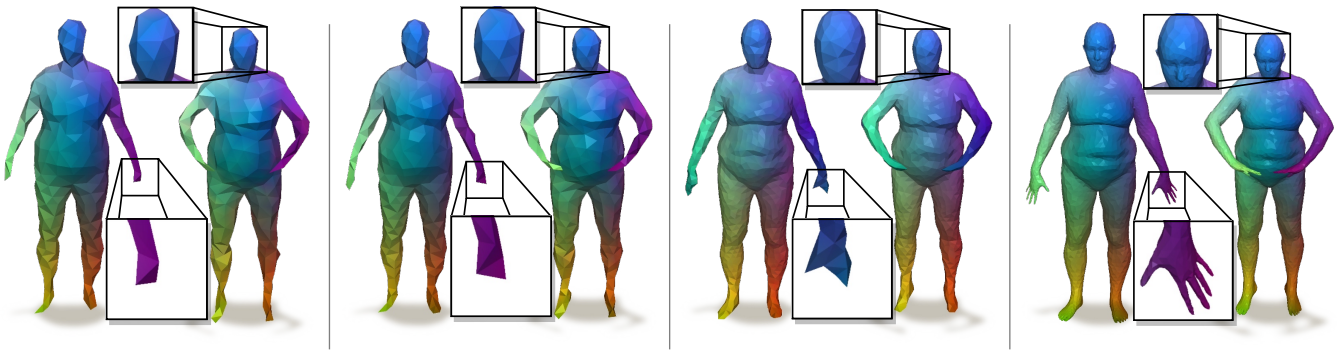
DOI: 10.1111/cgf.70208
Eurographics Symposium on Geometry Processing 2025
M. Attene and S. Sellán
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 5

High-Resolution 3D Shape Matching with Global Optimality and Geometric Consistency

N. El Amrani† and P. Roetzer† and F. Bernard

University of Bonn & Lamarr Institute, Germany



DiscoMatch [RAC*24]: max 750* △ | SM-Comb [RSCB22]: max 1,000 △ | GeCo [RB25]: max 3,000 △ | Ours: max **10,000** △

Figure 1: *Illustration of the maximal resolution (in terms of number of triangles △) of **geometrically consistent 3D shape matching methods** (i.e. methods that preserve local neighbourhood relations of surface elements) within a time budget of 1.5 hours (method marked with * runs on GPU and reached memory limits at shown resolution). Our approach scales almost an order of magnitude better than the previous best method [RB25] and with that bridges the gap to practically relevant resolutions. From the zoomed-in parts, we can see that high-resolution meshes are necessary to faithfully capture geometry of 3D shapes.*

Abstract

3D shape matching plays a fundamental role in applications such as texture transfer and 3D animation. A key requirement for many scenarios is that matchings exhibit geometric consistency, which ensures that matchings preserve neighbourhood relations across shapes. Despite the importance of geometric consistency, few existing methods explicitly address it, and those that do are either local optimisation methods requiring accurate initialisation, or are severely limited in terms of shape resolution, handling shapes with only up to 3,000 triangles. In this work, we present a scalable approach for geometrically consistent 3D shape matching that, for the first time, scales to high-resolution meshes with up to 10,000 triangles. Our method follows a two-stage procedure: (i) we compute a globally optimal and geometrically consistent mapping of surface patches on the source shape to the target shape via a novel integer linear programming formulation. (ii) we find geometrically consistent matchings of corresponding surface patches which respect correspondences of boundaries of patches obtained from stage (i). With this, we obtain dense, smooth, and guaranteed geometrically consistent correspondences between high-resolution shapes. Empirical evaluations demonstrate that our method is scalable and produces high-quality, geometrically consistent correspondences across a wide range of challenging shapes. Our code is publicly available: https://github.com/NafieAmrani/SuPa-Match.

† Authors contributed equally.

1. Introduction

The problem of 3D shape matching — finding correspondences between two non-rigidly deformed 3D surfaces — is central to a wide range of visual computing applications, including 3D reconstruction, shape interpolation, statistical shape modelling, and texture transfer. A crucial property for meaningful and smooth correspondences is geometric consistency, i.e. the preservation of local neighbourhood relations between matched surface elements (see also Definition 5). In discrete settings, such as triangle meshes, this notion parallels diffeomorphisms in the continuous domain [WSSC11a] and ensures that neighbouring regions on one shape are mapped to neighbouring regions on the other. This property is essential for robust regularisation of an otherwise (potentially) ill-posed problem, to achieve smoothness, and to obtain plausible correspondences for diverse downstream tasks. However, most existing shape matching methods ignore geometric consistency. This in particular includes machine learning approaches for which enforcing respective constraints is extremely challenging [DSO20; RSO19; CRB23; BXNL24]. Methods that do incorporate geometric consistency, either require accurate initialisation [SAPH04; SHCB11; SCBK20; Tak22; SPK23; EB17; VLR*17; EHA*19], consider a weaker notion of geometric consistency [RB24; REC*25], are limited to low-resolution shapes (see Table 1), or build on local optimisation within a coarse-to-fine-scheme [WSSC11b; RAC*24]. Especially, coarse-to-fine schemes without optimality guarantees can lead to arbitrarily bad matchings since “wrong" matchings on a coarse stage often cannot be improved on finer stages. In addition, low-resolution shapes cannot capture all geometric details (cf. also Fig. 1) which makes low-resolution matchings unsuitable for downstream tasks as e.g. texture transfer (which requires dense matching on high-resolution shapes). Consequently, there is a need for scalable, globally optimal and geometrically consistent 3D shape matching approach that can be applied to practically relevant, high-resolution shapes.

In this work, we address this gap by proposing a novel two-stage algorithm for geometrically consistent matching between two non-rigidly deformed 3D shapes that is capable of handling high-resolution shapes with up to 10,000 triangles. In particular, each stage of our approach computes matchings directly on the highest resolution, so that it does not suffer from potential discretisation artefacts, while at the same time achieving global optimality for each stage. With that, our two-stage approach remains scalable w.r.t. to runtime and memory requirements (compared to single-stage approaches) while (empirically) producing accurate results. In the first stage, we find a *globally optimal* and *geometrically consistent* matching between patches (defined on the high-resolution source shape) and the high-resolution target shape. We emphasise that we only need to define patches on the source shape, which are then *consistently* transferred to the target shape using our framework. In the second stage, we find a geometrically consistent matching of the corresponding patches on both shapes, so that overall we obtain a dense surface map between both shapes. We summarise our contributions as follows:

- We propose a geometrically consistent 3D shape matching algorithm that, for the first time, scales to practically relevant resolutions (up to 10,000 triangles).
- To this end, we propose a formalism to find a globally optimal matching of a set of patches (defined on the source shape) to the target shapes, allowing us to consistently transfer patches from source to target shape.
- Our empirical evaluation shows that our approach is substantially better scalable compared to existing geometrically consistent approaches.

Method	Globally Optimal	Max. Resolution* (Number of △)
Windheuser et al.[WSSC11a]	✗	≈ 250
SM-Comb [RSCB22]	✗	≈ 1,000
DiscoMatch [RAC*24]	✗	≈ 750 (GPU OOM)
SpiderMatch [RB24]	✓	≈ 3,500 (weak geo. cons.)
GeCo [RB25]	✓	≈ 3,000
Ours	✓	≈ 10,000

Table 1: *Comparison of **geometrically consistent** shape matching methods (*with 1.5 hours time budget, cf. also Fig. 6). We emphasise that SpiderMatch [RB24] only poses a weaker form of geometric consistency compared to the other methods.*

2. Related Work

In this section, we summarise the most relevant shape matching approaches to our work. For a detailed review of the shape matching literature, we refer to survey papers [VZHC11; TCL*12; DYDZ22].

2.1. Shape Matching

A wide range of axiomatic shape matching approaches have been proposed in the literature, including methods based on heat kernels [OMMG10], diffusion distances [BBK*10], game-theory [RBA*12], consensus maximisation [PPCG19], and convex relaxations [CK15; MDK*16]. Other approaches tackle the shape matching problem by finding correspondences between sets of keypoints using mixed integer programming formalisms [BST20; GRE*23]. A particularly influential line of work is the functional maps framework [OBS*12; MDK*16; GCR*17; MRR*19; ELC20; RMWO21], which formulates the matching problem in the spectral domain and allows for efficient computation of dense point-to-point maps. Building on this framework, many deep learning-based methods have been developed, either trained in a supervised [LRR*17; WEH20; LLHL20; GFK*18; TCM*21] or unsupervised setting [HLR*19; SACO22; DCO22; LDO22; CRB23; AO23; DMM24; BXNL24], leading to remarkable performance in terms of matching accuracy. However, commonly most shape matching approaches do not explicitly enforce geometric consistency. As a result, even highly accurate matchings may violate local neighbourhood preservation and may thus not be reliable in downstream applications, such as texture transfer or statistical shape analysis. In contrast, our approach incorporates geometric consistency directly into the optimisation using explicit constraints within an integer linear programming formalism, ensuring that neighbourhoods are preserved enforcing that resulting matchings are smooth.

2.2. Geometrically Consistent Shape Matching

Geometric consistency is crucial for producing smooth correspondences between shapes, as it ensures the preservation of local neighbourhood relations. Despite its importance, many shape matching methods overlook geometric consistency due to the inherent complexity of the resulting optimisation problems. A classical example is the quadratic assignment problem (QAP), which is known to be NP-hard [RPW94], and thus typically addressed using continuous relaxations [DML17; BK17; KMDL19] or heuristic optimisation techniques [SPKS16; HLC20; BLG*21; HHF*21]. Another family of approaches achieves geometric consistency by using continuos shape parametrisations and local optimisation techniques. Yet, these methods strongly depend on reliable initialisation — either from a sparse set of landmark correspondences [SAPH04; SHCB11; SCBK20; Tak22; SPK23], or from dense correspondences [EB17; VLR*17; EHA*19]. In contrast, several shape matching problems in the 2D-2D or 2D-3D settings can be solved efficiently to global optimality while enforcing geometric consistency using shortest path-based approaches. For instance, dynamic time warping [SK83] allows for matching open contours while closed contours can be matched through graph cuts in product graphs [STC09]. Shortest-paths can also be used for model-based image segmentation [CYES00; Fel05; SC09], while similar formalisms have also been successfully applied to matching 2D contours to 3D shapes [LRS*16; RLB23; PRLB25; REC*25]. Geometrically consistent matching of 3D shapes in the discrete domain has been tackled through integer linear programming (ILP) formalism, where correspondences are defined between triangles [WSSC11a; WSSC11b]. Building on this formalism, heuristic solvers both on CPU [RSCB22] and on GPU [RAC*24] have been proposed. Other works extend the formalism [WSSC11a] to partial shape matching for the inputs with open boundaries [ERE*24] and unknown overlap [EGR*24]. An alternative was introduced in [RB24], where the surface of the source shape is represented using a self-intersecting, closed curve embedded in 3D space. Here, geometric consistency is enforced by maintaining intersection points of the curve when matching the curve. Building on this idea, a new formalism was recently proposed in [RB25], which represents each individual triangle of a shape as a cyclic curve and constructs a hyper product graph by coupling these cycles along opposite edges. This enables global geometric consistency across the entire surface. We build on this idea and propose a two-stage matching pipeline. In the first stage, we compute geometrically consistent matchings of patches on the source shape while ensuring that their corresponding patches (which are induced by the matching) remain manifold on the target shape. In the second stage, we use [RB25] while incorporating additional boundary constraints from matchings computed with our first stage.

2.3. Scalable Shape Matching

The number of variables of shape matching problems grows quadratically with the number of surface elements, since each point on the source shape can in principle be matched to every point on the target shape. This quadratic growth poses a major challenge for scalability, particularly in high-resolution settings. To mitigate this issue, several strategies have been proposed to make shape matching more scalable. One class of approaches involves hierarchical matching frameworks, which incrementally refine coarse matchings across multiple resolutions. Such methods have been developed based on heat kernel signatures [VBG10] and for the functional maps framework [SVB19; NH22]. Other works aim to improve the scalability of functional map-based pipelines by subsampling the input shape, i.e. selecting a sparse set of points on high-resolution meshes to reduce computational cost [MO23]. Similarly, the widely used ZoomOut algorithm [MRR*19] has been adapted for GPU architectures, with memory-efficient modifications introduced to further enhance its scalability [MO24]. Additional advancements leverage intrinsic triangulations to perform down- and upsampling within the functional maps pipeline [MBRM24]. While these methods have made significant progress in handling large meshes, geometrically consistent shape matching remains particularly challenging in this context. This is due to additional geometric consistency constraints often leading to large integer linear programs which are generally only solvable in exponential time [WSSC11a; RB24; RB25]. As a result, most existing geometrically consistent methods are limited to low-resolution shapes with fewer than $3,000$ triangles. To scale beyond this, some methods employ coarse-to-fine strategies that first solve the problem using low-resolution shapes and then propagate the solution to higher resolution shapes [WSSC11b; RAC*24]. However, such approaches often suffer from discretisation artefacts, which is particularly prominent at low resolutions. In contrast, our method supports matching at much higher resolutions (up to $10,000$ triangles) without requiring any coarse-to-fine strategies. Both stages of our proposed approach operate directly on high-resolution shapes, which allows us to avoid severe discretisation artefacts.

3. Background

In this section, we discuss graph-based shape representations relevant for our method, define geometric consistency and explain the concept of cyclic product graphs [LRS*16].

3.1. Shape Representation

In our work, we aim to find a geometrically consistent matching between two non-rigidly deformed, manifold surface shapes embedded in 3D space. To this end, we consider the directed graphs induced by the triangle meshes of the shapes.

Definition 1 (Directed Graph) A *directed graph* $\mathcal{G}$ is defined as a tuple $(\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ of vertices $\mathcal{V}_\mathcal{G}$ and oriented edges $\mathcal{E}_\mathcal{G} \subset \mathcal{V}_\mathcal{G} \times \mathcal{V}_\mathcal{G}$ (i.e. oriented edge $(v, \bar{v}) \in \mathcal{E}_\mathcal{G}$ does not imply that the opposite edge $-(v, \bar{v}) := (\bar{v}, v) \in \mathcal{E}_\mathcal{G}$).

Definition 2 (3D Shape) A *3D shape* $\mathcal{X}$ is a directed graph $(\mathcal{V}_\mathcal{X}, \mathcal{E}_\mathcal{X})$ with vertices $\mathcal{V}_\mathcal{X}$ and oriented edges $\mathcal{E}_\mathcal{X} \subset \mathcal{V}_\mathcal{X} \times \mathcal{V}_\mathcal{X}$, such that $\mathcal{X}$ induces an oriented triangle mesh and consequently $\mathcal{X}$ forms an orientable continuous 2D manifold (possibly with boundary $\mathcal{B}_\mathcal{X} := \{e \mid e \in \mathcal{E}_\mathcal{X}; -e \notin \mathcal{E}_\mathcal{X}\}$) embedded in 3D space.

Furthermore, we consider patches defined on the surface of the respective shapes, i.e. subgraphs of the directed graph of a shape. For an illustration of patches, see also Fig. 2.

Figure 2: *Illustrations of **patches** which tile the surface of respective 3D shapes. All patches together cover the whole surface of the shape and each individual patch has a single boundary. See Section 5.1 on how we extract patches.*

Definition 3 (Surface Patch) A *surface patch* $\mathcal{S}$ of a shape $\mathcal{X}$ is a directed graph $\mathcal{S} = (\mathcal{V}_\mathcal{S}, \mathcal{E}_\mathcal{S})$ with vertices $\mathcal{V}_\mathcal{S} \subset \mathcal{V}_\mathcal{X}$ and oriented edges $\mathcal{E}_\mathcal{S} \subset \mathcal{E}_\mathcal{X}$ which (i) induces a manifold 2D surface embedded in 3D space and (ii) has a single boundary $\mathcal{B}_\mathcal{S} := \{e \mid e \in \mathcal{E}_\mathcal{S}; -e \notin \mathcal{E}_\mathcal{S}\}$.

For our matching formalism, we are especially interested in matching cyclic chain graphs representing the boundaries of surface elements of the source shape (e.g. a triangle or a patch) to the target shape.

Definition 4 (Cyclic Chain Graph) A *cyclic chain graph* $\mathcal{C} = (\mathcal{V}_\mathcal{C}, \mathcal{E}_\mathcal{C})$ is a special case of a directed graph that comprises of exactly one cycle. With that, every vertex $v \in \mathcal{V}_\mathcal{C}$ has exactly one incoming edge and one outgoing edge.

3.2. Geometric Consistency

We are interested in finding geometrically consistent matchings which we define as follows:

Definition 5 (Geometrically Consistent Matching) A *geometrically consistent matching* is a matching that preserves neighbourhoods such that whenever two surface elements are neighbouring on the source shape, their corresponding surface elements should also be neighbouring on the target shape.

For example, when two patches are neighbouring on the source shape, they should be matched to neighbouring patches on the target shape. Similarly, whenever two triangles are neighbouring on the source shape, they should be matched to neighbouring triangles (or edges or vertices) on the target shape.

3.3. Cyclic Product Graphs for Shape Matching

In [LRS*16], the authors introduce the *product graph* $\mathcal{P}$ between a cyclic chain graph $\mathcal{C}$ (i.e. a 2D shape) and a 3D shape $\mathcal{Y}$ to tackle the 2D-3D shape matching problem.

Definition 6 (Product Graph) *The product graph* $\mathcal{P}$ of a cyclic chain graph $\mathcal{C}$ and the directed graph of a 3D shape $\mathcal{Y}$ is a tuple $(\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P})$ of (product) vertices $\mathcal{V}_\mathcal{P}$ and (product) edges $\mathcal{E}_\mathcal{P}$ and reads

$$\begin{aligned} \mathcal{V}_\mathcal{P} &:= \mathcal{V}_\mathcal{C} \times \mathcal{V}_\mathcal{Y}, \\ \mathcal{E}_\mathcal{P} &:= \left\{ e = \left(\begin{pmatrix} v_\mathcal{C} \\ v_\mathcal{Y} \end{pmatrix}, \begin{pmatrix} \bar{v}_\mathcal{C} \\ \bar{v}_\mathcal{Y} \end{pmatrix} \right) \mid (v_\mathcal{C}, \bar{v}_\mathcal{C}) \in \mathcal{E}_\mathcal{C};\ (v_\mathcal{Y}, \bar{v}_\mathcal{Y}) \in \mathcal{E}_\mathcal{Y}^+ \right\} \end{aligned} \tag{1}$$

where $\mathcal{E}_\mathcal{Y}^+$ is the (with self-edges) extended set of edges which reads $\mathcal{E}_\mathcal{Y}^+ = \mathcal{E}_\mathcal{Y} \cup \{(v,v) \mid v \in \mathcal{V}_\mathcal{Y}\}$.

Each edge of $e \in \mathcal{E}_\mathcal{P}$ encodes a potential matching between an edge on the contour and an edge (or vertex) on the target shape $\mathcal{Y}$. We note that edge to vertex matchings are necessary to account for different discretisation of the shapes, see also Section 5. Usually, we define a cost for each edge which quantifies "how good" the potential matching is. In the 2D-3D shape matching case, globally optimal matchings based on $\mathcal{P}$ can be found by variants of Dijkstra's algorithm [LRS*16; RLB23]. Yet, in our work, we will consider an integer linear programming formalism to incorporate additional constraints, see also [RB24; RB25]. For more details, visualisations, and similar formalisms (e.g. for 3D shape matching) we refer to [LRS*16; RLB23; RB24; REC*25; RB25].

4. High-Resolution Shape Matching

On a high-level, we propose a two-stage approach to find geometrically consistent matchings between a source 3D shape $\mathcal{X}$ and a non-rigidly deformed target 3D shape $\mathcal{Y}$, see also Fig. 3 for an overview: (i) First, we represent the surface of the source shape with a set of patches (Section 4.1) and find geometrically consistent correspondences between the patches (i.e. correspondences that preserve the neighbourhood of *patches*) and the target shape (Section 4.2). (ii) Second, we match the surface of corresponding patches so that we obtain dense geometrically consistent correspondences (i.e. correspondences that preserve the neighbourhood of *triangles*) between source and target shape (Section 4.3).

We note that both of our stages build on formalisms introduced in [RB24; RB25]. In particular, in our first stage, we extend the integer linear program introduced in [RB25] (i.e. we use the proposed constraints (INJ), (FLOW), and (NEIGH)). Further, we include the additional constraints (MFOLD), which are inspired by [RB24], to avoid self-intersections of patch boundaries and with that non-manifoldness of matched boundaries. In our second stage, we extend the integer linear program introduced in [RB25] and integrate additional boundary constraints (BNDRY) and an orientation preservation regularisation term. In the following, we will elaborate on our overall framework. To ensure self-containedness, we will include a recap of the concepts introduced [RB24; RB25].

4.1. Surface Representation with Patches

We represent the surface of the source shape $\mathcal{X}$ by tiling it into n patches $\mathcal{S}_1, \dots, \mathcal{S}_n$, see Fig. 2 for a visualisation. For a faithful representation of the surface of $\mathcal{X}$, we require that each patch has a single boundary (cf. also Definition 3), and that all patches together cover the whole surface of $\mathcal{X}$. For more details on how we extract patches we refer to Section 5.1.

4.2. Geometrically Consistent Patch Matching

Building on [RB25], we propose an integer linear programming (ILP) formalism for geometrically consistent matching of patches. To this end, we represent the boundary $\mathcal{B}_{\mathcal{S}_i}$ of the i-th patch $\mathcal{S}_i$ with a cyclic chain graph $\mathcal{C}_i = (\mathcal{V}_{\mathcal{C}_i}, \mathcal{E}_{\mathcal{C}_i})$ with $\mathcal{V}_{\mathcal{C}_i} \subset \mathcal{V}_\mathcal{X}$ and $\mathcal{E}_{\mathcal{C}_i} = \mathcal{B}_{\mathcal{S}_i}$ (i.e. the cyclic chain graph induced by all boundary vertices

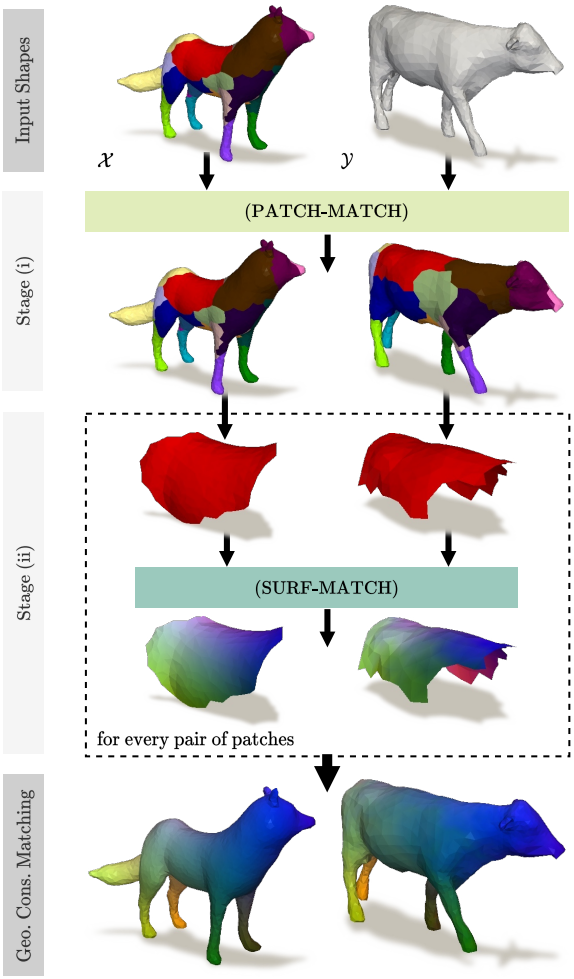


Figure 3: ***High-level overview*** *of our two stage approach. We find a geometrically consistent matching between two non-rigidly deformed 3D shapes $\mathcal{X}$ and $\mathcal{Y}$ by: (i) finding a geometrically consistent matching of patches (which we only define on the source shape and transfer them consistently to the target shape by the matching), and (ii) finding dense geometrically consistent matchings for each pair of corresponding patches.*

and all boundary edges of patch $\mathcal{S}_i$). For each cyclic chain graph $\mathcal{C}_i$ we construct an individual product graph $\mathcal{P}_i$ for matching $\mathcal{C}_i$ to the target shape $\mathcal{Y}$. Furthermore, we collect all n product graphs in what we call the *product graph soup* $\tilde{\mathcal{P}}$:

Definition 7 (Product Graph Soup) The *product graph soup* $\tilde{\mathcal{P}} = \left(\mathcal{V}_{\tilde{\mathcal{P}}}, \mathcal{E}_{\tilde{\mathcal{P}}}\right)$ of n cyclic chain graphs $\mathcal{C}_1, \dots, \mathcal{C}_n$ (e.g. the boundaries of n many patches $\mathcal{S}_1, \dots, \mathcal{S}_n$) and a 3D shape $\mathcal{Y}$ consists of n many product graphs $\mathcal{P}_1, \dots, \mathcal{P}_n$ such that

$$\begin{aligned} \mathcal{V}_{\tilde{\mathcal{P}}} &= \mathcal{V}_{\mathcal{P}_1} \cup \ldots \cup \mathcal{V}_{\mathcal{P}_n}, \\ \mathcal{E}_{\tilde{\mathcal{P}}} &= \mathcal{E}_{\mathcal{P}_1} \cup \ldots \cup \mathcal{E}_{\mathcal{P}_n}. \end{aligned} \tag{2}$$

We recall that each edge in the product graph soup $\tilde{\mathcal{P}}$ represents a potential matching between a boundary edge of a patch on $\mathcal{X}$ and an edge (or vertex) of $\mathcal{Y}$. Consequently, all potential matchings between boundary edges of the patches are contained within the product graph soup $\tilde{\mathcal{P}}$. Yet, we only want to find geometrically consistent matchings so that we need to incorporate additional constraints. To facilitate this, we consider an ILP formalism. To this end, we encode every edge $e \in \mathcal{E}_{\tilde{\mathcal{P}}}$ with a binary variable $x_k \in \{0,1\}$, where $x_k = 1$ means that the k-th product edge e is part of the final solution.

We start by constraining the solution space by requiring that every (directed) edge of each $\mathcal{C}_i$ is matched exactly once. With the binary encoding of edges, this can be ensured by imposing injectivity constraints, see Fig. 4 for an illustration.

$$\forall\, (v_{\mathcal{C}}, \bar{v}_{\mathcal{C}}) \in \mathcal{E}_{\mathcal{C}_1} \cup \cdots \cup \mathcal{E}_{\mathcal{C}_n} : \sum_{k : e_k = \left(\binom{v_{\mathcal{C}}}{\bullet}, \binom{\bar{v}_{\mathcal{C}}}{\bullet}\right) \in \mathcal{E}_{\tilde{\mathcal{P}}}} x_k = 1. \tag{INJ}$$

In addition, we want to enforce that matched boundaries form a continuous path on the target shape, see Fig. 4 for an illustration. In terms of the product graph soup, this means that whenever a product vertex $v \in \mathcal{V}_{\tilde{\mathcal{P}}}$ has an active incoming edge, it also needs to have an active outgoing edge. This can be ensured by employing flow-preservation constraints.

$$\forall\, v \in \mathcal{V}_{\tilde{\mathcal{P}}} : \sum_{k : e_k = (\bullet, v) \in \mathcal{E}_{\tilde{\mathcal{P}}}} x_k = \sum_{j : e_j = (v, \bullet) \in \mathcal{E}_{\tilde{\mathcal{P}}}} x_j. \tag{FLOW}$$

Furthermore, we want to ensure that neighbouring patches on $\mathcal{X}$ are also neighbouring patches on $\mathcal{Y}$. Since $\mathcal{X}$ and $\mathcal{Y}$ are oriented manifolds, neighbouring patches on $\mathcal{X}$ share opposite edges and consequently neighbouring patches on $\mathcal{Y}$ should also share opposite edges, see also Fig. 4. Thus, we enforce neighbourhood of patches by coupling opposite product edges.

$$\forall\, kj;\ e_k e_j \in \mathcal{E}_{\tilde{\mathcal{P}}};\ e_k = -e_j : \quad x_k = x_j. \tag{NEIGH}$$

Finally, we want to ensure that matched patches on $\mathcal{Y}$ yield manifold surfaces with a single boundary that induces a cyclic chain graph. For this, we ensure that the boundaries of each patch $\mathcal{S}_i$ do not intersect itself. Our manifold preservation constraints read

$$\forall\, i;\ v_{\mathcal{Y}}, \bar{v}_{\mathcal{Y}} \in \mathcal{V}_{\mathcal{Y}};\ v_{\mathcal{Y}} \neq \bar{v}_{\mathcal{Y}} : \sum_{k : e_k = \left(\binom{\bullet}{v_{\mathcal{Y}}}, \binom{\bullet}{\bar{v}_{\mathcal{Y}}}\right) \in \mathcal{E}_{\mathcal{P}_i}} x_k \leq 1 \tag{MFOLD}$$

We note that (MFOLD) allows to match a whole patch on $\mathcal{X}$ to a single vertex on $\mathcal{Y}$, which could be considered a non-manifold matching. Yet, we argue that this is a necessary feature to account for severe non-isometries between both shapes (which requires severe shrinking or stretching of patches).

To obtain our resulting ILP, we collect all binary variables (which encode product edges) in a vector $x \in \{0,1\}^{|\mathcal{E}_{\tilde{\mathcal{P}}}|}$. Additionally, we define a matching cost $c(e)$ for each product edge $e \in \mathcal{E}_{\tilde{\mathcal{P}}}$ which quantifies the cost of the potential matching. In particular, we use the difference of vertex-wise features (e.g. feature vector $f_{v_{\mathcal{C}}}$ of vertex $v_{\mathcal{C}} \in \mathcal{V}_{\mathcal{X}}$ and feature vector $f_{v_{\mathcal{Y}}}$ of vertex $v_{\mathcal{C}} \in \mathcal{V}_{\mathcal{Y}}$, see Section 5 for more details), such that the matching cost reads

$$c(e) = c\left(\binom{v_{\mathcal{C}}}{v_{\mathcal{Y}}}, \binom{\bar{v}_{\mathcal{C}}}{\bar{v}_{\mathcal{Y}}}\right) = \Psi(f_{v_{\mathcal{C}}} - f_{v_{\mathcal{Y}}}) + \Psi(f_{\bar{v}_{\mathcal{C}}} - f_{\bar{v}_{\mathcal{Y}}}). \tag{3}$$

Here $\Psi(\cdot)$ is the adaptive loss function proposed in [Bar19].

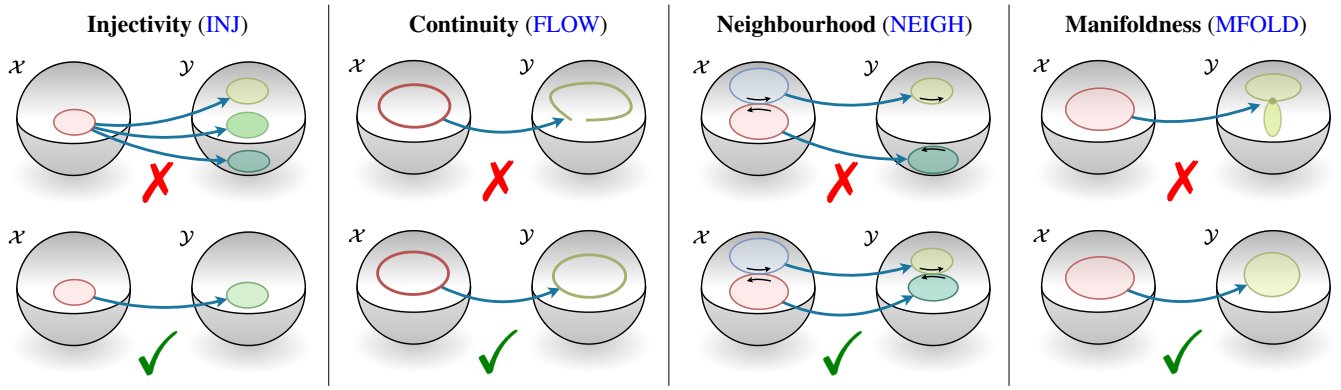


Figure 4: ***Illustration of the constraints*** *used in our integer linear program (PATCH-MATCH) for matching patches. With injectivity constraints, we ensure that each patch is matched exactly once. The continuity constraints ensure that each patch has a continuous boundary when matched to the target shape. We ensure geometric consistency of patches by employing geometric consistency constraints, i.e. neighbouring patches which share opposite edges on the source shape should remain neighbouring patches on target shape by sharing opposite edges. With manifoldness constraints, we ensure that the matched patches do not contain non-manifold vertices nor non-manifold edges.*

With that, our integer linear program for geometrically consistent matching of patches on $\mathcal{X}$ to the surface of the target shape $\mathcal{Y}$ reads

$$\min_{x\in\{0,1\}^{|\mathcal{E}_{\tilde{\mathcal{P}}}|}} c^T x \quad \text{s.t.} \quad \text{(INJ)}, \text{(FLOW)}, \text{(NEIGH)}, \text{(MFOLD)}. \tag{PATCH-MATCH}$$

A solution x^* of (PATCH-MATCH) yields a set of edges of shape $\mathcal{X}$ and corresponding edges of $\mathcal{Y}$, i.e. pairs of edges of both shapes collected in set $\mathcal{M} := \{\left(\binom{v_\mathcal{X}}{v_\mathcal{Y}}, \binom{\bar{v}_\mathcal{X}}{\bar{v}_\mathcal{Y}}\right) \mid (v_\mathcal{X}, \bar{v}_\mathcal{X}) \in \mathcal{E}_\mathcal{X}, (v_\mathcal{Y}, \bar{v}_\mathcal{Y}) \in \mathcal{E}_\mathcal{Y}^+\}$. Furthermore, a solution x^* of (PATCH-MATCH) induces for every patch on $\mathcal{X}$ a corresponding patch on $\mathcal{Y}$. In the second stage of our two-stage approach, we aim to find dense, geometrically consistent matchings between surfaces of corresponding patches.

4.3. Patch Surface Matching

Using a solution x^* of (PATCH-MATCH) (which in turn yields a set of corresponding edges $\mathcal{M}$), we now aim to find dense, geometrically consistent correspondences of the surfaces of corresponding patches. To this end, we use the formalism proposed in [RB25], while incorporating additional constraints which ensure that the resulting matching respects the matched boundaries of the patches, i.e. respects the set of corresponding edges $\mathcal{M}$.

In particular, for every patch $\mathcal{S}$ (on shape $\mathcal{X}$), we follow [RB25] and represent each of the m oriented triangles belonging to this patch $\mathcal{S}$ with a cyclic chain graph $\mathcal{C}_1^\triangle, \dots, \mathcal{C}_m^\triangle$. Analogous to above, we can define a product graph $\mathcal{P}_i^\triangle$ obtained from cycle $\mathcal{C}_i^\triangle$ and the corresponding patch on shape $\mathcal{Y}$. Eventually, we collect all product graphs in a product graph soup $\tilde{\mathcal{P}}^\triangle$, cf. Definition 7.

We represent each product edge $e_k \in \mathcal{E}_{\tilde{\mathcal{P}}^\triangle}$ with a binary variable $z_k \in \{0,1\}$. We ensure that the matching of the surfaces of the patches respects the matching of the boundary of the patches (i.e. $\mathcal{M}$ obtained by solving (PATCH-MATCH)) by employing boundary constraints, which reads:

$$\forall j; e_j \in \mathcal{M};\ e_k \in \mathcal{E}_{\tilde{\mathcal{P}}^\triangle};\ e_j = e_k: \quad z_k = 1 \tag{BNDRY}$$

Again, we collect all binary variables in a vector $z \in \{0,1\}^{|\mathcal{E}_{\tilde{\mathcal{P}}^\triangle}|}$. Furthermore, we incorporate injectivity, continuity and neighbourhood constraints, so that our integer linear program for finding geometrically consistent matchings between the surfaces of matched patches reads:

$$\min_{z\in\{0,1\}^{|\mathcal{E}_{\tilde{\mathcal{P}}^\triangle}|}} (d+\lambda r)^T z \quad \text{s.t.} \quad \text{(INJ)}, \text{(FLOW)}, \text{(NEIGH)}, \text{(BNDRY)}. \tag{SURF-MATCH}$$

Here, $d \in \mathbb{R}_+^{|\mathcal{E}_{\tilde{\mathcal{P}}^\triangle}|}$ is the per-edge matching cost (e.g. per-vertex feature difference, cf. Eq. (3) and Section 5), $r \in \mathbb{R}_+^{|\mathcal{E}_{\tilde{\mathcal{P}}^\triangle}|}$ our orientation preserving regulariser and $\lambda > 0$ a weight for the latter term. We note that here (i.e. in (SURF-MATCH)) we do not need to enforce manifoldness constraints as each cycle represents a single triangle of a patch of $\mathcal{X}$. By the definition of the product space and according to flow constraints, such a cycle can only be matched to another triangle, edge or vertex of the corresponding patch on shape $\mathcal{Y}$.

The solution space of (SURF-MATCH) includes matchings which do not preserve extrinsic orientation of surface (i.e. matchings that flip normal vectors). We regularise this by employing an orientation preservation term. In particular, for every edge $e = \left(\binom{v_\mathcal{C}}{v_\mathcal{Y}}, \binom{\bar{v}_\mathcal{C}}{\bar{v}_\mathcal{Y}}\right) \in \tilde{\mathcal{P}}^\triangle$ in the product graph we compute $r(e) = \max(\phi - \frac{\pi}{4}, 0)$ where ϕ is the angle (in radians) between edges $(v_\mathcal{C}, \bar{v}_\mathcal{C}) \in \mathcal{E}_{\mathcal{C}_1}^\triangle \cup \dots \cup \mathcal{E}_{\mathcal{C}_m}^\triangle$ and $(v_\mathcal{Y}, \bar{v}_\mathcal{Y}) \in \mathcal{Y}$ *after* mapping the vertices of both patches to the unit disk and subsequently aligning their boundaries, see Fig. 5 for an illustration. The intuition for this is, that finding matchings between two unit disks (e.g. through nearest neighbours) after placing them on top of each other (such that normal vectors align) does overall preserve extrinsic orientations.

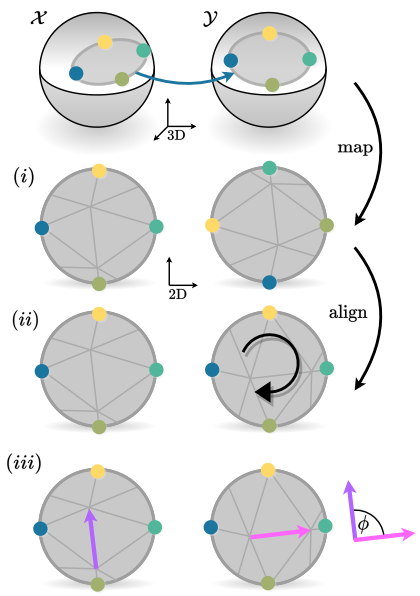


Figure 5: *We compute the* ***orientation preservation regulariser*** *as follows: (i) we map corresponding patches to the unit disk using [EDD*95] implemented in [JP*18]. (ii) we rigidly align both unit disks by solving an orthogonal Procrustes problem using the matched boundaries [Sch66]. (iii) for every edge* $e = \left(\binom{v_\mathcal{C}}{v_\mathcal{Y}}, \binom{\bar{v}_\mathcal{C}}{\bar{v}_\mathcal{Y}}\right) \in \tilde{\mathcal{P}}^\triangle$ *of the product graph (i.e. for every edge pair* $(v_\mathcal{C}, \bar{v}_\mathcal{C})$ *and* $(v_\mathcal{Y}, \bar{v}_\mathcal{Y})$*) we compute the angle* ϕ *between respective edges of* $\mathcal{X}$ *and* $\mathcal{Y}$ *mapped and aligned onto the 2D unit disk.*

5. Experiments

In this section, we evaluate the performance of our method. To this end, we first describe how we setup experiments, which metrics we use and to which baselines we compare on which dataset. After that, we evaluate runtime as well as the shape matching performance of our method compared to baselines.

5.1. Setup

Hardware and Solver

We conduct all runtime experiments on CPU on a AMD EPYC 9274F 24-Core Processor with 756 GB DDR5-4800 RAM. For runtime evaluation on GPU, we use a single NVIDIA L40 with 48GB GDDR6 VRAM. We use off-the-shelf integer linear programming solver Gurobi [Gur23] to solve (PATCH-MATCH) and (SURF-MATCH). For quantitative shape matching evaluation we use a time budget of one hour for each stage, i.e. we allow for a total computation time of *at most* two hours.

Shape Processing and Patch Extraction

For quantitative evaluation, we decimate shapes to 1,000 triangles using algorithms provided in [JP*18], so that quantitative results are comparable with previous methods (we note that we explicitly use the older version v2.5 of [JP*18] as its decimate function produces less discretisation artefacts). We repair any defects in the shapes using [Att10]. To avoid issues with discretisation, we follow [RB25] and use a distortion bound with $k = 2$ (the distortion bound additionally allows to match edges of $\mathcal{Y}$ to vertices of $\mathcal{X}$). For cost computation, similar to previous methods [RB24; RAC*24; RB25], we extract features on full-resolution shapes using pre-trained feature extractor [CRB23] using 25 refinement iterations and transfer these features to lower-resolution shapes. Furthermore, for the adaptive loss function $\Psi(\cdot)$ used in Eq. (3), we set $\alpha = 2$ and $c = 0.3$, see [Bar19] for more details. For our orientation preservation regulariser, we use $\lambda = d_{\max}$, i.e. the maximum of d. We extract $60 - 100$ patches on the shapes as follows: (i) we perform furthest point sampling and region growing while constraining regions to a maximum number of faces (we use 100). (ii) we detect regions with more than one boundary and subdivide such regions into smaller regions so that each region only has a single boundary and consequently yields a valid patch.

Metrics

Next to computation times we use the following two metrics to evaluate the performance of methods w.r.t. matching quality and geometric consistency respectively.

Geodesic errors describe the geodesic distance (on the surface of a shape) of a matched vertex to its ground-truth vertex. We follow the Princeton protocol [KLF11] and normalise errors by the square-root of the shape area, see [KK91, Sec. 8.2].

Dirichlet energies describe the deformation energy induced by the matching and with that allow to evaluate geometric consistency, see [RB24, Sec. 8] for more details.

Datasets

We evaluate our method on three datasets including humanoid shapes as well as animal shapes:

FAUST [BRLB14] comprises of 100 human shapes from which we sample 100 pairs from the test set for evaluation. We use the more challenging, re-meshed version [RPWO18; DSO20].

SMAL [ZKJB17] contains 49 four-legged animal shapes from 8 different animal categories. We sample 100 pairs from the test set for evaluation.

DT4D [MRSO22] is based on the large animation dataset DeformingThings4D [LTT*21]. It consists of 9 different classes of game character shapes. We sample 100 inter class and 100 intra class test pairs from the test set and denote them DT4D Inter and DT4D intra, respectively.

Baselines

We compare our method to axiomatic shape matching approaches and furthermore to a representative learning-based method.

SpiderMatch [RB24] represents the source shape with a single, self-intersecting, cyclic curve. It enforces geometric consistency through constraints in an integer linear program which preserve intersections of the curve and avoid new intersections of the curve when matched to the target shape.

GeCo [RB25] represents every triangle of the source shape with an individual cyclic curve and ensures geometric consistency by glueing cyclic curves together at opposite edges through constraints in an integer linear program.

DiscrOpt [RMWO21] is a functional-map based method that uses discrete optimisation with hard constrains which enforce that the functional map gives rise to a point map.

SmoothShells [ELC20] is a functional-map based method that aligns shapes and iteratively adds more geometric information to refine the matching

ULRSSM [CRB23] is an unsupervised learning-based approach that builds on [RMWO21] and relates the functional map to a point map softly through a loss. We note that we use this method to extract features for our method. In addition, this method provides features for SpiderMatch [RB24] and GeCo [RB25].

For runtime comparisons, we compare our method against several more, geometrically consistent shape matching methods (we note that we compute cost functions for all methods using features extracted with [CRB23]).

Windheuser et al. [WSSC11a] introduce an integer linear program for matching triangles on both shapes. Geometric consistency is enforced through constraints which ensure that neighbouring triangles are glued together at opposite edges.

SM-COMB [RSCB22] builds on the formalism by Windheuser et al. [WSSC11a] and provides a heuristic solver running on CPU.

DiscoMatch [RAC*24] also builds on the formalism by Windheuser et al. [WSSC11a] and provides a heuristic solver running on GPU employing second order updates.

5.2. Runtime Experiments

In Fig. 6, we compare the runtime of several geometrically consistent methods. To this end, we sample 5 instances from FAUST dataset and vary the resolution of both shapes, for which we show qualitative results at a resolution of 10,000 triangles in Fig. 1 right and in Fig. 14 in the supplementary. From the runtime results, we can see that our method is the only scalable method and with that, the only method that can compute geometrically consistent matching results for shapes with up to 10,000 triangles. This improved scalability stems from our two-stage approach which keeps the size of the optimisation problems, especially of (PATCH-MATCH) in stage (i), smaller.

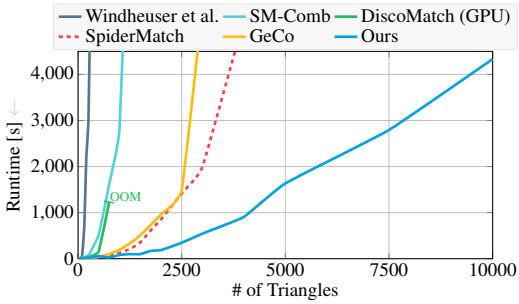


Figure 6: ***Runtime comparison*** *over five pairs of FAUST dataset (we show median runtimes) of various geometrically consistent shape matching methods of which our method is the most scalable. We note that SpiderMatch [RB24] only supports a weak form of geometric consistency and thus we plot it dashed. The bump of our method at around 4,000 triangles very likely stems from discretisation artefacts which make our problem less easy to solve.*

Furthermore, in Fig. 7, we compare the computation times of the two stages of our method, which shows that the first stage is computationally more demanding than the second stage. We show detailed numbers to the plots in Fig. 7 in the supplementary in Section 10.

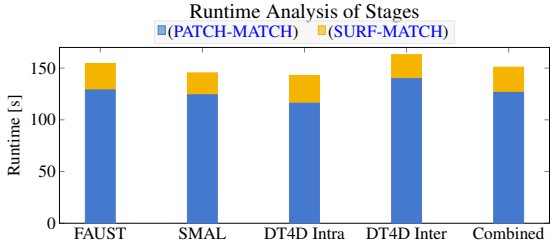


Figure 7: ***Runtime analysis of the two stages*** *of our method on all datasets for shape resolution of* 1000 *triangles. We show bar plots with bar height indicating runtime in seconds. Combined is the median over all datasets. We can see that most time of our method is spent in solving* (PATCH-MATCH).

5.3. 3D Shape Matching Experiments

In Fig. 8, we show the relation of runtime, geodesic errors and Dirichlet energies for results computed with competing methods. From that, we can deduce that our method yields the best balance between runtime, matching smoothness and matching quality among the geometrically consistent methods. In Section 9 in the supplementary, we provide detailed results on geodesic errors and Dirichlet energies.

This is also confirmed through the qualitative results shown in Fig. 9 from which can see that our method produces visually indistinguishable results compared to other geometrically consistent methods. In addition to that, in Fig. 1 and Fig. 10, we show matching results of our method on high-resolution shapes with up to 10,000 triangles. For more qualitative results on high-resolution shapes, we refer to Section 8 in the supplementary.

Finally, we evaluate the robustness of our method. To this end, we show quantitative results on shapes with different discretisation in Fig. 11, we evaluate the influence of the number of patches in Table 2, and we show, as a proof-of-concept, qualitative results of partial shapes in Fig. 12. These experiments show that our method is applicable to more challenging settings.

Approx. # of Patches	25	50	75	100
Mean Geo. Errors	0.030	0.032	0.030	0.029
Mean Runtime [s]	114	143	211	280

Table 2: *We evaluate the influence of the* ***choice of patches*** *by extracting a different number of patches on the source shapes for five pairs of FAUST dataset. We can see that more patches increase the runtime (due to a resulting larger optimisation problem in our first stage), yet matching errors remain similar. This indicates that our method is robust to different numbers of patches.*

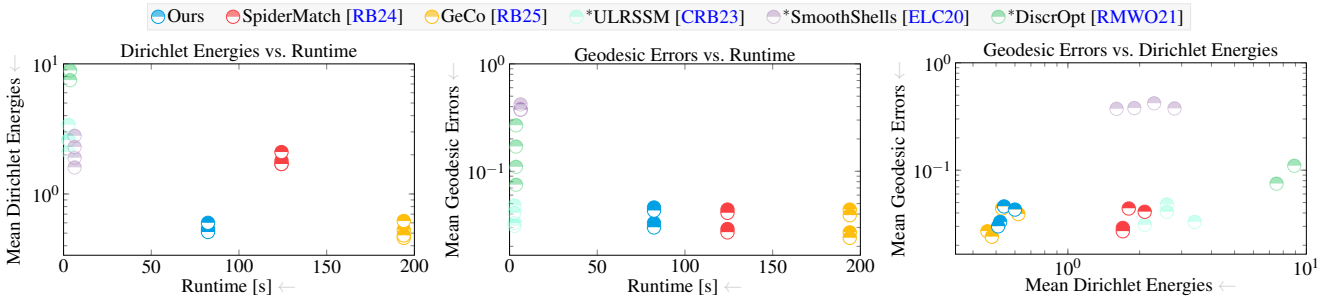


Figure 8: *Scatter plots of competing methods showcasing the **relationship between runtime, geodesic errors and Dirichlet energies**. Each point represents mean geodesic errors or rather mean Dirichlet energies over a full dataset. For all plots, bottom left corners are best. For completeness, we include methods which do not incorporate geometric consistency (i.e. methods marked with*: ULRSSM [CRB23], SmoothShells [MRSO22], and DiscrOpt [RMWO21]) while drawing them with decreased opacity. These scatter plots show that our method poses the best balance between matching quality, matching smoothness, and runtime.*

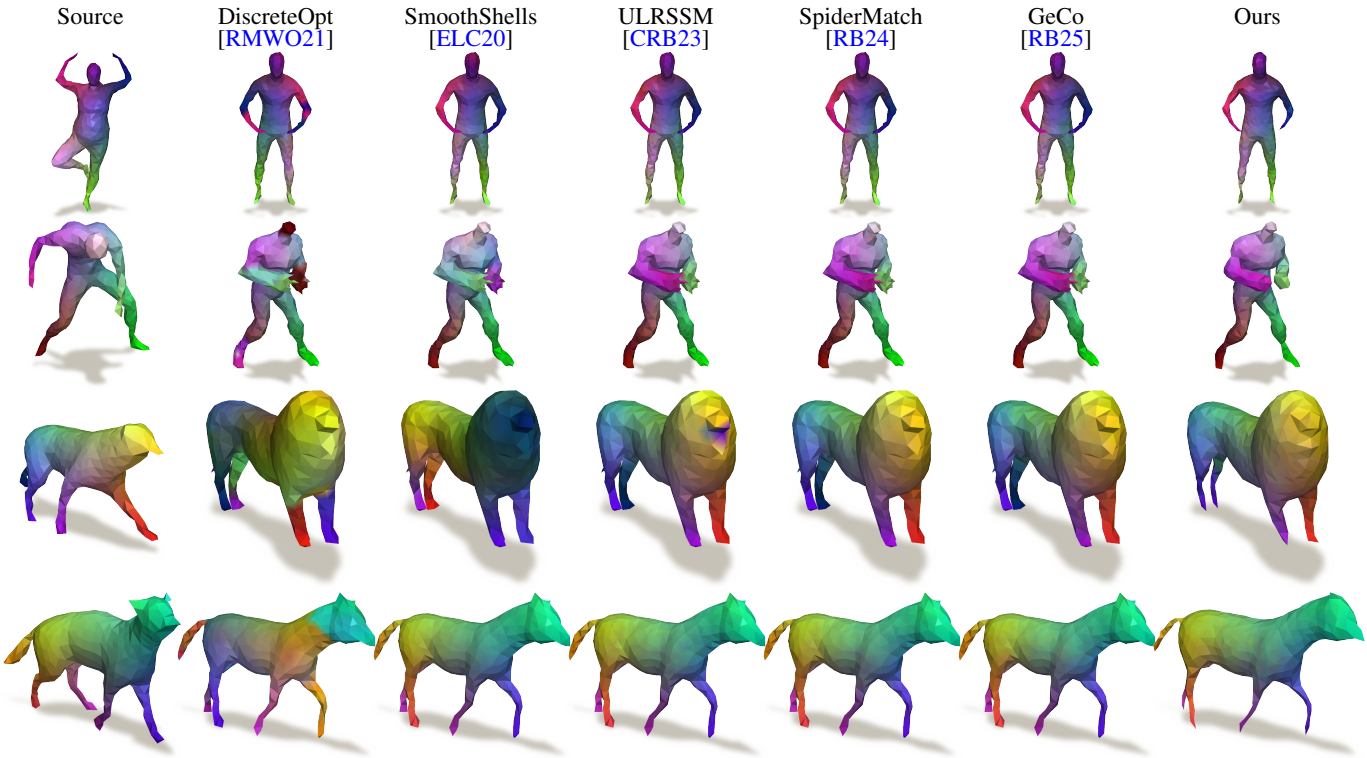


Figure 9: ***Qualitative comparison** on shapes with a resolution of 1,000 triangles. DiscreteOpt [RMWO21] does not employ any smoothness constraints and thus matchings violate neighbourhoods in many cases. SmoothShells [ELC20] yields smooth results but suffers from flips (see e.g. upper body of shape in second row). Overall ULRSSM [CRB23] produces very good results. Yet, it still suffers from local miss-matches (cf. e.g. left hand of shape in second row or head of lion in third row) due to not enforcing geometric consistency. In contrast, such mis-matches are not apparent in any of the geometrically consistent methods (i.e. SpiderMatch [RB24], GeCo [RB25] and ours).*

6. Discussion & Limitations

We have shown that our method produces high-quality matching results and scales to high-resolution shapes. Yet, our approach has certain drawbacks.

Even though we can effectively avoid orientation flips in the second stage (using our orientation preserving regularisation term), we cannot employ this term for the first stage. Consequently, in the worst case, the resulting matching of the first stage might be completely flipped inside-out and thus constrains the second stage such that the resulting matching will also be flipped inside-out. Nevertheless, we did not experience this case in any of our experiments of more than 400 shape matching instances.

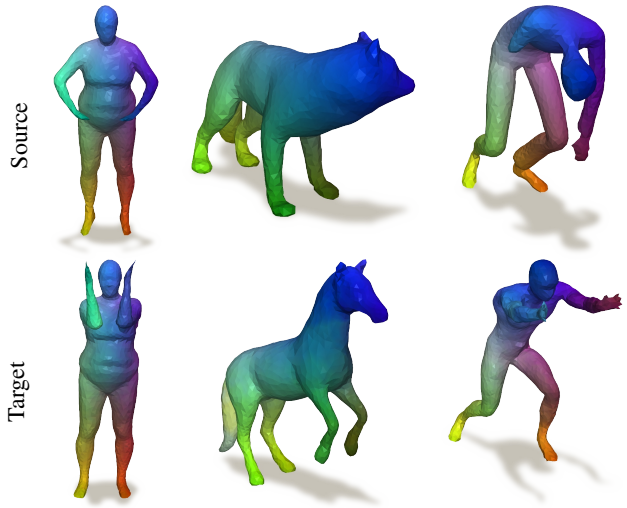


Figure 10: ***Qualitative results*** *of our method on shapes with a resolution of 5,000 triangles. By comparison to qualitative results in Fig. 9 (were shapes have only 1,000 triangles), we can see that the higher resolution shapes shown here allows the shapes to have much more detailed geometry.*

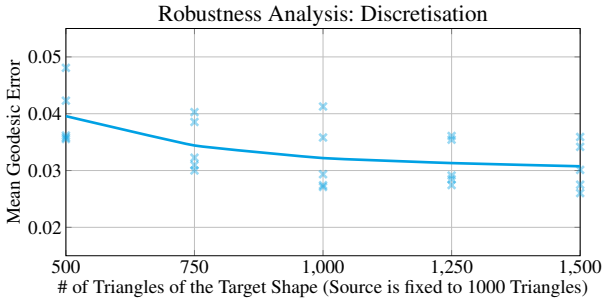


Figure 11: *Evaluation of our method for shapes with different* ***discretisation*** *on five pairs of FAUST dataset. The line shows the mean of individual pairs (crosses). To this end, we vary the resolution of the target shape while keeping the source shape fixed to 1,000 triangles. At lower resolutions, discretisation artefacts yield higher mean geodesic errors while overall errors are similar. This shows that our method supports different discretisation of shapes.*

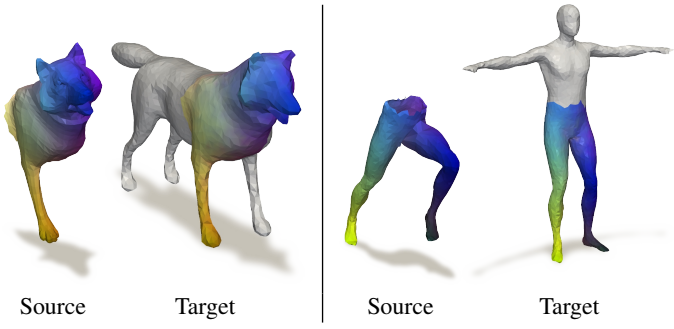


Figure 12: *Qualitative results for* ***partial shape matching*** *on shapes from SHREC'16 [CRB*16] dataset.*

In our experiments, we have shown that our method supports different choices of patches, shapes with boundaries, and shapes with different discretisation. Yet, our geometric consistency constraints (which enforce neighbourhood preservation as a hard constraints) restrict its applicabilities to shape pairs which are manifold and which have the same genus. In future works, non-manifold shapes could be tackled with soft geometric consistency constraints (i.e. constraints which allow for local geometric consistency violations). Further, shapes with different genus would require to redefine the notion of geometric consistency in an appropriate sense.

For the first time, we present an efficiently solvable integer linear program for geometrically consistent shape matching that is not limited by runtime but rather by memory consumption. In fact, our method requires up to 500GB of memory on shape pairs with 10,000 triangles due to the quadratic growth of the number of variables and linear programming solvers requiring constraint matrices to be pre-computed and stored in memory. In future works, high memory requirements could be resolved through e.g. custom solvers that construct constraint matrices on the fly and further exploit structural properties of our constraints, while runtimes could be improved using approximate solvers, e.g. on GPU [RAC*24].

We show that we can push our method to handle meshes with up to 10,000 triangles, a resolution for which geometrically consistent shape matching has not been achieved before. Yet, resulting runtimes and memory consumption might still be too demanding for certain applications, that for example require real-time processing. Despite this, our method is faster than previous (geometrically consistent) methods, even for lower resolutions shapes. While the trade-off between resolution and runtime depends on the application, our method is particularly favourable for shapes with about 1,500-10,000 triangles (cf. Fig. 6).

7. Conclusion

We have presented the first geometrically consistent and globally optimal approach which scales to shape resolutions of up to 10,000 triangles. In contrast to previous (geometrically consistent) methods, we do not rely on a local coarse-to-fine strategy, but rather can compute global matchings directly on high-resolution shapes. Overall, we believe that our method is an important contribution to the visual computing community as it provides a tool to compute guaranteed smooth correspondences on shapes with practically relevant resolutions and thus brings novel downstream applications into reach.

Acknowledgments

This work is supported by the ERC starting grant no. 101160648 (Harmony). Open Access funding enabled and organized by Projekt DEAL

References

[AO23] ATTAIKI, SOUHAIB and OVSJANIKOV, MAKS. "Understanding and improving features learned in deep functional maps". *CVPR*. 2023 2.

[Att10] ATTENE, MARCO. "A lightweight approach to repairing digitized polygon meshes". *The visual computer* (2010) 7.

[Bar19] BARRON, JONATHAN T. "A general and adaptive robust loss function". *CVPR*. 2019 5, 7.

[BBK*10] BRONSTEIN, ALEXANDER M, BRONSTEIN, MICHAEL M, KIMMEL, RON, et al. "A Gromov-Hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching". *International Journal of Computer Vision* (2010) 2.

[BK17] BURGHARD, OLIVER and KLEIN, REINHARD. "Efficient lifted relaxations of the quadratic assignment problem". *Proceedings of the conference on Vision, Modeling and Visualization*. 2017 3.

[BLG*21] BENKNER, MARCEL SEELBACH, LÄHNER, ZORAH, GOLYANIK, VLADISLAV, et al. "Q-match: Iterative shape matching via quantum annealing". *ICCV*. 2021 3.

[BRLB14] BOGO, FEDERICA, ROMERO, JAVIER, LOPER, MATTHEW, and BLACK, MICHAEL J. "FAUST: Dataset and evaluation for 3D mesh registration". *CVPR*. 2014 7.

[BST20] BERNARD, FLORIAN, SURI, ZEESHAN KHAN, and THEOBALT, CHRISTIAN. "MINA: Convex mixed-integer programming for non-rigid shape alignment". *CVPR*. 2020 2.

[BXNL24] BASTIAN, LENNART, XIE, YIZHENG, NAVAB, NASSIR, and LÄHNER, ZORAH. "Hybrid functional maps for crease-aware non-isometric shape matching". *CVPR*. 2024 2.

[CK15] CHEN, QIFENG and KOLTUN, VLADLEN. "Robust nonrigid registration by convex optimization". *ICCV*. 2015 2.

[CRB*16] COSMO, LUCA, RODOLA, EMANUELE, BRONSTEIN, MICHAEL M, et al. "SHREC'16: Partial matching of deformable shapes". *Proc. 3DOR* (2016) 10.

[CRB23] CAO, DONGLIANG, ROETZER, PAUL, and BERNARD, FLORIAN. "Unsupervised Learning of Robust Spectral Shape Matching". *TOG* (2023) 2, 7–9, 13.

[CYES00] COUGHLAN, JAMES, YUILLE, ALAN, ENGLISH, CAMPER, and SNOW, DAN. "Efficient deformable template detection and localization without user initialization". *Computer Vision and Image Understanding* (2000) 3.

[DCO22] DONATI, NICOLAS, CORMAN, ETIENNE, and OVSJANIKOV, MAKS. "Deep Orientation-Aware Functional Maps: Tackling Symmetry Issues in Shape Matching". *CVPR*. 2022 2.

[DML17] DYM, NADAV, MARON, HAGGAI, and LIPMAN, YARON. "DS++ - A Flexible, Scalable and Provably Tight Relaxation for Matching Problems". *TOG* (2017) 3.

[DMM24] DUTT, NILADRI SHEKHAR, MURALIKRISHNAN, SANJEEV, and MITRA, NILOY J. "Diffusion 3D Features (Diff3F): Decorating Untextured Shapes with Distilled Semantic Features". *CVPR*. 2024 2.

[DSO20] DONATI, NICOLAS, SHARMA, ABHISHEK, and OVSJANIKOV, MAKS. "Deep geometric functional maps: Robust feature learning for shape correspondence". *CVPR*. 2020 2, 7.

[DYDZ22] DENG, BAILIN, YAO, YUXIN, DYKE, ROBERTO M, and ZHANG, JUYONG. "A Survey of Non-Rigid 3D Registration". *Computer Graphics Forum*. 2022 2.

[EB17] EZUZ, DANIELLE and BEN-CHEN, MIRELA. "Deblurring and denoising of maps between shapes". *Computer Graphics Forum*. Wiley Online Library. 2017 2, 3.

[EDD*95] ECK, MATTHIAS, DEROSE, TONY, DUCHAMP, TOM, et al. "Multiresolution analysis of arbitrary meshes". *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 1995 7.

[EGR*24] EHM, VIKTORIA, GAO, MAOLIN, ROETZER, PAUL, et al. "Partial-to-partial shape matching with geometric consistency". *CVPR*. 2024 3.

[EHA*19] EZUZ, DANIELLE, HEEREN, BEHREND, AZENCOT, OMRI, et al. "Elastic correspondence between triangle meshes". *Computer Graphics Forum*. 2019 2, 3.

[ELC20] EISENBERGER, MARVIN, LAHNER, ZORAH, and CREMERS, DANIEL. "Smooth shells: Multi-scale shape registration with functional maps". *CVPR*. 2020 2, 8, 9, 13.

[ERE*24] EHM, VIKTORIA, ROETZER, PAUL, EISENBERGER, MARVIN, et al. "Geometrically Consistent Partial Shape Matching". *3DV*. 2024 3.

[Fel05] FELZENSZWALB, PEDRO F. "Representation and detection of deformable shapes". *TPAMI* (2005) 3.

[GCR*17] GASPARETTO, ANDREA, COSMO, LUCA, RODOLA, EMANUELE, et al. "Spatial maps: From low rank spectral to sparse spatial functional representations". *3DV*. IEEE. 2017 2.

[GFK*18] GROUEIX, THIBAULT, FISHER, MATTHEW, KIM, VLADIMIR G, et al. "3d-coded: 3d correspondences by deep deformation". *ECCV*. 2018 2.

[GRE*23] GAO, MAOLIN, ROETZER, PAUL, EISENBERGER, MARVIN, et al. "SIGMA: Scale-Invariant Global Sparse Shape Matching". *ICCV*. 2023 2.

[Gur23] GUROBI OPTIMIZATION, LLC. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com 7.

[HHF*21] HUTSCHENREITER, LISA, HALLER, STEFAN, FEINEIS, LORENZ, et al. "Fusion moves for graph matching". *CVPR*. 2021 3.

[HLC20] HOLZSCHUH, BENJAMIN, LÄHNER, ZORAH, and CREMERS, DANIEL. "Simulated annealing for 3d shape correspondence". *3DV*. 2020 3.

[HLR*19] HALIMI, OSHRI, LITANY, OR, RODOLA, EMANUELE, et al. "Unsupervised learning of dense shape correspondence". *CVPR*. 2019 2.

[JP*18] JACOBSON, ALEC, PANOZZO, DANIELE, et al. *libigl: A simple C++ geometry processing library*. https://libigl.github.io/. 2018 7.

[KK91] KIM, WHOI-YUL and KAK, AVINASH C. "3-D object recognition using bipartite matching embedded in discrete relaxation". *IEEE Transactions on Pattern Analysis & Machine Intelligence* (1991) 7.

[KLF11] KIM, VLADIMIR G, LIPMAN, YARON, and FUNKHOUSER, THOMAS. "Blended intrinsic maps". *TOG* (2011) 7.

[KMDL19] KUSHINSKY, YAM, MARON, HAGGAI, DYM, NADAV, and LIPMAN, YARON. "Sinkhorn Algorithm for Lifted Assignment Problems". *SIAM Journal on Imaging Sciences* (2019) 3.

[LDO22] LI, LEI, DONATI, NICOLAS, and OVSJANIKOV, MAKS. "Learning Multi-resolution Functional Maps with Spectral Attention for Robust Shape Matching". *NIPS* (2022) 2.

[LLHL20] LI, QINSONG, LIU, SHENGJUN, HU, LING, and LIU, XINRU. "Shape correspondence using anisotropic Chebyshev spectral CNNs". *CVPR*. 2020 2.

[LRR*17] LITANY, OR, REMEZ, TAL, RODOLA, EMANUELE, et al. "Deep functional maps: Structured prediction for dense shape correspondence". *ICCV*. 2017 2.

[LRS*16] LÄHNER, ZORAH, RODOLA, EMANUELE, SCHMIDT, FRANK R, et al. "Efficient globally optimal 2d-to-3d deformable shape matching". *CVPR*. 2016 3, 4.

[LTT*21] LI, YANG, TAKEHARA, HIKARI, TAKETOMI, TAKAFUMI, et al. "4dcomplete: Non-rigid motion estimation beyond the observable surface". *ICCV*. 2021 7.

[MBRM24] MAGGIOLI, FILIPPO, BAIERI, DANIELE, RODOLÀ, EMANUELE, and MELZI, SIMONE. "Rematching: Low-resolution representations for scalable shape correspondence". *ECCV*. 2024 3.

[MDK*16] MARON, HAGGAI, DYM, NADAV, KEZURER, ITAY, et al. "Point registration via efficient convex relaxation". *TOG* (2016) 2.

[MO23] MAGNET, ROBIN and OVSJANIKOV, MAKS. "Scalable and efficient functional map computations on dense meshes". *Computer Graphics Forum*. 2023 3.

[MO24] MAGNET, ROBIN and OVSJANIKOV, MAKS. "Memory-scalable and simplified functional map learning". *CVPR*. 2024 3.

[MRR*19] MELZI, S, REN, J, RODOLÀ, E, et al. "ZoomOut: Spectral upsampling for efficient shape correspondence". *TOG* (2019) 2, 3.

[MRSO22] MAGNET, ROBIN, REN, JING, SORKINE-HORNUNG, OLGA, and OVSJANIKOV, MAKS. "Smooth non-rigid shape matching via effective Dirichlet energy optimization". *3DV*. 2022 7, 9.

[NH22] NASIKUN, AHMAD and HILDEBRANDT, KLAUS. "The hierarchical subspace iteration method for laplace–beltrami eigenproblems". *TOG* (2022) 3.

[OBS*12] OVSJANIKOV, MAKS, BEN-CHEN, MIRELA, SOLOMON, JUSTIN, et al. "Functional maps: a flexible representation of maps between shapes". *TOG* (2012) 2.

[OMMG10] OVSJANIKOV, MAKS, MÉRIGOT, QUENTIN, MÉMOLI, FACUNDO, and GUIBAS, LEONIDAS. "One point isometric matching with the heat kernel". *Computer Graphics Forum*. 2010 2.

[PPCG19] PROBST, THOMAS, PAUDEL, DANDA PANI, CHHATKULI, AJAD, and GOOL, LUC VAN. "Unsupervised learning of consensus maximization for 3D vision problems". *CVPR*. 2019 2.

[PRLB25] PETZSCH, CHRISTOPH, ROETZER, PAUL, LÄHNER, ZORAH, and BERNARD, FLORIAN. "Approximate 2D-3D Shape Matching for Interactive Applications". *3DV*. 2025 3.

[RAC*24] ROETZER, PAUL, ABBAS, AHMED, CAO, DONGLIANG, et al. "DiscoMatch: Fast Discrete Optimisation for Geometrically Consistent 3D Shape Matching". *ECCV*. 2024 1–3, 7, 8, 10.

[RB24] ROETZER, PAUL and BERNARD, FLORIAN. "SpiderMatch: 3D Shape Matching with Global Optimality and Geometric Consistency". *CVPR*. 2024 2–4, 7–9, 13.

[RB25] ROETZER, PAUL and BERNARD, FLORIAN. "Fast Globally Optimal and Geometrically Consistent 3D Shape Matching". *arXiv preprint arXiv:2504.06385* (2025) 1–4, 6–9, 13.

[RBA*12] RODOLA, EMANUELE, BRONSTEIN, ALEX M, ALBARELLI, ANDREA, et al. "A game-theoretic approach to deformable shape matching". *CVPR*. 2012 2.

[REC*25] ROETZER, PAUL, EHM, VIKTORIA, CREMERS, DANIEL, et al. "Higher-Order Ratio Cycles for Fast and Globally Optimal Shape Matching". *CVPR*. 2025 2–4.

[RLB23] ROETZER, PAUL, LÄHNER, ZORAH, and BERNARD, FLORIAN. "Conjugate product graphs for globally optimal 2d-3d shape matching". *CVPR*. 2023 3, 4.

[RMWO21] REN, JING, MELZI, SIMONE, WONKA, PETER, and OVSJANIKOV, MAKS. "Discrete optimization for shape matching". *Computer Graphics Forum*. 2021 2, 8, 9, 13.

[RPW94] RENDL, F, PARDALOS, P, and WOLKOWICZ, H. "The Quadratic Assignment Problem: A Survey and Recent Developments". *DIMACS workshop*. 1994 3.

[RPWO18] REN, JING, POULENARD, ADRIEN, WONKA, PETER, and OVSJANIKOV, MAKS. "Continuous and orientation-preserving correspondences via functional maps". *TOG* (2018) 7.

[RSCB22] ROETZER, PAUL, SWOBODA, PAUL, CREMERS, DANIEL, and BERNARD, FLORIAN. "A scalable combinatorial solver for elastic geometrically consistent 3d shape matching". *CVPR*. 2022 1–3, 8.

[RSO19] ROUFOSSE, JEAN-MICHEL, SHARMA, ABHISHEK, and OVSJANIKOV, MAKS. "Unsupervised deep learning for structured shape matching". *ICCV*. 2019 2.

[SACO22] SHARP, NICHOLAS, ATTAIKI, SOUHAIB, CRANE, KEENAN, and OVSJANIKOV, MAKS. "Diffusionnet: Discretization agnostic learning on surfaces". *TOG* (2022) 2.

[SAPH04] SCHREINER, JOHN, ASIRVATHAM, ARUL, PRAUN, EMIL, and HOPPE, HUGUES. "Inter-surface mapping". *SIGGRAPH*. 2004 2, 3.

[SC09] SCHOENEMANN, THOMAS and CREMERS, DANIEL. "A combinatorial solution for model-based image segmentation and real-time tracking". *TPAMI* (2009) 3.

[SCBK20] SCHMIDT, PATRICK, CAMPEN, MARCEL, BORN, JANIS, and KOBBELT, LEIF. "Inter-surface maps via constant-curvature metrics". *TOG* (2020) 2, 3.

[Sch66] SCHÖNEMANN, PETER H. "A generalized solution of the orthogonal procrustes problem". *Psychometrika* (1966) 7.

[SHCB11] SHARMA, AVINASH, HORAUD, RADU, CECH, JAN, and BOYER, EDMOND. "Topologically-robust 3D shape matching based on diffusion geometry and seed growing". *CVPR*. 2011 2, 3.

[SK83] SANKOFF, DAVID and KRUSKAL, JOSEPH B. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley Publishing Company, 1983 3.

[SPK23] SCHMIDT, PATRICK, PIEPER, DÖRTE, and KOBBELT, LEIF. "Surface Maps via Adaptive Triangulations". *Computer Graphics Forum* (2023) 2, 3.

[SPKS16] SOLOMON, JUSTIN, PEYRÉ, GABRIEL, KIM, VLADIMIR G, and SRA, SUVRIT. "Entropic metric alignment for correspondence problems". *RoG* (2016) 3.

[STC09] SCHMIDT, FRANK R., TÖPPE, ENO, and CREMERS, DANIEL. "Efficient Planar Graph Cuts with Applications in Computer Vision". *CVPR*. 2009 3.

[SVB19] SHOHAM, MEGED, VAXMAN, AMIR, and BEN-CHEN, MIRELA. "Hierarchical functional maps between subdivision surfaces". *Computer Graphics Forum*. 2019 3.

[Tak22] TAKAYAMA, KENSHI. "Compatible intrinsic triangulations". *ACM Transactions on Graphics (TOG)* (2022) 2, 3.

[TCL*12] TAM, GARY KL, CHENG, ZHI-QUAN, LAI, YU-KUN, et al. "Registration of 3D point clouds and meshes: A survey from rigid to nonrigid". *IEEE transactions on visualization and computer graphics* (2012) 2.

[TCM*21] TRAPPOLINI, GIOVANNI, COSMO, LUCA, MOSCHELLA, LUCA, et al. "Shape registration in the time of transformers". *NeurIPS* (2021) 2.

[VBG10] VAXMAN, AMIR, BEN-CHEN, MIRELA, and GOTSMAN, CRAIG. "A multi-resolution approach to heat kernels on discrete surfaces". *SIGGRAPH*. 2010 3.

[VLR*17] VESTNER, MATTHIAS, LITMAN, ROEE, RODOLA, EMANUELE, et al. "Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space". *CVPR*. 2017 2, 3.

[VZHC11] VAN KAICK, OLIVER, ZHANG, HAO, HAMARNEH, GHASSAN, and COHEN-OR, DANIEL. "A survey on shape correspondence". *Computer graphics forum*. Wiley Online Library. 2011 2.

[WEH20] WIERSMA, RUBEN, EISEMANN, ELMAR, and HILDEBRANDT, KLAUS. "Cnns on surfaces using rotation-equivariant features". *ACM Transactions on Graphics (ToG)* (2020) 2.

[WSSC11a] WINDHEUSER, THOMAS, SCHLICKEWEI, ULRICH, SCHMIDT, FRANK R, and CREMERS, DANIEL. "Geometrically consistent elastic matching of 3d shapes: A linear programming solution". *ICCV*. 2011 2, 3, 8.

[WSSC11b] WINDHEUSER, THOMAS, SCHLICKWEI, ULRICH, SCHIMDT, FRANK R, and CREMERS, DANIEL. "Large-scale integer linear programming for orientation preserving 3d shape matching". *Computer Graphics Forum*. Wiley Online Library. 2011 2, 3.

[ZKJB17] ZUFFI, SILVIA, KANAZAWA, ANGJOO, JACOBS, DAVID W, and BLACK, MICHAEL J. "3D menagerie: Modeling the 3D shape and pose of animals". *CVPR*. 2017 7.