pp. 173–198.

[AMHH08] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering, Third Edition*. Taylor & Francis, 2008. ISBN: 9781439865293.

[CRS98] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. “Metro: Measuring error on simplified surfaces”. In: *Computer Graphics Forum*. Vol. 17. 2. Wiley Online Library. 1998, pp. 167–174.

[Ebk+13] Hans-Christian Ebke et al. “QEx: robust quad mesh extraction”. In: *ACM TOG* 32.6 (2013), p. 168.

[FM98] Kikuo Fujimura and Mihail Makarov. “Foldover-free image warping”. In: *Graphical models and image processing* 60.2 (1998), pp. 100–111.

[GJC01] Guido Gerig, Matthieu Jomier, and Miranda Chakos. “Valmet: A new validation tool for assessing and improving 3D object segmentation”. In: *MICCAI*. Springer. 2001, pp. 516–523.

[GKL15] Roman Getto, Arjan Kuijper, and Tatiana von Landesberger. “Extended surface distance for local evaluation of 3D medical image segmentations”. English. In: *The Visual Computer* (2015), pp. 1–11. ISSN: 0178-2789. DOI: 10.1007/s00371-015-1113-z.

[Gom99] J. Gomes. *Warping and Morphing of Graphical Objects*. Computer Graphics and Geometric Modeling Series Bd. 1. Morgan Kaufmann Publishers, 1999.

[LYY08] Tong-Yee Lee, Shao-Wei Yen, and I-Cheng Yeh. “Texture Mapping with Hard Constraints Using Warping Scheme”. In: *IEEE TVCG* 14.2 (2008), pp. 382–395.

[MTT13] Bogdan Mocanu, Ruxandra Tapu, and Ermina Tapu. “Mesh deformation with hard constraints”. In: *Signals, Circuits and Systems (ISSCS), 2013 International Symposium on*. IEEE. 2013, pp. 1–4.

[MZX14] Y. Ma, J. Zheng, and J. Xie. “Foldover-Free Mesh Warping for Constrained Texture Mapping”. In: *IEEE TVCG* 21.3 (2014), pp. 375–388.

[Tam+13] Gary KL Tam et al. “Registration of 3D point clouds and meshes: a survey from rigid to nonrigid”. In: *IEEE TVCG* 19.7 (2013), pp. 1199–1217.

[Tam+14] Gary KL Tam et al. “Diffusion pruning for rapidly and robustly selecting global correspondences using local isometry.” In: *ACM Trans. Graph.* 33.1 (2014), p. 4.

[VH01] Remco C Veltkamp and Michiel Hagedoorn. *State of the art in shape matching*. Springer, 2001.

[van+11] Oliver van Kaick et al. “A survey on shape correspondence”. In: *Computer Graphics Forum*. Vol. 30. 6. Wiley Online Library. 2011, pp. 1681–1707.