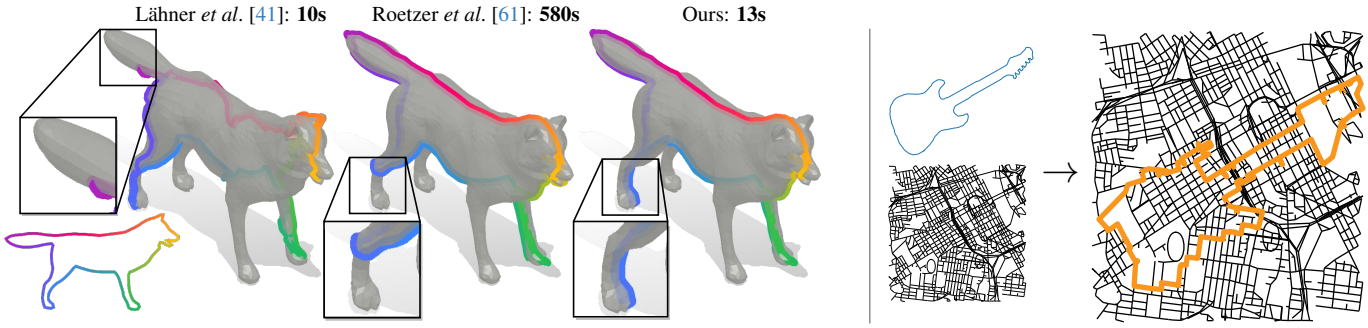
This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

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

Paul Roetzer[1] Viktoria Ehm[3,4] Daniel Cremers[3,4] Zorah Lähner[1,2] Florian Bernard[1,2]
[1] University of Bonn [2] Lamarr Institute [3] TU Munich [4] MCML



2D-3D shape matching on wolf (2D outline is shown in the bottom left) with various methods

Guitar outline and road network to create GPS art

Figure 1. **Left:** We show results on 2D-3D shape matching computed with Lähner *et al*. [41], Roetzer *et al*. [61] and our approach. Competing methods either have fast computation times, or high-quality matchings, but not both at the same time. Instead, our method combines both desirable properties and even further improves accuracy as we effectively avoid bias towards shorter matchings (see zoomed-in parts). **Right:** Our algorithm can be used to find object-shaped tracks in road networks, known as GPS-art, such as the shape of a guitar in the road network of Nashville (represented as a graph, compute time 1s).

Abstract

In this work we address various shape matching problems that can be cast as finding cyclic paths in a product graph. This involves for example 2D-3D shape matching, 3D shape matching, or the matching of a contour to a graph. In this context, matchings are typically obtained as the minimum cost cycle in the product graph. Instead, inspired by related works on model-based image segmentation [68], we consider minimum ratio cycles, which we combine with the recently introduced conjugate product graph in order to allow for higher-order matching costs. With that, on the one hand we avoid the bias of obtaining matchings that involve fewer/shorter edges, while on the other hand we are able to impose powerful geometric regularisation, e.g. to avoid zig-zagging. In our experiments we demonstrate that this not only leads to improved matching accuracy in most cases, but also to significantly reduced runtimes (up to two orders of magnitude, depending on the setting). Our GPU implementations are publicly available: https://github.com/paul0noah/product-graph-cycles/.

1. Introduction

Shape matching problems lie at the heart of visual computing, as they are of central importance for numerous downstream applications, including image segmentation [35, 66, 68], 3D reconstruction [24, 45, 71], statistical shape modelling [33] and many more. Many recent shape matching approaches build upon learning-based frameworks [11, 12, 23, 29, 47, 69] that have lead to impressive results. Yet, these frameworks lack important regularisation abilities, such as continuity, geometric consistency, or local rigidity. An important class of approaches that can take into account such desirable properties are *combinatorial/graph-theoretic* shape matching methods.

Combinatorial Approaches to Shape Matching. Recently, an efficient method for geometrically consistent 3D shape matching was proposed [59]. It was shown that state of the art 3D shape matching results can be obtained when building upon cost functions that employ learned feature descriptors. To achieve scalability in 3D to 3D shape matching, the method leverages a graph-theoretic 2D-3D shape matching formalism [41], highlighting the high relevance of graph-based methods for shape matching. In this work, we address several shortcomings of existing graph-based shape

matching approaches in order to push this line of research forward.

Current graph-based approaches [41, 61] find cyclic minimum cost paths (computed with variants of Dijkstra's method) to find globally optimal matchings between 2D and 3D shapes. However, respective methods have a bias towards shorter cycles (see Fig. 1, left, Lähner *et al*. [41] and Roetzer *et al*. [61]) which leads to undesirable results (e.g. unmatched object parts), even if higher-order regularisers that penalise non-rigid deformations are used [61]. Furthermore, the currently best-performing 2D-3D shape matching approach [61] is slow and currently takes around $10-20$ minutes for matching a single shape pair with moderate resolution (e.g. a 3D shape with 3000 vertices and a 2D shape with 500 vertices), see Fig. 1, which is due to the large size of the involved *conjugate product graph*.

Several works have shown that bias towards short cycles can be avoided by considering the *minimum ratio cycle* [13, 35, 66, 68] instead of the minimum cost cycle. These works demonstrate impressive results on image segmentation with near-real time computation times reported on two-decades old hardware. Yet, the underlying formalisms have not been transferred to other tasks, possibly due to the lack of a broader exposure and the unavailability of high-performance implementations.

Contribution. In this work, we fill this gap and demonstrate that minimum ratio cycle algorithms [66, 68] are suitable to solve a much broader range of shape matching problems. To this end, we consider 2D-3D shape matching, 3D shape matching and shape to graph matching. We use the latter to obtain GPS-tracks that resemble a certain user-defined shape (also known as GPS-art), see Fig. 1, right. Especially for 2D-3D shape matching on conjugate graphs we get speed-ups of $10\times$ to $100\times$, while having comparable or better matching accuracy. In summary, our contributions are as follows, see also Tab. 1:

- For the first time we address shape matching problems by finding ratio cycles in product graphs.
- By doing so, we are able to simultaneously avoid bias towards shorter solutions, and to take into account powerful higher-order regularisers that can for example be used to achieve local rigidity.
- We make our high-performance GPU implementations for computing minimum cost *and* ratio cycles publicly available. They can be easily adopted to address different variants of shape matching problems, including 2D-3D shape matching, 3D shape matching, and shape to graph matching.
- Experimentally we demonstrate that our approach achieves state of the art shape matching accuracy, while at the same time being significantly faster compared to its (best-performing product graph-based) competitors.

Method	Demonstrated Applications	Unbiased	Scalable	Higher-Order
Lähner *et al*. [41]	2D3D	✗	✓	✗
Roetzer *et al*. [61]	2D3D	✗	✗	✓
SpiderMatch [59]	3D3D	✗	(✓)	✗
Ours	2D3D, 3D3D, GPS-Art	✓	✓	✓

Table 1. Comparison of **path-based** shape matching methods.

2. Related Work

In this section we summarise works most relevant to our approach. We start with reviewing general matching problems, followed by path-based matching problems. Eventually we discuss optimal ratio methods, along with related vision applications.

Matching Problems. Finding correspondences between shapes can, in principle, be formulated as a linear assignment problem (LAP). However, this typically does not lead to accurate matchings, as it omits geometric relations, such as neighbourhoods. The quadratic assignment problem (QAP) considers pairs of vertices and is thus able to take geometric relations into account. Yet, the QAP is NP-hard [58], so that it does not scale to large problems as they occur in practical visual computing settings. To circumvent this, existing approaches consider convex relaxations for the QAP [6, 19, 40, 64], or use some forms of heuristics [30, 44, 70]. While the LAP and QAP are general (graph) matching problems and grow with the size of individual shapes, shape matching can be solved in a lower dimensional space using functional maps [22, 27, 50, 55, 57]. Furthermore, there are sparse methods [7, 14, 26] that exhibit reduced computation times. A range of other dense shape matching approaches have been developed that are able to enforce desirable geometric properties of the matching [20, 21, 60, 72, 73], including solvers on GPU [1, 62].

Our approach combines many of the desired properties of the aforementioned methods: it can work on dense shapes, enforces certain geometric properties, and is fast due to algorithmic simplicity that allows for a convenient GPU implementation.

Path-Based Matching. A special line of shape matching are path-based approaches. While there are methods that find geodesic shortest paths [28, 36] in shape space, these are conceptually quite different to the path-based matching approaches that we consider in this work. Instead, we will turn our attention to graph-based approaches operating on the discrete domain. In the 2D domain, this includes matching of open contours based on dynamic time warping [63], and the matching of closed contours [51, 65]. Furthermore, there are works that use path-based methods to find matchings between 2D and 3D shapes [41, 61]. This idea was

recently extended to 3D shape matching [59], where additional geometric consistency constraints have been employed.

In this work we tackle path-based shape matching and show that we outperform previous path-based works, both in terms of matching quality and computation time. Opposed to existing path-based shape matching that consider minimum costs, we consider minimum ratios as objective.

Ratio Cycles. A special case of the ratio cycle problem is the minimum mean cycle problem, for which the first algorithm was developed by Karp [37]. Several follow-up works have improved the time and memory complexity [15, 17, 32, 49, 54, 74]. There also exists approximate algorithms for the minimum mean cycle problem based on optimal transport [2].

The ratio cycle problem allows for arbitrary per-edge cost and time functions. Various algorithms exists for finding optimal ratio cycles on general graphs, including algorithms by Howard [34], Hartmann-Orlin [32], Lawler [42], and Karp & Orlin [38]. For a comparison of these algorithms we refer to the survey [16]. Further, recent algorithmic improvements have been made in [10].

The concept of ratio optimisation was first introduced to the computer vision audience in [35] for segmenting images, and later extended to curvature-regularized segmentation [66, 67] and multiview-reconstruction [39]. Other works use ratio cycles to find minimum curves and surfaces, e.g. used for graph cuts [53], or to smoothen initial contours on surface meshes [13].

In our work, we build upon these insights in order to smoothly match a contour onto a mesh (or a general graph).

3. Ratio-Based Shape Matching

In our work we aim to efficiently compute matchings between geometric data, for which we specifically focus on a diverse range of shape matching problems that can be cast as finding a (cyclic) path in a product graph. This in particular includes the matching of a contour to a 3D shape [41, 61] (2D-3D shape matching), the matching of pairs of 3D shapes [59] (3D shape matching), and the matching of a contour to a graph (see Fig. 1, right). While most existing approaches in this direction find matchings based on the shortest (cyclic) path [41, 59, 61], we instead consider an optimal ratio cycle, as proposed for example by Schoenemann and Cremers [68] in the context of model-based image segmentation. In this formulation, the ratio allows to employ two competing properties (which are often called cost and time, hence cost-time ratio cycles). Thus, this enables substantially more modelling flexibility. One important aspect is that this formalism can effectively avoid biased solutions towards paths with fewer/shorter edges, which is the case for shortest paths due to the involved summation of edge weights. Furthermore, this formalism allows to easily integrate a distortion bound, so that the maximum compression of edges from $\mathcal{Y}$ can be controlled, see also Tab. 1 for a comparison of properties of various path-based shape matching approachs.

In the following we first illustrate the relation between cyclic paths in product graphs and shape matching, both for the case of 2D-3D and for 3D matching (Sec. 3.1). Subsequently, we summarise the minimum ratio cycle problem and an algorithm to compute them efficiently in product graphs for shape matching (Sec. 3.2). Eventually, we extend the idea of higher-order matching costs to higher-order ratios – as we will demonstrate, this is crucial to (geometrically) regularise matchings so that for example zig-zagging is avoided (Sec. 3.3). Our notation is summarised in Tab. 2.

Symbol	Description
$\mathcal{C} = (\mathcal{V}_\mathcal{C}, \mathcal{E}_\mathcal{C})$	cyclic chain graph (embedded in 2D or 3D)
$\mathcal{Y} = (\mathcal{V}_\mathcal{Y}, \mathcal{E}_\mathcal{Y})$	directed graph (e.g. 3D surface mesh)
$\mathcal{P} = (\mathcal{V}, \mathcal{E})$	product graph of $\mathcal{C} \times \mathcal{Y}$ with product vertices $\mathcal{V}$ and product edges $\mathcal{E}$
$e,\ v$	edge e, vertex v of $\mathcal{P}$
$\bar{\mathcal{P}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$	conjugate product graph of $\mathcal{P}$ with vertices $\bar{\mathcal{V}}$ and edges $\bar{\mathcal{E}}$
$\bar{e},\ \bar{v}$	edge $\bar{e}$, vertex $\bar{v}$ of $\bar{\mathcal{P}}$

Table 2. Summary of the notation used in this paper.

3.1. Cycle-Based Matching in Product Graphs

Various works have cast shape matching as finding shortest (cyclic) paths in product graphs [41, 59, 61]. In our work we build upon this formalism, which we now summarise. To formulate shape matching as finding such cycles, we first define directed graphs, paths, cycles and cyclic chain graphs:

Definition 1 (Directed Graph). *A graph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ consists of a set of vertices $\mathcal{V}_\mathcal{G}$ and a set of oriented edges $\mathcal{E}_\mathcal{G} \subset \mathcal{V}_\mathcal{G} \times \mathcal{V}_\mathcal{G}$. Every edge $e \in \mathcal{E}_\mathcal{G}$ has an associated edge cost $C(e) \in \mathbb{R}_0^+$ and an edge time $T(e) \in \mathbb{R}_0^+$.*

Definition 2 (Path). *A path $\zeta = \big(e_1, \dots, e_{|\zeta|}\big)$ in graph $\mathcal{G}$ is a sequence of edges $e_i \in \mathcal{E}_\mathcal{G}$, such that for $i = 1, \dots, |\zeta| - 1$ we have that $v'_i = v_{i+1}$, where $e_i = (v_i, v'_i)$, $e_{i+1} = (v_{i+1}, v'_{i+1})$.*

Definition 3 (Cycle). *A cycle $\Gamma = \big(e_1, \dots, e_{|\Gamma|}\big)$ is a path with the additional property that $v_1 = v'_{|\Gamma|}$, where $e_1 = (v_1, v'_1)$, $e_{|\Gamma|} = (v_{|\Gamma|}, v'_{|\Gamma|})$.*

Definition 4 (Cyclic Chain Graph). *A cyclic graph 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 and one outgoing edge.*

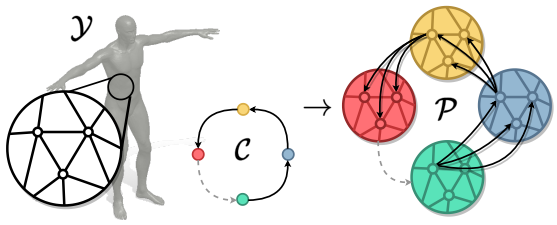


Figure 2. The **product graph** between shapes $\mathcal{C}$ and $\mathcal{Y}$. Every coloured set of vertices of $\mathcal{P}$ refers to a single vertex of $\mathcal{C}$ and all vertices of $\mathcal{Y}$, Thus, $\mathcal{P}$ is structured into components which have intra-component edges (—, —, —, —). Furthermore, components are connected via inter-component edges (black). Note, we do *not* show all inter-component edges for a lighter visual appearance.

The shape matching problems that we consider (2D-3D, 3D, contour to graph) can be cast as matching a cyclic chain graph $\mathcal{C}$ (e.g. a contour) to the graph $\mathcal{Y}$ (e.g. the graph of a triangular surface mesh obtained by converting each undirected edge into two directed edges, or a general directed graph). The product graph $\mathcal{P} = (\mathcal{E}, \mathcal{V})$ is a directed graph defined via the (strong) graph product, i.e. $\mathcal{P} = \mathcal{C} \boxtimes \mathcal{Y}$ [31]. In Fig. 2, we visualise the product graph for a 2D outline and a 3D shape. We can see that the product graph $\mathcal{P}$ is structured into components (having individual colours in the figure), where each of the components refers to a single vertex of $\mathcal{C}$ and the whole graph $\mathcal{Y}$. For more details about $\mathcal{P}$, we refer to [41, 59, 61].

To find an optimal matching, most existing formalisms [41, 59, 61] find the shortest (minimum cost) cycle in $\mathcal{P}$, i.e. a cycle which minimises

$$\sigma(\Gamma) := \sum_{e \in \Gamma} C(e), \tag{SP}$$

so that edge costs $C(e)$ are summed along the cycle Γ.

3.2. Minimum Ratio Cycles

Opposed to summing edge weights along a cycle, [35] and [66, 68] have demonstrated the strong advantages of considering minimum ratio cycles in the context of image segmentation.

The Minimum Ratio Cycle Problem. In the minimum ratio cycle problem we aim to find an optimal cycle $\Gamma^\star$ in a graph $\mathcal{G}$ that has the smallest value of

$$\lambda(\Gamma) := \frac{\sum_{e \in \Gamma} C(e)}{\sum_{e \in \Gamma} T(e)}. \tag{CTR}$$

In the literature, $C(e)$ is called the edge-cost, and $T(e)$ is called the edge-time. We emphasise that $T(e)$ does not necessarily have to refer to a temporal quantity. We note that the minimum ratio cycle problem is a generalisation of the minimum mean cycle problem (the minimum mean problem refers to setting $T(e) = 1$ for all edges $e \in \mathcal{E}_\mathcal{G}$) [16, 42, 43]. This emphasises why the minimum ratio cycle can effectively be used to avoid biased solutions containing only few edges (which the minimum cost cycle in (SP) is prone to).

Minimum Ratio Cycle Algorithm. We build upon Lawler's algorithm [35, 42, 43, 66, 68] to find a minimum ratio cycle in a (connected) directed graph. Lawler's algorithm is based on the fact that any sub-optimal chosen ratio λ, i.e. $\lambda > \lambda^\star$, leads to a *negative cycle* in $\mathcal{G}$ when redefining edge weights as $W_\lambda(e) = C(e) - \lambda\, T(e)$. Hence, Lawler's algorithm is based on detecting negative cycles, and iteratively updating λ until no negative cycle exists.

The existence of a negative cycle Γ in $\mathcal{G}$ (with edge weights $W_\lambda(e)$) for a sub-optimal λ can be understood as follows: Γ being a negative cycle means that

$$0 > \sum_{e \in \Gamma} C(e) - \lambda T(e) \tag{1}$$

$$\Leftrightarrow \quad \sum_{e \in \Gamma} \lambda T(e) > \sum_{e \in \Gamma} C(e) \tag{2}$$

$$\Leftrightarrow \quad \lambda > \frac{\sum_{e \in \Gamma} C(e)}{\sum_{e \in \Gamma} T(e)} = \lambda(\Gamma). \tag{3}$$

This shows that the ratio λ is greater than the ratio $\lambda(\Gamma)$ of Γ, so that λ cannot be optimal [16, 42, 43, 68].

Overall, Lawler's algorithm has two main steps that are repeated until no negative cycle exists:

In the **first step**, given a λ, a negative cycle has to be found (in the graph with modified cost $W_\lambda(e)$). This can efficiently be detected by searching for a shortest path from any vertex in the graph to all other vertices with maximum length $|\mathcal{V}_\mathcal{G}|$ (since the cycle length is bounded by the number of vertices in the graph). We tackle this based on a modification of the Moore-Bellman-Ford algorithm [5, 25, 52, 68] that allows for negative edge weights. To improve the running time, during the shortest path search it is beneficial to check for negative cycles in regular intervals, since we are interested in finding *any* negative cycle (and not the one with smallest cost).

In the **second** step, λ needs to be updated to the ratio $\lambda(\Gamma)$ of the negative cycle Γ obtained in the first step (from (3) we know that any negative cycle w.r.t $W_\lambda(e)$ has a smaller ratio than λ). Once there does not exist a negative cycle, there does not exist a cycle Γ' that leads to a lower cost-time ratio, and consequently Γ must be optimal. For the first iteration, λ can be determined by considering any cycle in the graph and selecting λ such that $\sum_{e \in \Gamma} W_\lambda(e) < 0$.

Optimal Ratio Cycles in Product Graphs. Lawler's algorithm can be used to find optimal ratio cycles in arbitrary graphs. However, for matching problems in product graphs we have the additional requirement, that every vertex of $\mathcal{C}$

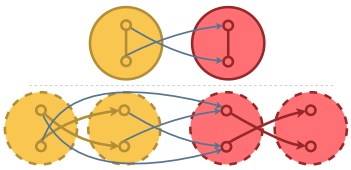


Figure 3. To **avoid intra-component** cycles we duplicate each component k times ($k = 2$ shown in this figure; duplicates are shown at the bottom of the figure). Basically, intra-component edges are carried over to one of their k duplicates, and inter-component edges are duplicated so that every duplicate of one component is connected to the first duplicate of the second component.

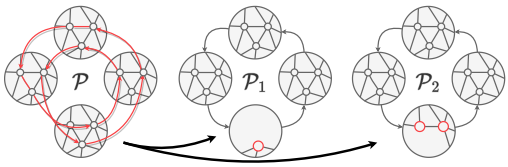


Figure 4. We employ a **branch and bound** strategy whenever the optimal ratio cycle in $\mathcal{P}$ goes through components of $\mathcal{P}$ multiple times (see left). In such scenarios, $\mathcal{P}$ is divided into $\mathcal{P}_1, \mathcal{P}_2$ by removing sets of vertices of one component. Note that we do not show component duplicates here for easier visual exposure.

must be matched to $\mathcal{Y}$. In terms of the cyclic path in $\mathcal{P}$, this means that the cycle must visit every component (cf. Fig. 2, right). This gives rise to two requirements [66, 68]:

(i) We need to avoid that the found cyclic path only involves intra-component edges. This can be achieved by creating $k \in \mathbb{N}^+$ (see below) duplicates of each component, and connecting all of the duplicates to the first duplicate of the sub-sequent component, see Fig. 3 for an illustration. We note that the value of k can be user-defined to bound the maximum distortion, i.e. how many edges in $\mathcal{Y}$ can maximally be compressed into a single vertex of $\mathcal{C}$, see Sec. B in the supplementary. This strategy not only avoids intra-component cycles, but further allows for an efficient implementation: every duplicate of each component can be processed in parallel when performing dynamic programming updates. To achieve faster runtimes, we implement this on the GPU.

(ii) Secondly, we need to avoid multiple passes through the components. This can be achieved by employing a branch-and-bound strategy, i.e. creating duplicates $\mathcal{P}_1$ and $\mathcal{P}_2$ of the product graph $\mathcal{P}$, in which different vertices of one component have been removed, see Fig. 4. The optimal cycle ratio is the minimum of the ratios found in $\mathcal{P}_1$ and $\mathcal{P}_2$. If $\mathcal{P}_1$ or $\mathcal{P}_2$ again yield cycles which go through components multiple times, the branch-and-bound strategy continues.

3.3. Higher-Order Ratios for Shape Matching

Recently it was shown that using higher-order edge costs (which we denote as $\bar{C}(e)$) is beneficial for shape matching problems, as this allows to incorporate geometric regularisation to avoid zig-zagging and better constrain the matching [61]. In a similar spirit, in the following we introduce higher-order ratio cycles. This involves computing the so-called conjugate graph of $\mathcal{P}$ (also known as the line-graph).

Definition 5 (Conjugate Graph). *Let $\bar{\mathcal{G}} = (\mathcal{V}_{\bar{\mathcal{G}}}, \mathcal{E}_{\bar{\mathcal{G}}})$ denote the conjugate graph of $\mathcal{G} = (\mathcal{V}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}})$. The vertices of the conjugate graph $\mathcal{V}_{\bar{\mathcal{G}}}$ are the edges $\mathcal{E}_{\mathcal{G}}$ of $\mathcal{G}$, i.e. $\bar{\mathcal{V}} = \mathcal{E}_{\mathcal{G}}$. The edges of the conjugate graph $\bar{\mathcal{E}} = \{\big((v_1, v_2), (v_3, v_4)\big) \in \mathcal{V}_{\mathcal{G}} \times \mathcal{V}_{\mathcal{G}}\}$ result from the adjacency of edges in $\mathcal{E}_{\mathcal{G}}$, i.e. whenever two (directed) edges of $\mathcal{G}$ are an incoming and an outgoing edge of the same vertex, then the respective vertices in $\bar{\mathcal{G}}$ are connected with a directed edge (going from the vertex representing the incoming edge to the vertex representing the outgoing edge).*

We now consider the conjugate product graph $\bar{\mathcal{P}}$, i.e. the conjugate graph of the product graph $\mathcal{P}$. Since vertices in $\bar{\mathcal{P}}$ are the edges of $\mathcal{P}$, the edges of $\bar{\mathcal{P}}$ have scope of *two* edges of $\mathcal{P}$. This enables the definition of higher-order edge costs $\bar{C}(e)$ and times $\bar{T}(e)$, i.e. costs (and times, respectively) which involve multiple edges of $\mathcal{P}$.

To this end, we follow Roetzer *et al.* [61], who introduced a local-rigidity regulariser which exploits higher-order costs, see also Sec. A in the supplementary. A local 3D coordinate frame on every edge of $\mathcal{C}$ and $\mathcal{Y}$ can be defined by using the edge direction, the edge-normal as well as their cross-product. From that, a rotation matrix for each edge in $\mathcal{P}$ is computed. To favour local rigidity, we consider edges of $\bar{\mathcal{P}}$, which have the scope of two neighbouring edges of $\mathcal{P}$. With that, we can use the geodesic distance between the two aforementioned rotation matrices on the Lie group $\mathrm{SO}(3)$ to quantify the local rigidity of a matching. Intuitively, this can be interpreted as penalising rotation differences of consecutive matched edges.

We use the $\mathrm{SO}(3)$-distance in addition with a data term in our higher-order edge cost $\bar{C}$. For the data term, any problem-specific feature difference can be considered (for 2D shape matching we follow [61] and use local-thickness, while for 3D shape matching we use deep features).

4. Experiments

In this section, we compare our method to various other methods on four different datasets. We specifically consider 2D-3D shape matching, 3D shape matching, and an application of our method to create GPS-art on road networks.

Implementation Details. We implement our cost-time ratio algorithm (as well as our re-implementations of [41, 60], see also Sec. C) to run on graphics cards using the

CUDA programming language. The algorithm can either be directly called via C++ or through convenient Python or Matlab wrappers. Our implementations are independent of the choice of problem (so that they can easily be re-used for general graphs occurring in other problems), they automatically ensure maximal utilisation of processing cores of the graphics hardware, are memory-efficient, and they handle distortion bounds (for any suitable input graph).

Experimental Setup. For 2D-3D shape matching we set the distortion-bound $k = 2$ and use the vanilla higher-order cost function reported in [61] with $\bar{T}(e) = 1$ for every edge e, i.e. we essentially compute higher-order mean cycles. For 3D shape matching, we set $k = 2$ and we also use $\bar{T}(e) = 1$ for every edge. Furthermore, we use the local-rigidity regulariser [61] with robust loss function [4] using parameters $\alpha = -0.4$ and $c = 0.4$. For GPS-art we set $k = 10$ and use $T(e) = 1$. For computation time comparisons we run our algorithm on a single NVIDIA L40 graphics card and the CPU-based algorithms on a 12-th Gen Core i9-12900K.

Metrics. For comparing shape matching performance we use geodesic errors, which allow to measure the correctness of the resulting matching [41, 59]. We evaluate geodesic distance from matched vertices to ground-truth vertex annotations. For TOSCA$^{2D}_{3D}$, where no such annotations are available, we evaluate the distance from matched vertices to their ground-truth segmentation [61].

4.1. 2D-3D Shape Matching

Methods. Non-rigid 2D-3D shape matching is underexplored and thus there only exists two competing methods: Lähner *et al*. [41] introduce the product graph $\mathcal{P}$ for 2D-3D shape matching and employ a single-core heap-based Dijkstra variant to compute the minimum cost cycle. Similar to [61], we evaluate Lähner *et al*. [41] *without* the segmentation pre-matching for a fair comparison to other methods. Roetzer *et al*. [61] extend the idea of Lähner *et al*. to conjugate product graphs and hence operate on a much larger graph than Lähner *et al*. (with additional runtime costs). This allows them to introduce higher-order costs. Similarly, they find the minimum cost cycle with a single-core heap-based Dijkstra variant.

Datasets. We compare 2D-3D shape matching on two different datasets: FAUST$^{2D}_{3D}$[41]: comprising of 100 3D shapes of ten different humans, where each human comes with one 2D shape. The data has ground-truth correspondences as well as segmentation correspondences of segments. TOSCA$^{2D}_{3D}$[41]: 80 3D shapes of different classes: cat, dog, horse, wolf, humans and centaur. Each class comes with at least one 2D shape. The data has only correspondences of segments.

Results. In Fig. 5 we show quantitive matching results on FAUST$^{2D}_{3D}$ and TOSCA$^{2D}_{3D}$ using geodesic distances to ground-truth correspondences and ground-truth segments,

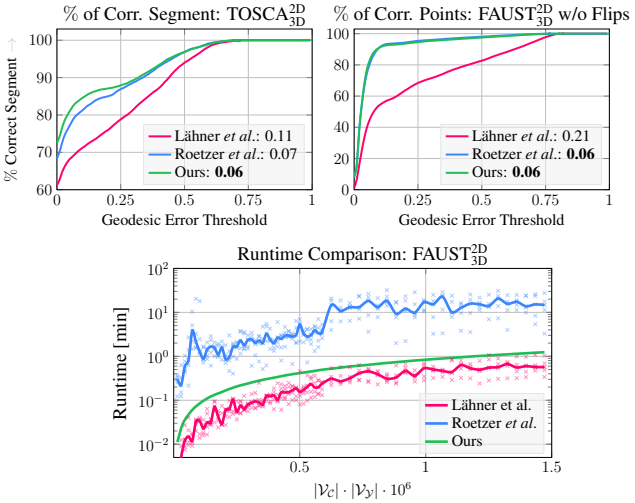


Figure 5. **Top:** Quantitive results for 2D-3D shape matching. The numbers in the legends are mean geodesic errors. **Bottom:** Computation times (including graph and cost computation) for fixed resolution of 3D shapes ($|V_\mathcal{Y}| = 3500$), while varying the resolution of the 2D contour. Lines are median computation times. Our method almost never needs to employ the branch and bound scheme (cf. Sec. 3) and thus runtimes do not fluctuate. We emphasise that only Lähner *et al*. runs on $\mathcal{P}$, while Roetzer *et al*. and ours run on the larger conjugate product graph $\bar{\mathcal{P}}$ (which is approx. $10\times$ as large as $\mathcal{P}$).

respectively, as well as runtimes of respective methods. For more results we refer to Sec. C in the supplementary. Note that, similar to previous works [61], for all methods we remove left-right flips from the evaluation with ground-truth correspondences, as these would lead to high geodesic errors but still represent valid matchings.

We can see that we perform equally well or better compared to Roetzer *et al*. [61] while having orders of magnitude faster computation times. Furthermore, in Fig. 6 and Fig. 7 we show qualitative results of respective methods. It becomes clear that optimising cost-time ratio cycles removes the bias towards shorter paths and in turn leads to better matchings.

4.2. 3D Shape Matching

We now evaluate our method on 3D shape matching. Since we require a cyclic chain graph as an input, we follow [59] to discretise one 3D shape with a so-called SpiderCurve (a cyclic chain graph embedded in 3D space visiting every vertex of a 3D shape), which we use as input $\mathcal{C}$ to our method.

Methods. We consider the following competitors: ULRSSM [12] is a deep learning method which couples functional maps with point maps to learn shape correspondences in unsupervised fashion. DiscrOpt [57] is an axiomatic method which uses the coupling between functional maps and point maps as a hard con-

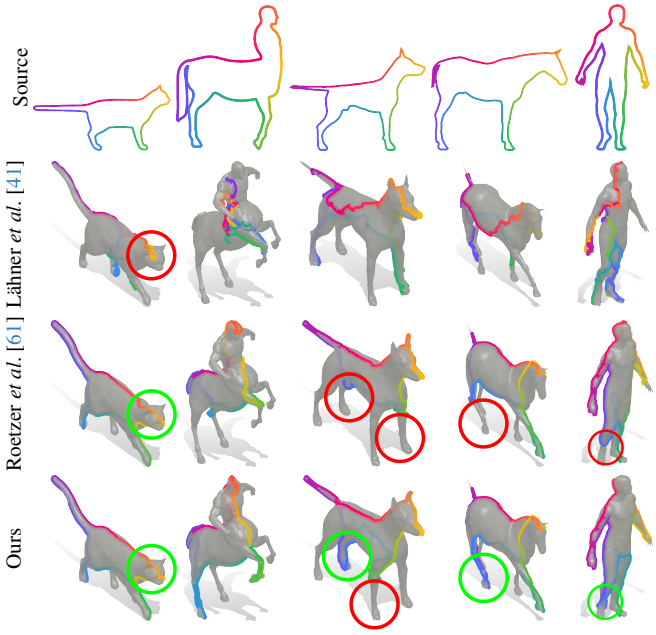


Figure 6. **Qualitative comparison** of 2D-3D shape matching on the TOSCA$^{2D}_{3D}$ dataset. Our method computes consistently better results than Roetzer *et al*. while avoiding bias towards shorter paths (which is especially visible at the legs of animals and humans). Only on the leg of dog our method fails.

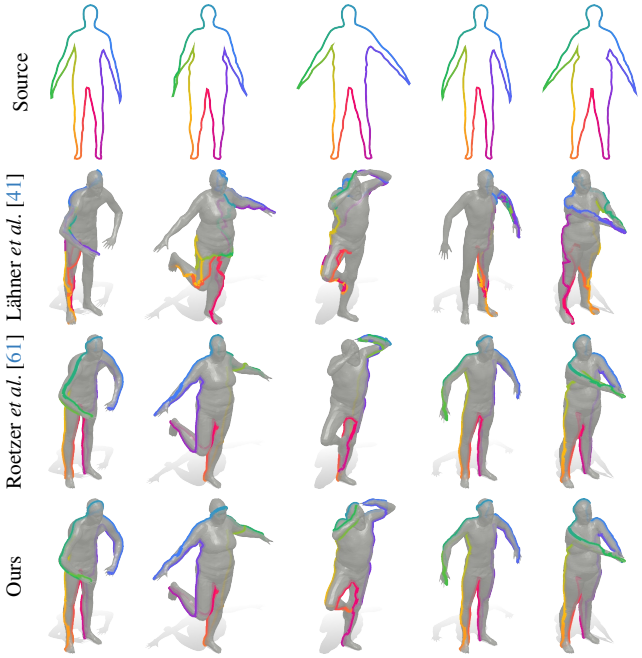


Figure 7. **Qualitative comparison** of 2D-3D shape matching on FAUST$^{2D}_{3D}$. On this dataset our method and Roetzer *et al*. produce similar results, likely because arms and legs are thicker compared to the animal shapes of TOSCA$^{2D}_{3D}$ and shortcuts (which the minimum cost cycle has a bias towards) are more costly.

straint. SmoothShells [22] is an functional map-based iterative alignment scheme which incrementally adds high-frequency details of shapes to compute the matching. SpiderMatch [59] builds upon the product graph $\mathcal{P}$, but incorporates additional structural constraints to ensure some form of 3D geometric consistency. Both, SpiderMatch [59] and ours use the same feature extractor to define costs [12].

Datasets. We compare 3D shape matching on two different datasets: FAUST [9, 18, 56] consists of 100 near-isometric non-rigidly deformed human shapes. We evaluate on the 100 randomly sampled pairs (from the test set) reported in [59]. DT4D [48] is based upon [46] and consists of 9 different classes of human-like shapes in different non-rigidly deformed poses. We use the 100 randomly (from the test set) sampled intra-class pairs reported in [59].

Results. In Fig. 8 we show quantitative results. W.r.t. shape matching performance, our method performs equally well as the SOTA method SpiderMatch [59], despite not employing geometric consistency constraints. Furthermore, our method is faster compared to SpiderMatch [59], despite operating on a much larger graph ($\bar{\mathcal{P}}$ is approx. $10\times$ larger compared to $\mathcal{P}$). For a detailed comparison of graph sizes of $\mathcal{P}$ and $\bar{\mathcal{P}}$, we refer to [61].

In Fig. 9, we showcase qualitative results of our method, SpiderMatch [59], and ULRSSM [12] (comparison with all other methods as well as more qualitative results can be found in Sec. D in the supplementary). We can see that SpiderMatch [59] and ours produce very similar results, which is also confirmed through quantitative comparison in Fig. 8.

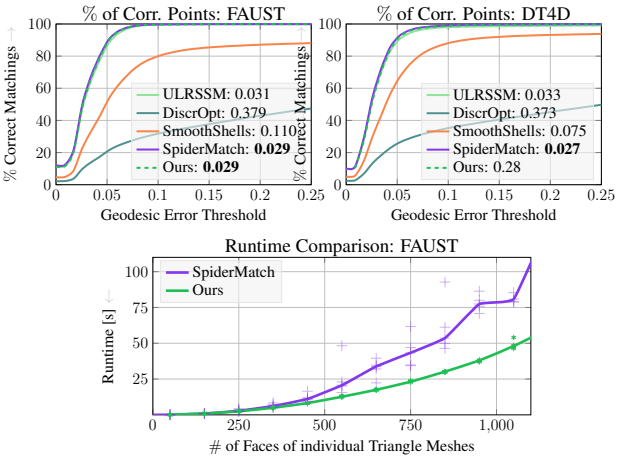


Figure 8. **Top:** Quantitative results on 3D shape matching. Numbers in legends are mean geodesic errors. Our approach achieves similar matching accuracy as SpiderMatch [59], even though we do not enforce geometric consistency constraints. **Bottom:** We compare computation times of our approach with the other path-based shape matching approach SpiderMatch [59]. Lines are median runtimes. We can see that our approach is faster, even though we operate on the much larger graph $\bar{\mathcal{P}}$.

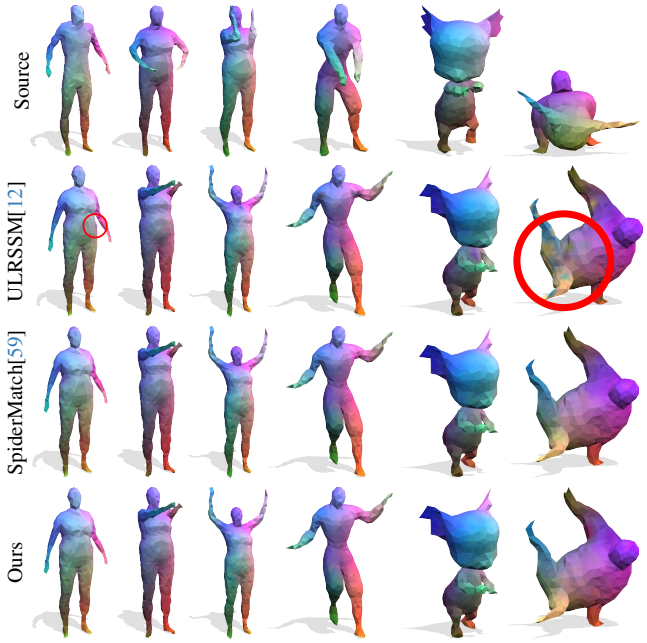


Figure 9. **Qualitative results on 3D shape matching** visualised by transferring colour from the source to the target shape in each column. Our method performs visually similarly to SpiderMatch despite not employing geometric consistency constraints. We believe that regularisation with local rigidity and the fact that subsequent edges of the matching are connected (which could be interpreted as weak form of neighbourhood preservation) leads to such good matching quality.

4.3. GPS-Art on City Maps

As a proof of concept, we demonstrate that our formalism can also be used for matching a 2D shape to a graph. To this end, we match the outline of 2D objects to a road network of Nashville [8], which is a task that occurs in so-called *GPS-art*. For qualitative results, see Fig. 1, right and Fig. 10.

To compute the track, we compute the product graph between the 2D outline and the road network. As cost, we use the xy-coordinate difference between vertices of the 2D object outline and the vertex locations of the graph of the road network. As time, we use the trivial case of $T(e) = 1$. Consequently, the computed track shown in Fig. 1 amounts for a minimum mean cycle through the product graph.

5. Discussion

Our experimental evaluation confirms that finding minimum ratio cycles indeed avoids the bias towards shorter paths (which especially in the 2D-3D setting is undesirable due to the unknown number of matched edges). This results in better matching quality in most cases since edges with good *average* matching cost are preferred. Yet, in some rare cases, our approach yields undesirable results, e.g. the leg of the dog in Fig. 6. We believe that more carefully curated matching costs may circumvent such issues.

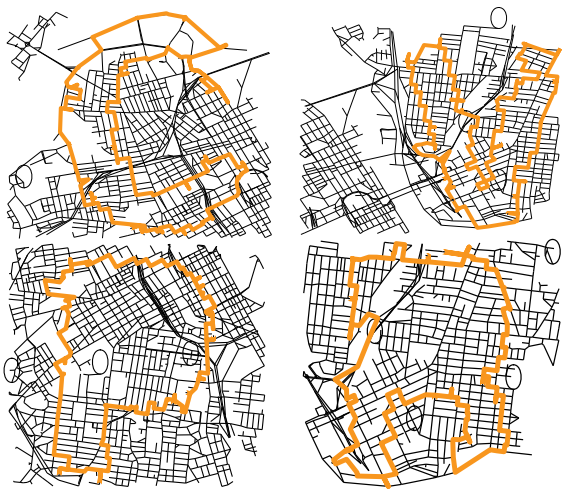


Figure 10. **GPS-art** examples that find tracks on different parts of the road-network of Nashville resembling the letters C-V-P-R.

We demonstrate that runtimes of our approach are significantly lower compared to the other path-based algorithm that operates on a conjugate product graph [61]. This runtime decrease mainly stems from a combination of an efficient GPU implementation that is enabled by distortion bound k, and the fact that in practice far less branch and bound iterations compared to [41, 61] are employed.

In our work, we specifically consider matching problems that can strictly be modelled using a product graph of two input graphs. In principle, the overall framework can be applied to more general forms of component-based graphs (e.g. a product graph that is constructed from dynamic graphs). We leave an investigation of such cases for future work.

6. Conclusion

We present an alternative to the prevalent minimum cost cycle formulation for shape matching problems on product graphs. To this end, we consider the minimum ratio cycle formalism that allows for a higher modelling flexibility and that empirically leads to favourable results, both in terms of accuracy and runtime. We believe that improved runtimes are an important step towards integrating such principled matching approaches into end-to-end learning-based pipelines. Further, the ratio-based formalism effectively avoids a bias towards solutions with fewer/shorter edges. Overall we believe that our work is not only an important contribution to the field of shape analysis, but that it may also be beneficial for more general visual computing downstream tasks. To foster follow-up works, we make our efficient GPU implementation publicly available.

Acknowledgments

We thank Ahmed Abbas for valuable advice on CUDA implementations. Zorah Lähner acknowledges the support of the DFG Sachbeihilfe grant LA 5191/2-1. This work was supported by the ERC Starting Grant "Harmony" (#101160648) and by the ERC Advanced Grant "SIMULACRON" (agreement #884679).

References

[1] Ahmed Abbas and Paul Swoboda. Fastdog: Fast discrete optimization on gpu. In *CVPR*, 2022. 2

[2] Jason M Altschuler and Pablo A Parrilo. Approximating min-mean-cycle for low-diameter graphs in near-optimal time and memory. *SIAM Journal on Optimization*, 2022. 3

[3] Marco Attene. A lightweight approach to repairing digitized polygon meshes. *The visual computer*, 2010. 3

[4] Jonathan T Barron. A general and adaptive robust loss function. In *CVPR*, 2019. 6

[5] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 1958. 4, 2

[6] Florian Bernard, Christian Theobalt, and Michael Moeller. Ds*: Tighter lifting-free convex relaxations for quadratic matching problems. In *CVPR*, 2018. 2

[7] Florian Bernard, Zeeshan Khan Suri, and Christian Theobalt. Mina: Convex mixed-integer programming for non-rigid shape alignment. In *CVPR*, 2020. 2

[8] Geoff Boeing. Modeling and analyzing urban networks and amenities with osmnx. 2024. 8

[9] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *CVPR*, 2014. 7

[10] Karl Bringmann, Thomas Dueholm Hansen, and Sebastian Krinninger. Improved Algorithms for Computing the Cycle of Minimum Cost-to-Time Ratio in Directed Graphs. In *ICALP*, 2017. 3

[11] Dongliang Cao and Florian Bernard. Unsupervised deep multi-shape matching. In *ECCV*, 2022. 1

[12] Dongliang Cao, Paul Roetzer, and Florian Bernard. Unsupervised learning of robust spectral shape matching. *ACM Transactions on Graphics (TOG)*, 2023. 1, 6, 7, 8, 4

[13] Andrew Clements and Hao Zhang. Minimum ratio contours on surface meshes. In *SMI*, 2006. 2, 3

[14] Luca Cosmo, Emanuele Rodola, Andrea Albarelli, Facundo Mémoli, and Daniel Cremers. Consistent partial matching of shape collections via sparse modeling. In *Computer Graphics Forum*, 2017. 2

[15] RA Cuninghame-Green and Yixun Lin. Maximum cycle-means of weighted digraphs. *Applied Mathematics*, 1996. 3

[16] Ali Dasdan. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2004. 3, 4

[17] Ali Dasdan and Rajesh K Gupta. Faster maximum and minimum mean cycle algorithms for system-performance analysis. *IEEE transactions on computer-aided design of integrated circuits and systems*, 1998. 3

[18] Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. Deep geometric functional maps: Robust feature learning for shape correspondence. In *CVPR*, 2020. 7

[19] Nadav Dym, Haggai Maron, and Yaron Lipman. DS++ - A Flexible, Scalable and Provably Tight Relaxation for Matching Problems. *ACM Transactions on Graphics (TOG)*, 36(6), 2017. 2

[20] Viktoria Ehm, Maolin Gao, Paul Roetzer, Marvin Eisenberger, Daniel Cremers, and Florian Bernard. Partial-to-partial shape matching with geometric consistency. In *CVPR*, 2024. 2

[21] Viktoria Ehm, Paul Roetzer, Marvin Eisenberger, Maolin Gao, Florian Bernard, and Daniel Cremers. Geometrically consistent partial shape matching. In *3DV*, 2024. 2

[22] Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *CVPR*, 2020. 2, 7, 3, 4

[23] Barakeel Fanseu Kamhoua, Lin Zhang, Yongqiang Chen, Han Yang, MA Kaili, Bo Han, Bo Li, and James Cheng. Exact shape correspondence via 2d graph convolution. *Advances in Neural Information Processing Systems*, 2022. 1

[24] Alex Fisher, Ricardo Cannizzaro, Madeleine Cochrane, Chatura Nagahawatte, and Jennifer L Palmer. Colmap: A memory-efficient occupancy grid mapping framework. *Robotics and Autonomous Systems*, 2021. 1

[25] L. R. Ford. *Network Flow Theory.* RAND Corporation, Santa Monica, CA, 1956. 4, 2

[26] Maolin Gao, Paul Roetzer, Marvin Eisenberger, Zorah Lähner, Michael Moeller, Daniel Cremers, and Florian Bernard. Sigma: Scale-invariant global sparse shape matching. In *ICCV*, 2023. 2

[27] Andrea Gasparetto, Luca Cosmo, Emanuele Rodola, Michael Bronstein, and Andrea Torsello. Spatial maps: From low rank spectral to sparse spatial functional representations. In *3DV*. IEEE, 2017. 2

[28] Joan Glaunès, Anqi Qiu, Michael I Miller, and Laurent Younes. Large deformation diffeomorphic metric curve mapping. *International journal of computer vision*, 2008. 2

[29] Oshri Halimi, Or Litany, Emanuele Rodola, Alex M Bronstein, and Ron Kimmel. Unsupervised learning of dense shape correspondence. In *CVPR*, 2019. 1

[30] Stefan Haller, Lorenz Feineis, Lisa Hutschenreiter, Florian Bernard, Carsten Rother, Dagmar Kainmüller, Paul Swoboda, and Bogdan Savchynskyy. A comparative study of graph matching algorithms in computer vision. In *ECCV*, 2022. 2

[31] Richard H Hammack, Wilfried Imrich, Sandi Klavžar, Wilfried Imrich, and Sandi Klavžar. *Handbook of product graphs*. CRC press Boca Raton, 2011. 4

[32] Mark Hartmann and James B Orlin. Finding minimum cost to time ratio cycles with small integral transit times. *Networks*, 1993. 3

[33] Tobias Heimann and Hans-Peter Meinzer. Statistical shape models for 3d medical image segmentation: a review. *Medical image analysis*, 2009. 1

[34] R. A. Howard. Dynamic programming and markov processes. *The Mathematical Gazette*, 1962. 3, 2

[35] Ian H Jermyn and Hiroshi Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *TPAMI*, 2001. 1, 2, 3, 4

[36] Ian H Jermyn, Sebastian Kurtek, Eric Klassen, and Anuj Srivastava. Elastic shape matching of parameterized surfaces using square root normal fields. In *ECCV*. Springer, 2012. 2

[37] Richard M Karp. A characterization of the minimum cycle mean in a digraph. *Discrete mathematics*, 1978. 3

[38] Richard M Karp and James B Orlin. Parametric shortest path algorithms with an application to cyclic staffing. *Discrete Applied Mathematics*, 1981. 3

[39] K. Kolev and D. Cremers. Continuous ratio optimization via convex relaxation with applications to multiview 3d reconstruction. In *CVPR*, 2009. 3

[40] Yam Kushinsky, Haggai Maron, Nadav Dym, and Yaron Lipman. Sinkhorn Algorithm for Lifted Assignment Problems. *SIAM Journal on Imaging Sciences*, 12(2):716–735, 2019. 2

[41] Zorah Lahner, Emanuele Rodola, Frank R Schmidt, Michael M Bronstein, and Daniel Cremers. Efficient globally optimal 2d-to-3d deformable shape matching. In *CVPR*, 2016. 1, 2, 3, 4, 5, 6, 7, 8

[42] Eugene L Lawler. Optimal cycles in graphs and the minimal cost-to-time ratio problem. *Periodic Optimization: Volume I: Course Held at the Department of Automation and Information*, 1972. 3, 4

[43] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 2001. 4

[44] D. Khuê Lê-Huu and Nikos Paragios. Alternating Direction Graph Matching. In *CVPR*, 2017. 2

[45] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *ECCV*, 2024. 1

[46] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *ICCV*, 2021. 7

[47] O. Litany, T. Remez, E. Rodolà, A. M. Bronstein, and M. M. Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *ICCV*, 2017. 1

[48] Robin Magnet, Jing Ren, Olga Sorkine-Hornung, and Maks Ovsjanikov. Smooth non-rigid shape matching via effective dirichlet energy optimization. In *3DV*, 2022. 7

[49] Supriyo Maji and Cheng-Kok Koh. A scalable, memory-efficient algorithm for minimum cycle mean calculation in directed graphs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021. 3

[50] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 2016. 2

[51] Damien Michel, Iasonas Oikonomidis, and Antonis Argyros. Scale invariant and deformation tolerant partial shape matching. *Image and Vision Computing*, 2011. 2

[52] Edward F Moore. The shortest path through a maze. In *Proc. of the International Symposium on the Theory of Switching*, 1959. 4, 2

[53] Fred Nicolls and Philip HS Torr. Discrete minimum ratio curves and surfaces. In *CVPR*, 2010. 3

[54] James B Orlin and Ravindra K Ahuja. New scaling algorithms for the assignment and minimum mean cycle problems. *Mathematical programming*, 1992. 3

[55] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):30, 2012. 2

[56] Jing Ren, Adrien Poulenard, Peter Wonka, and Maks Ovsjanikov. Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (ToG)*, 2018. 7

[57] Jing Ren, Simone Melzi, Peter Wonka, and Maks Ovsjanikov. Discrete optimization for shape matching. In *Computer Graphics Forum*, pages 81–96. Wiley Online Library, 2021. 2, 6, 3, 4

[58] F Rendl, P Pardalos, and H Wolkowicz. The Quadratic Assignment Problem: A Survey and Recent Developments. In *DIMACS workshop*, 1994. 2

[59] Paul Roetzer and Florian Bernard. Spidermatch: 3d shape matching with global optimality and geometric consistency. In *CVPR*, 2024. 1, 2, 3, 4, 6, 7, 8

[60] Paul Roetzer, Paul Swoboda, Daniel Cremers, and Florian Bernard. A scalable combinatorial solver for elastic geometrically consistent 3d shape matching. In *CVPR*, 2022. 2, 5

[61] Paul Roetzer, Zorah Lähner, and Florian Bernard. Conjugate product graphs for globally optimal 2d-3d shape matching. In *CVPR*, 2023. 1, 2, 3, 4, 5, 6, 7, 8

[62] Paul Roetzer, Ahmed Abbas, Dongliang Cao, Florian Bernard, and Paul Swoboda. Discomatch: Fast discrete optimisation for geometrically consistent 3d shape matching. In *ECCV*, 2024. 2

[63] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 1978. 2

[64] Christian Schellewald and Christoph Schnörr. Probabilistic Subgraph Matching Based on Convex Relaxation. In *EMMCVPR*, 2005. 2

[65] Frank R Schmidt, Dirk Farin, and Daniel Cremers. Fast matching of planar shapes in sub-cubic runtime. In *ICCV*, 2007. 2

[66] Thomas Schoenemann and Daniel Cremers. Globally optimal image segmentation with an elastic shape prior. In *ICCV*, 2007. 1, 2, 3, 4, 5

[67] Thomas Schoenemann and Daniel Cremers. Introducing curvature into globally optimal image segmentation: Minimum ratio cycles on product graphs. In *ICCV*, 2007. 3

[68] Thomas Schoenemann and Daniel Cremers. A combinatorial solution for model-based image segmentation and real-time tracking. *TPAMI*, 2009. 1, 2, 3, 4, 5

[69] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Trans. Graph.*, 2022. 1

[70] Matthias Vestner, Zorah Lähner, Amit Boyarski, Or Litany, Ron Slossberg, Tal Remez, Emanuele Rodola, Alex Bronstein, Michael Bronstein, Ron Kimmel, and Daniel Cre-

mers. Efficient Deformable Shape Correspondence via Kernel Matching. In *3DV*, 2017. 2

[71] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 1

[72] Thomas Windheuser, Ulrich Schlickewei, Frank R. Schmidt, and Daniel Cremers. Geometrically consistent elastic matching of 3D shapes: A linear programming solution. In *ICCV*, 2011. 2

[73] Thomas Windheuser, Ulrich Schlickwei, Frank R. Schimdt, and Daniel Cremers. Large-Scale Integer Linear Programming for Orientation Preserving 3D Shape Matching. *Computer Graphics Forum*, 2011. 2

[74] Neal E Young, Robert E Tarjant, and James B Orlin. Faster parametric shortest path and minimum-balance algorithms. *Networks*, 1991. 3