Higher-Order Ratio Cycles for Fast and Globally Optimal Shape Matching

Supplementary Material

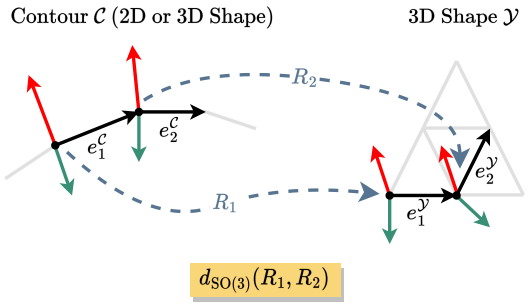

Figure A.1. Illustration of the **key idea of the higher-order regulariser** which favours local rigidity. We define two coordinate frames for two connected edges $e_1^{\mathcal{C}}, e_2^{\mathcal{C}}$ on the contour $\mathcal{C}$ and two connected edges $e_1^{\mathcal{Y}}, e_2^{\mathcal{Y}}$ on 3D shape $\mathcal{Y}$. To this end, we use the normal vectors (red arrows) at source vertices of respective edges and the cross-product (green arrows) between respective normal vector and respective edge. This allows us to compute two rotation matrices R_1 and R_2 between which we compute the geodesic distance $d_{\mathrm{SO}(3)}(R_1, R_2)$ which is essentially our higher-order cost.

A. Details on Higher-Order Costs

In the following, we summarise the main aspects of the higher-order (local rigidity) regulariser presented in [61]. The key idea is to define two coordinate frames on the contour $\mathcal{C}$ and the 3D shape $\mathcal{Y}$ and to compute the difference between the rotation matrices describing the rotation from first (second) coordinate frame on $\mathcal{C}$ to first (second) coordinate frame on $\mathcal{Y}$, see also Fig. A.1.

To compute the higher-order cost, we consider two connected edges $e_1^{\mathcal{C}}, e_2^{\mathcal{C}}$ on the contour $\mathcal{C}$ and two connected edges $e_1^{\mathcal{Y}}, e_2^{\mathcal{Y}}$ on the 3D shape $\mathcal{Y}$. We note that the edge pairs $e_1^{\mathcal{C}}$ and $e_1^{\mathcal{Y}}$ as well as $e_2^{\mathcal{C}}$ and $e_2^{\mathcal{Y}}$ resemble vertices in the conjugate product $\bar{\mathcal{P}}$ and that these conjugate product vertices are connected by an edge for which we compute the higher-order cost [61]. Furthermore, we also consider the normals at respective source vertices of the edges on both shapes. Then, we can define the local coordinate frames by using an edge, its respective normal and their cross-product, see Fig. A.1. From that, we compute two rotation matrices R_1, R_2, namely from the first (second) coordinate frame on the contour $\mathcal{C}$ to the first (second) coordinate frame on the 3D shape $\mathcal{Y}$. Our resulting higher-order cost, i.e. the local rigidity regulariser, can then be computed by computing the geodesic distance (see Fig. A.2 for an illustration) between R_1 and R_2 on the Lie group $\mathrm{SO}(3)$ to quantify the difference in rotation of consecutive matched edges, and thus to quantify the local rigidity of a matching.

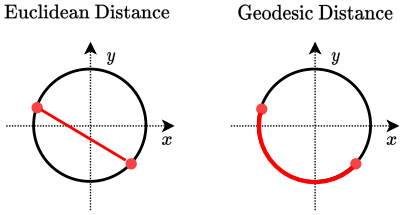

Figure A.2. Illustration of the difference between the **euclidean distance** (straight red line) and the **geodesic distance** (red arc) in 1D for two points on the unit circle.

B. Details on Distortion Bound

The distortion bound value k means that a single vertex of the curve $\mathcal{C}$ (2D or 3D shape) can be matched to at most k connected vertices within $\mathcal{Y}$ (3D shape), see Fig. A.3 for an illustration.

C. Additional Insights 2D-3D Shape Matching

Data Preparation. We follow [61] for data preparation: we decimate all 3D shapes from FAUST$^{\mathrm{2D}}_{\mathrm{3D}}$ and TOSCA$^{\mathrm{2D}}_{\mathrm{3D}}$ datasets to half of their resolution and resample 2D shapes such that resulting edge lengths are equal to median edge lengths of 3D shapes.

Additional Results. In Fig. A.5, we show more quantitative results on FAUST$^{\mathrm{2D}}_{\mathrm{3D}}$ and TOSCA$^{\mathrm{2D}}_{\mathrm{3D}}$ dataset including segmentation error on FAUST$^{\mathrm{2D}}_{\mathrm{3D}}$ as well as matching error on FAUST$^{\mathrm{2D}}_{\mathrm{3D}}$ with flips (for convenience, quantitive results from main paper are also reported). Furthermore, in Fig. A.7, we show more qualitative results on 2D-3D shape matching which showcase that our approach does not have the bias towards shorter cycles. In Tab. 3, we ablate on the distortion bound k w.r.t. geodesic errors and computation times. Finally, in Fig. A.4, we show results of our method in presence of noise.

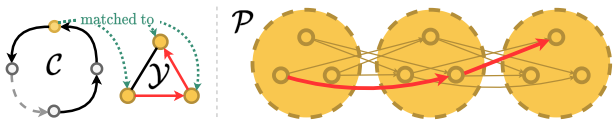

Figure A.3. **Intuition of the distortion bound** k. We show on the left that the yellow vertex of contour $\mathcal{C}$ is matched to 3 vertices of 3D shape $\mathcal{Y}$. In the product graph $\mathcal{P}$ (shown on the right), this matching is encoded with the illustrated red path going to the k duplicates of the yellow component. This illustrates the relationship between duplicates of components, i.e. the distortion bound, and possible matchings between $\mathcal{C}$ and $\mathcal{Y}$.

Distortion bound k	**2**	3	4	5	6	7	8	9	10
Mean Geo.Err w/o Flips ·100	**2.5**	2.7	2.6	2.9	2.9	2.9	2.9	2.8	2.8
Mean Runtime in s	**69.0**	69.2	**69.0**	69.3	70.2	70.8	71.6	71.5	71.7

Table 3. Ablation study on the **distortion bound** k on 10 pairs of $\text{FAUST}^{\text{2D}}_{\text{3D}}$. We vary the distortion bound k and compare mean geodesic errors (without left-right flips) as well as mean computation times (best is bold). Runtimes are smaller for smaller values of k due to increasing graph size with increasing k. Furthermore, $k = 2$ yields best geodesic errors.

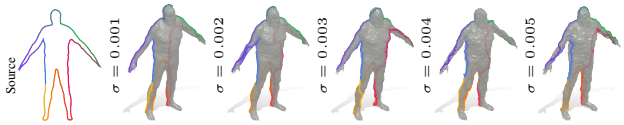

Figure A.4. We show results of our method in the **presence of noise** (we disturb 3D vertex coordinates of a shape from $\text{FAUST}^{\text{2D}}_{\text{3D}}$ with varying Gaussian noise $\mathcal{N}(0, \sigma^2)$). We can see (best viewed zoomed-in) local artefacts, but overall matchings are consistent across different levels of noise.

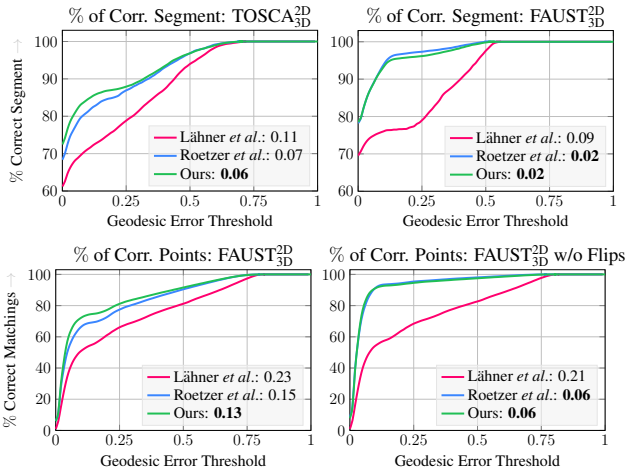

Figure A.5. More **quantiative** results on 2D-3D shape matching (numbers in legends are mean geodesic errors). **Top:** we compare segmentation errors on $\text{TOSCA}^{\text{2D}}_{\text{3D}}$ and $\text{FAUST}^{\text{2D}}_{\text{3D}}$. **Bottom:** we compare matching errors with respect to ground-truth correspondences (we remove left-right flipped matchings on the right).

Additional Runtime Experiments. In Fig. A.6, we compare runtimes of all 2D-3D shape matching approaches with additional GPU re-implementation of minimum cost cycle problems (i.e. Lähner *et al*. and Roetzer *et al*. with distortion bound). For GPU re-implementations, we use the graph modification that we present in Sec. 3 (i.e. using distortion bound k to avoid intra-component cycles). We set $k = 2$. Furthermore, we use dynamic programming on GPU (i.e. the Moore-Bellman-Ford algorithm [5, 25, 52]) to compute a (potentially acyclic) minimum cost path from the first component to itself (going through all other components). To finally obtain a *cyclic* minimum cost path, we use the branch and bound strategy presented in [41, 61].

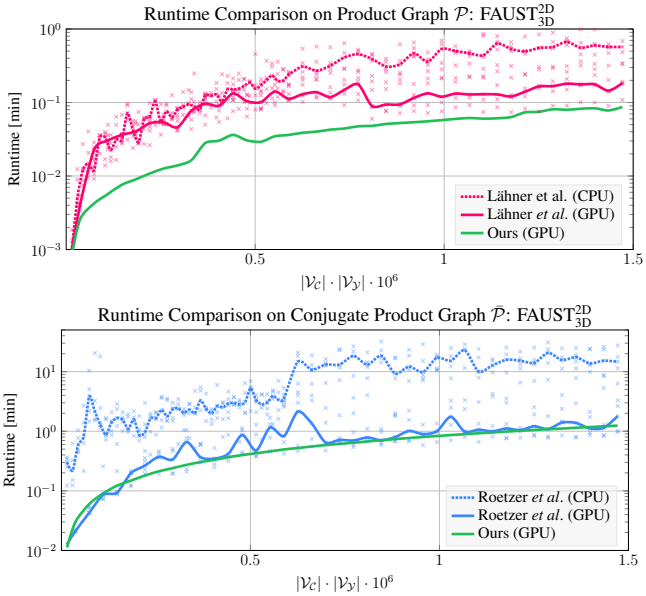

Figure A.6. Comparison of **computation times** (including graph and cost computation) for fixed resolution of 3D shapes ($|V_{\mathcal{Y}}| = 3500$), while varying the resolution of the 2D contour. Lines are median computation times. **Top:** we compare computation times of approaches running on $\mathcal{P}$: Lähner *et al*., our GPU re-implementation of Lähner *et al*. and our approach. **Bottom:** we compare computation times of approaches running on the approx. 10× larger *conjugate* product graph $\bar{\mathcal{P}}$: Roetzer *et al*., our GPU re-implementation of Roetzer *et al*. and our approach. Both comparisons confirm that improved computation times of our approach not only stem from implementation on GPU but also from requiring less amount of branches.

This implementation is similar to algorithms presented in [41, 61] except that we integrate the distortion bound (cf. Fig. 3) which allows for parallelisation (algorithms [41, 61] use heaps which prohibits parallel implementation). We release the GPU implementation for minimum cost cycles along with our implementation for minimum ratio cycles.

Finally, we compare to Howard's algorithm [34] to conduct runtime experiments[1]. To this end, we use $\mathcal{P}$ and set $k = 0$ to keep the problem as small as possible. Even though we run on a much smaller problem (i.e. $\mathcal{P}$ instead of $\bar{\mathcal{P}}$ and $k = 0$ instead of $k = 2$), Howard's algorithm did not finish within a day on the smallest instance reported in Fig. A.6 (very likely caused by its exponential worst-case time complexity).

[1] Implementation taken from https://lemon.cs.elte.hu/.

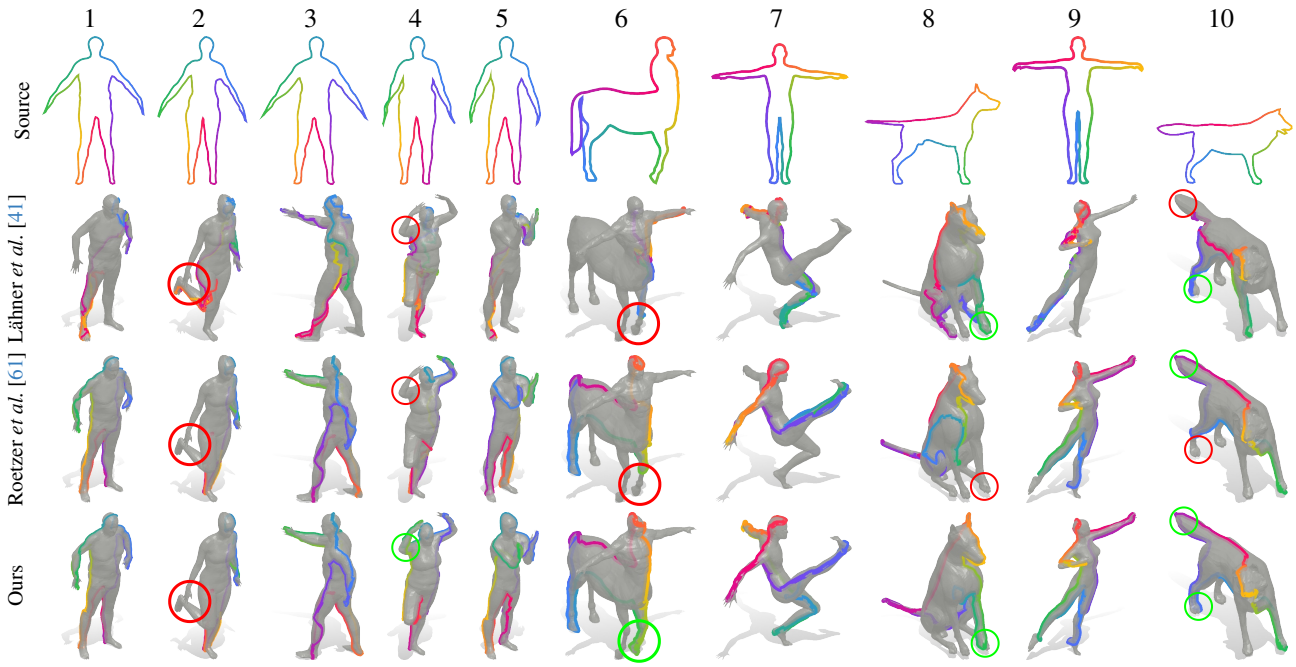

Figure A.7. More **qualitative** results on 2D-3D shape matching. The first five columns are shapes from FAUST$^{\text{2D}}_{\text{3D}}$ dataset while the last five columns are shapes from TOSCA$^{\text{2D}}_{\text{3D}}$ dataset. In column two, we can see a failure case of our method where the matching collapses to one side of the 3D shape (very likely stemming from intrinsic symmetries). Apart from that, Roetzer *et al*. and ours consistently produce better results than Lähner *et al*. while ours has less bias towards shorter paths (see green and red circles).

D. Additional Insights 3D-3D Shape Matching

Data Preparation. We follow [59] and decimate all 3D shapes to 1000 triangles. This conveniently allows to compare results directly with the ones reported in [59]. Furthermore, we repair any mesh defects using [3]. Nevertheless, we note that in principle shapes with boundary or genus $g > 0$ can be matched with our approach.

Additional Results. In Fig. A.8, we show additional qualitative results for 3D shape matching for all methods (including DiscrOpt [57] and SmoothShells [22]).

Figure A.8. More **qualitative** results for 3D shape matching for all methods. The first six columns are the shape pairs shown in the main paper ($1-3$ from FAUST dataset and $4-6$ from DT4D) including results computed using methods DiscrOpt and SmoothShells. The last six columns are additional qualitative results ($7-9$ from FAUST dataset and $10-12$ from DT4D dataset).