Partial-to-Partial Shape Matching with Geometric Consistency

Supplementary Material

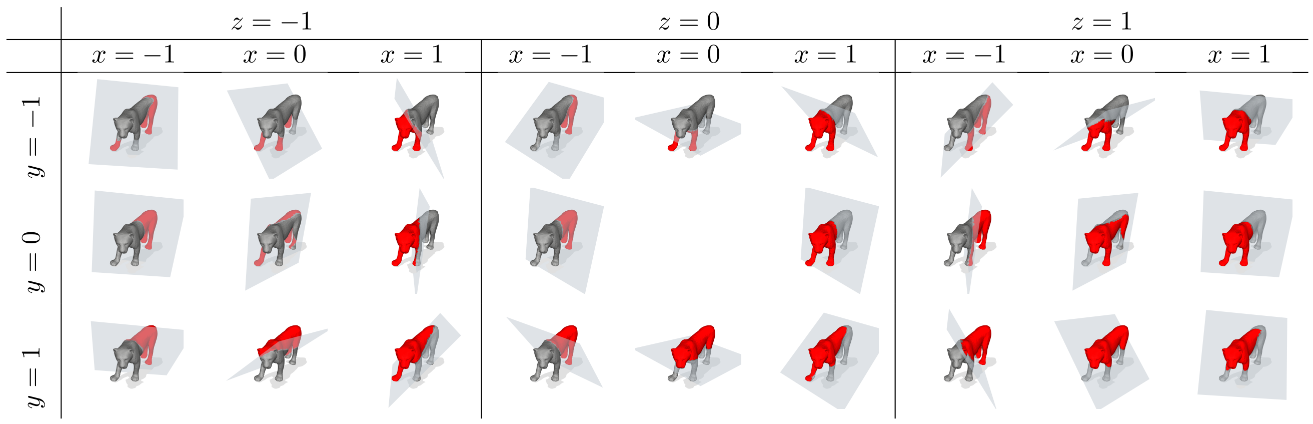

Figure 9. We generate our **PARTIALSMAL dataset** from shapes of the SMAL dataset [47] by rotating a plane with normal vector (x, y, z) around the origin and cutting off one half of the shape (as shown in gray). We show the resulting 26 generated partial shapes (red) of an exemplary cougar shape. The shown values x, y, z in the table correspond to the normal vector (x, y, z).

7. Data Preprocessing

Orientation and Manifoldness. Our algorithm requires all shapes in our datasets to be edge-manifold, vertex-manifold, and oriented. We consider a shape to be oriented if all triangle normals point outwards which we ensure for every shape. We achieve vertex manifoldness by duplicating each non-manifold vertex, and use the *libigl*[2] library to achieve edge-manifoldness. Furthermore, we remove all disconnected shape parts.

Shape Decimation. The complexity of our Integer Linear Program increases quadratically with the number of triangles of respective shapes $\mathcal{X}$ and $\mathcal{Y}$. For manageable runtimes and memory consumption when solving our Integer Program (Eq. (6)), we reduce the amount of triangles of shapes $\mathcal{X}$ and $\mathcal{Y}$. To achieve reduction in the number of triangles, we employ a straightforward edge collapse method, where two vertices are merged into one during each step of the reduction process. We apply these reduction steps until a predefined number of vertices is attained. We reduce a partial shape $\mathcal{X}$ proportionally to its surface area $A(\cdot)$ to $100 \cdot A(\mathcal{X})$ number of triangles.

Feature Computation. We compute features $\mathbf{W}^{(\mathcal{X})}$ on the full-resolution shape $\mathcal{X}$ and transfer them to the low-resolution counterpart. This involves associating each low-resolution vertex with a high-resolution vertex using the edge-collapse algorithm. Additionally, we compute Voronoi areas for each vertex on the low-resolution shape. We then average the features of all high-resolution vertices that fall within the Voronoi area around the corresponding mapped high-resolution vertex.

[2] https://libigl.github.io

8. PARTIALSMAL Dataset Generation

The first step of generating the PARTIALSMAL dataset is mean-centering all shapes of the SMAL test dataset [47]. We then define planes with origin at $(0, 0, 0)$ and normals (x, y, z), with $(x, y, z) \in \{-1, 0, 1\}^3 \backslash (0, 0, 0)$. The shape is then cut along this plane, i.e., all triangles above the plane (all triangles lying on the side of the plane with positive normal direction) are discarded. We consider the largest connected part of the remaining triangles as the resulting partial shape. We show examples of generated shapes in Figure 9.

9. Ablation Studies

We show analysis in terms of runtime, performance and memory consumption. Additionally, we reason the choice of our objective function and compare our method to the comparison method in terms of overlapping regions.

9.1. Reduction of Optimization Time

For a sample shape, we display the optimization time over x (i.e. number of binary elements equal to one) in Figure 10.

9.2. Scalability

Our method's scalability is constrained by quadratically growing memory consumption and exponentially increas-

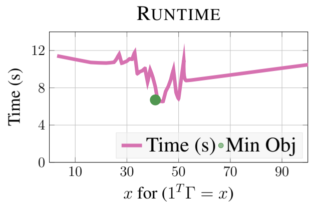

Figure 10. **Optimization Time** for linear subproblems of the minimum mean algorithm of one example shape w.r.t. number of binary variables, that equal 1. We observe that the minimal mean solution often needs the least optimization time.

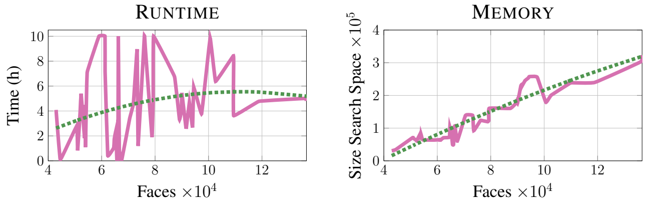

Figure 11. The **runtime and memory consumption** on high resolution shapes highly relate on the problem instance and the feature quality. The green line visualizes the trend.

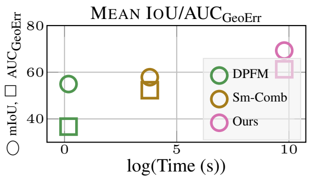

Figure 12. Our method exceeds the comparison methods in terms of mean IoU (circle) and AUC$_{\text{GeoErr}}$ (square), but is slower in terms of optimization time.

ing runtime. However, our coarse-to-fine scheme extends our method's applicability to high-resolution shapes by propagating the matching to the full resolution (in the main paper, we use the full resolution given in the datasets). In Figure 11 we show runtime and memory consumption (w.r.t. search space size) of high resolution shapes. Note that the runtime and memory consumption heavily rely on the problem instance. The green line shows the trend of runtime and memory.

9.3. Runtime vs. Performance

We plot mean IoU (↑) and AUC$_{\text{GeoErr}}$(↑) over log of optimization time (see Figure 12). Despite slower average opt. time (Sm-Comb: 49s; DPFM train: 118.2min, test: 1.31s; Ours 299.05min) our method yields improved performance. We hope that our work will inspire follow-up works that explore improved solvers for such problems. While our work is not competitive to current SOTA methods w.r.t. runtime, we consider our contribution as a vital first step to solving the extremely challenging and yet underexplored partial-to-partial shape matching setting.

Sum [0]	Sum [0.01]	Sum [0.1]	Sum [1]	**Ours**
51.23	52.53	53.12	52.53	**69.29**

Table 3. We compare **mean IoU** of our normalized objective function with the sum objective and weighting factor λ (in brackets) on the CP2P TEST dataset. Our normalized version performs best.

9.3.1 Objective Function

We normalize our objective function by the number of elements in $\mathbf{\Gamma}$: $\frac{\langle \mathbf{C},\mathbf{\Gamma}\rangle}{\mathbf{1}^{\mathbf{T}}\mathbf{\Gamma}}$.

We show in our ablation study in Table 3 that this normalization improves performance over using only the sum as the objective function $\langle \mathbf{C}, \mathbf{\Gamma}\rangle$ or adding a penalty function to the sum objective $\langle \mathbf{C}, \mathbf{\Gamma}\rangle - \lambda \mathbf{1}^{\mathbf{T}}\mathbf{\Gamma}$ in terms of mean IoU.

9.3.2 Performance in terms of Percentage of overlapping region

With an increasing percentage of the overlapping region, we see increased performance in our method. For a substantial amount of overlapping region, we observe similar results between the method from Sm-comb [37], DPFM [3], and our method, which makes sense as it is close to a full-to-full matching. For more partiality, our approach outperforms the comparison methods. We show the results in Figure 13.

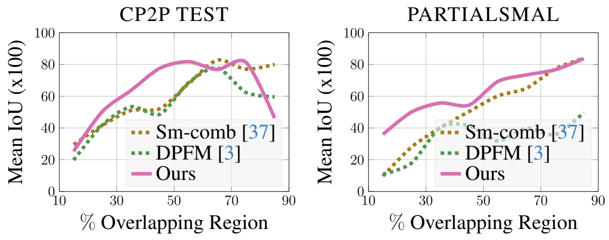

Figure 13. Our methods' performance increases with an **increasing percentage of overlapping region**. With larger overlapping regions, it performs similarly to Sm-comb [37] and DPFM [3]. With smaller overlapping regions, our method outperforms the others.

10. Failure Cases & Artefacts

We can not guarantee geometric consistency at boundary edges because of the lack of adjacent neighboring triangles. The interior of two triangle patches is still matched geometrically consistent, see Sec. 5 in the main paper. We show some failure cases in Figure 14.

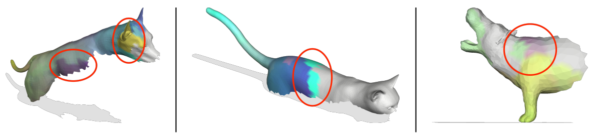

Figure 14. We observe some **failure cases** based on the missing geometric consistency on boundary edges.

11. Dataset Analysis

During our research we noticed that qualitatively similar shapes in the SHREC16 CUTS test dataset do also appear in the SHREC16 train dataset, although the vertex positions have some slight jitter. In Figure 15 we visualize two plots that show the cumulative number of test shapes (vertical axis) versus the distance between each test shape and the nearest training shape. While in the CUTS dataset we oberserve 47 shapes with this problem the HOLES dataset is not affected.

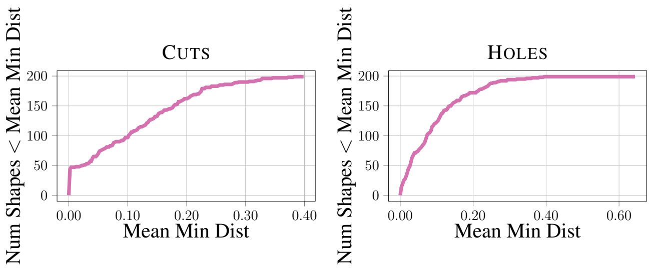

Figure 15. The Mean Minimum distance of the SHREC16 dataset is almost zero for 47 shape in the SHREC16 CUTS dataset. The SHREC16 HOLES dataset is not affected by this problem.

12. Search Space Pruning

Below we elaborate on the bounds that we use for our search space pruning. Let $j, k \in \mathbb{N}$. We know that $g(j+1)-g(j) \leq 1$. We start from this and first derive that $h(j+1)-h(j) \leq \frac{1}{j}$, where $h(j) = \frac{g(j)}{j}$.

Starting with

$$g(j+1) - g(j) \leq 1, \tag{7}$$

we divide by j, so that we get

$$\frac{g(j+1)}{j} - \frac{g(j)}{j} \leq \frac{1}{j}. \tag{8}$$

Plugging in $h(j) = \frac{g(j)}{j}$ gives

$$\frac{g(j+1)}{j} - h(j) \leq \frac{1}{j}. \tag{9}$$

As $g(j+1) \geq 0$ and $j+1 \geq 0$, $\frac{g(j+1)}{j} \geq \frac{g(j+1)}{j+1}$, so that we get

$$\frac{g(j+1)}{j+1} - h(j) \leq \frac{1}{j}. \tag{10}$$

By definition $h(j+1) = \frac{g(j+1)}{j+1}$, so that we obtain

$$h(j+1) - h(j) \leq \frac{1}{j}. \tag{11}$$

Let us now consider $h(k) - h(j)$, where $k \geq j$. We can use inequality (11) to bound this expression from above: We know that $h(j+1)-h(j) \leq \frac{1}{j}$, $h(j+2)-h(j+1) \leq \frac{1}{j+1}$, .., $h(k) - h(k-1) \leq \frac{1}{k-1}$. By adding up this sequence of inequalities we obtain $h(k) - h(j) \leq \sum_{i=j}^{k-1} \frac{1}{i}$.