

DiscoMatch: Fast Discrete Optimisation for Geometrically Consistent 3D Shape Matching

Paul Roetzer¹, Ahmed Abbas², Dongliang Cao¹, Florian Bernard¹, and Paul Swoboda^{2,3}

¹ University of Bonn

² Max Planck Institute for Informatics, Saarland Informatics Campus

³ Heinrich-Heine University Düsseldorf

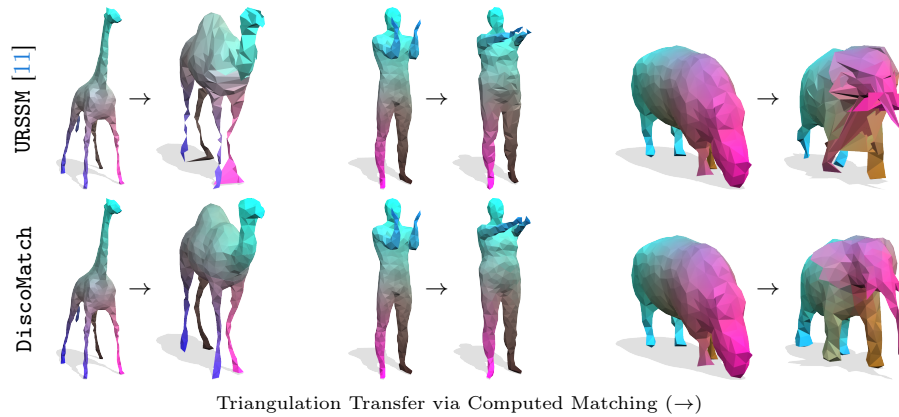


Fig. 1: We efficiently solve a large integer linear program (ILP) to guarantee geometric consistency of matchings between pairs of 3D shapes. Our matchings are smooth and more accurate than the recent state-of-the-art shape matching approach URSSM [11]. Improved solution quality can be seen when transferring triangulation between shapes.

Abstract. In this work we propose to combine the advantages of learning-based and combinatorial formalisms for 3D shape matching. While learning-based methods lead to state-of-the-art matching performance, they do not ensure geometric consistency, so that obtained matchings are locally non-smooth. On the contrary, axiomatic, optimisation-based methods allow to take geometric consistency into account by explicitly constraining the space of valid matchings. However, existing axiomatic formalisms do not scale to practically relevant problem sizes, and require user input for the initialisation of non-convex optimisation problems. We work towards closing this gap by proposing a novel combinatorial solver that combines a unique set of favourable properties: our approach (i) is initialisation free, (ii) is massively parallelisable and powered by a quasi-Newton method, (iii) provides optimality gaps, and (iv) delivers improved match-

ing quality with decreased runtime and globally optimal results for many instances. Our code is publicly available⁴.

Keywords: Shape matching · Discrete optimisation

1 Introduction

The shape matching problem, i.e., finding correspondences between two non-rigidly deformed 3D shapes, lies at the heart of many visual computing tasks. Resulting correspondences are relevant for many downstream applications in computer vision (3D reconstruction, shape space modelling), graphics (texture transfer, shape interpolation) or medical image analysis (shape-based segmentation, navigation). Thus, shape matching has received a lot of attention in recent years and has been tackled by numerous variants of both axiomatic and learning-based methods.

Learning-based methods usually provide the best matching performance on the most challenging benchmarks [11]. However, a major downside is that they do not provide any guarantees about crucial properties of the final correspondences, such as *geometric consistency* of matchings: for triangular surface meshes, geometric consistency means that neighbouring triangles of one shape are consistently matched to neighbouring triangles of the other shape (analogous to diffeomorphisms in the continuous domain), cf. Fig. 2 left.

While axiomatic methods can – in principle – take geometric consistency into account, most existing shape matching approaches disregard it [5, 24, 41, 47, 58, 61, 64], mainly due to the resulting non-convexity of the optimisation problem or hard-to-solve constraints. Although there are some exceptions, respective formulations can typically be solved only for small instances, such as [68], or they rely on manual user initialisations to avoid poor local optima and to better constrain the problem [52, 54, 60].

We argue that geometric consistency is of crucial importance and it should be considered as an essential aspect for any shape matching pipeline. Hence, in this work we propose to combine expressive and powerful shape features obtained from a learning-based method with a novel combinatorial solver for the geometrically consistent shape matching formalism [68]. We summarise our main contributions as follows:

- a. We propose a theoretically well-grounded solver for global shape matching that is able to quantify optimality gaps. We are empirically able to certify global optimality in 90% of the considered shape matching instances.
- b. Our solver extends the dual decomposition approach [1] and also addresses specific challenges posed by large shape matching instances. By doing so we demonstrate a 9 to 11 times faster convergence compared to [1] on shape matching problems. Additionally some components of our solver are generally applicable and yield 5 to 10 times runtime improvements over [1].

⁴ <https://github.com/paul0noah/disco-match>

- c. We outperform the current state-of-the-art learning-based shape matching approach on most considered datasets w.r.t. matching accuracy and smoothness, while at the same time guaranteeing geometric consistency.

2 Related Work

In the following we summarise works that are most relevant to our approach. For an extensive review of shape matching methods we refer to the surveys [48, 64].

Deep Shape Matching. With the rapid development of deep learning, many learning-based approaches are proposed for 3D shape matching, which can be categorised into supervised and unsupervised methods. In the context of supervised learning, some methods [32, 67] formulate shape matching as a classification problem, while others [20, 63] solve the problem based on non-rigid shape alignment. Nevertheless, supervised methods require ground-truth correspondences, which in practice are hard to obtain. Therefore, many unsupervised methods were introduced to get rid of the requirement of ground-truth correspondences. Notably, the functional map framework [41] is one of the most dominant ingredients to enable unsupervised learning.

Earlier works [22, 46] impose isometry regularisation in the spatial and spectral domain, respectively. Follow-up works [14, 55] use point-based networks [42, 62] and lead to better matching performance. Later, learnable implicit diffusion processes pushed the state of the art in shape matching for a broad range settings [57], including near-isometry [2], non-isometry [10, 13, 31], partiality [3], multi-shape matching [8] as well as multi-modal matching [9]. Meanwhile, instead of solely learning features, several works [26, 37] also attempted to learn the basis functions at the same time.

One important ingredient of our work is the utilisation of recent learned feature embeddings. While existing shape matching approaches that are purely learning-based do not provide any guarantees about geometric consistency (i.e., the smoothness of maps), we propose to combine learned embeddings with an axiomatic optimisation approach in order to guarantee geometric consistency via hard constraints.

Geometric Consistency. Several 3D shape matching methods provide geometrically consistent matchings by performing local optimisation on initial maps between two shapes. Continuous maps starting from user-defined correspondences are computed in [54]. Refinement of an initial matching in a geometrically consistent manner using heat diffusion in [56]. 3D shape matching is casted as image matching problem to obtain smooth maps via a smooth surface-to-image parameterisation in [34]. In [66] the so-called product manifold filter is used to find smooth maps via a linearisation of the quadratic assignment problem. Local map refinement of initial maps is done in [16] via denoising, or in [17] via minimising an elastic energy. Common triangulations between two shapes have been utilised in [51] or via intrinsic triangulation in [60]. Mapping of shapes to spheres and then optimising for a homeomorphism is done in [52].

While all of the above methods provide high quality and smooth maps they rely on initial correspondences which are non-trivial to find. Thus these methods can only be considered as *local* shape matching methods. Our approach allows to infer smooth maps (i.e., a discrete analogue to a diffeomorphism) between 3D shapes based on an efficient relaxation solver for integer programs.

Algorithms for Global Shape Matching. Practically relevant and globally optimal 3D shape matching is considered extremely challenging. Nevertheless, in certain cases respective problems are solvable efficiently: for example 2D to 2D matching via dynamic programming [12] analogous to dynamic time warping [49], or via graph cuts [50]; another example is contour-to-image matching via dynamic programming [18] or shortest paths [53]. Matching a 2D contour to a 3D mesh can also be phrased as shortest path problem [29, 44].

For the 3D to 3D matching scenario there exist several sparse matching methods [5, 19, 38], or the popular functional map [41] approach that operates in the spectral domain. Relaxations to bi-stochastic matrices for the QAP are used in [65] to map nearby points on one shape to nearby points on the other shape. Furthermore, an optimisation problem to find a dense discrete diffeomorphism between two 3D shapes is introduced in [68]. While this optimisation problem purely relies on linear constraints to ensure geometric consistency, it is nevertheless hard to solve since it optimises over a large number of binary variables. Thus, various strategies to address this problem have been proposed: a linear programming relaxation [68] combined with a coarse-to-fine scheme [69], as well as an approach using a combinatorial solver with a problem-specific primal heuristic [45]. Still, the mentioned methods do not scale to practically relevant resolutions and oftentimes cannot find good lower-bounds (which as a consequence results in bad primal solutions in many cases). We show that with our combinatorial optimiser we can find solutions much faster and often times find a globally optimal solution due to faster and better lower bound computation.

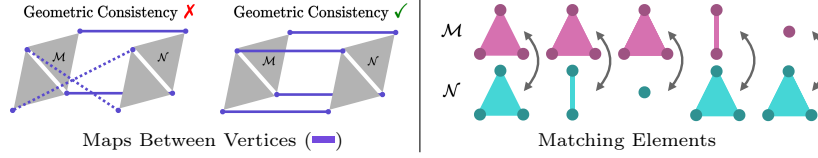
3 Background

We first introduce the integer linear programming (ILP) formulation for shape matching and further touch upon a method for solving large scale ILPs which we build on. Our notation is summarised in Tab. 1.

3.1 3D Shape Matching as ILP

The matching formalism in [68] aims to find a *discrete diffeomorphism* between two triangle meshes \mathcal{M} and \mathcal{N} by solving a constrained binary integer linear program. The matching is found within the search space P (i.e. the product space) which consists of three types of potential correspondences between \mathcal{M} and \mathcal{N} : triangle-triangle, triangle-edge and triangle-vertex, see Fig. 2 right. The latter two are needed to account for different triangulation as well as stretching and compression of shapes. Employing geometric consistency constraints $A^\partial x = \mathbf{0}$

Symbol	Description	Symbol	Description
$\mathcal{M} = (V_{\mathcal{M}}, T_{\mathcal{M}})$	Triangle mesh	$x \in \{0, 1\}^{ P }$	Indicator representation of P
$V_{\mathcal{M}} \in \mathbb{R}^{ V_{\mathcal{M}} \times 3}$	Vertices of \mathcal{M}	$c \in \mathbb{R}_+^{ P }$	Cost vector
$T_{\mathcal{M}} \in \mathbb{N}^{ T_{\mathcal{M}} \times 3}$	Triangles of \mathcal{M}	$A^{\partial}x = \mathbf{0}$	Geo. consistency constraints
$F_{\mathcal{M}} \in \mathbb{R}^{ V_{\mathcal{M}} \times 128}$	Features of \mathcal{M}	$A^{\mathcal{M}}x = \mathbf{1}$	Projection on \mathcal{M} constraints
$\mathcal{N} = (V_{\mathcal{N}}, T_{\mathcal{N}})$	Triangle mesh	$A^{\mathcal{N}}x = \mathbf{1}$	Projection on \mathcal{N} constraints
...	(analog. to \mathcal{M})	\mathcal{I}, \mathcal{J}	Set of vars., constraints resp.
P	Product space	$\mathcal{I}_j, (\mathcal{J}_i)$	Vars. of j in \mathcal{J} (viceversa)

Table 1: Summary of the notation used in this paper.**Fig. 2:** *Left:* Geometric consistency is an essential property of correspondences between shapes, meaning that neighbouring elements in \mathcal{M} are consistently matched to neighbouring elements in \mathcal{N} . *Right:* Matching elements of the approach by [68]: triangle-triangle, triangle-edge and triangle-vertex between shapes \mathcal{M} and \mathcal{N} . At least one triangle is involved in every matching element.

ensures that neighbouring elements in \mathcal{M} are matched only to neighbouring elements in \mathcal{N} , see Fig. 2 left. Additional projection constraints $A^{\mathcal{M}}x = \mathbf{1}$ and $A^{\mathcal{N}}x = \mathbf{1}$ ensure that each triangle in $T_{\mathcal{M}}$ and $T_{\mathcal{N}}$ respectively are matched exactly once and thus ensure surjectivity. A solution x fulfilling all constraints yields a discrete diffeomorphism between shapes \mathcal{M} and \mathcal{N} [68]. The full optimisation problem reads

$$\min_{x \in \{0,1\}^{|P|}} c^{\top} x \quad \text{s.t.} \quad \underbrace{\begin{bmatrix} A^{\partial} \\ A^{\mathcal{M}} \\ A^{\mathcal{N}} \end{bmatrix}}_{:=A} x = \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}}_{:=b}. \quad (\text{P})$$

For a detailed derivation of (P) and more insights we refer readers to [45, 68, 69]. As mentioned, the solution of this optimisation problem yields a discrete diffeomorphism between two triangle meshes. This ensures geometric consistency. Since the problem is a difficult and large combinatorial optimisation problem, only heuristic solvers (cf. [45, 69]) exist to solve the problem for reasonable shape resolutions. We show that by combining deep feature embeddings and adapting a novel combinatorial solver, we can solve the problem significantly faster than before, while at the same time certifying global optimality in most cases.

3.2 Dual Decomposition for ILPs

In order to solve large shape matching integer linear programs (ILPs) (P), we take [1] as a starting point. Its massive parallelism and generic applicability make it a suitable choice. We give an overview of their approach below.

Since ILPs are NP-hard to solve, a relaxation based on Lagrangian dual is proposed in [1]. Formally, denote \mathcal{I} and \mathcal{J} as the set of variables and constraints, respectively, of the ILP (P). Moreover, denote the variables in constraint j as \mathcal{I}_j , and rows containing variable i as \mathcal{J}_i . Then represent each constraint j as an independent subproblem $\mathcal{S}_j := \{s \in \{0, 1\}^{\mathcal{I}_j} : \sum_{i \in \mathcal{I}_j} a_{ji}x_i = b_j\}$. The relaxed (dual) problem is written as

$$\max_{\lambda} \sum_{j \in \mathcal{J}} \min_{s \in \mathcal{S}_j} s^\top \lambda^j \quad \text{s.t.} \quad \sum_{j \in \mathcal{J}_i} \lambda_i^j = c_i \quad \forall i \in \mathcal{I}, \quad (\text{D})$$

where λ_i^j is the dual variable for primal variable i and row (constraint) j and λ^j is the vector containing all variables of j , i.e., $\lambda^j = (\lambda_i^j)_{i \in \mathcal{I}_j}$. If optima of individual subproblems \mathcal{S}_j agree, then the dual problem solves the original problem (P). In general it provides a lower bound to the original problem. For more details we refer to [1, 30].

For optimising the dual problem a block coordinate ascent algorithm based on parallel min-marginal averaging is utilized in [1]. Their update scheme further guarantees that the dual objective improves or stays the same. Lastly, the scheme of [1] to find a feasible solution (i.e., a matching) to (P) critically relies on optimising the dual problem. An improved scheme for solving the dual therefore not only helps in finding better lower bounds but also in recovering better solutions of the original problem.

4 Method

We now present our shape matching framework. First, we will discuss improvements to [1] for efficiently solving the ILP (P). Next we propose our improved cost function which helps to obtain more accurate results by injecting more information into the optimisation problem. The dataflow of our proposed matching pipeline is shown in Fig. 3.

4.1 Faster Optimisation

Although the method of [1] delivers good performance on large scale ILPs, it exhibits two weaknesses. First, it can be quite slow to converge as it only relies on first-order updates for optimising the dual (D) and indirectly also the associated primal (P). We aim for better optimisation of the dual problem by leveraging second-order information. Second, the equalities $A^{\mathcal{M}}x = \mathbf{1}$ and $A^{\mathcal{N}}x = \mathbf{1}$ involve a large number of variables (cf. Fig. A.6), thus creating large subproblems, each of which must be processed sequentially, causing bottlenecks when computing

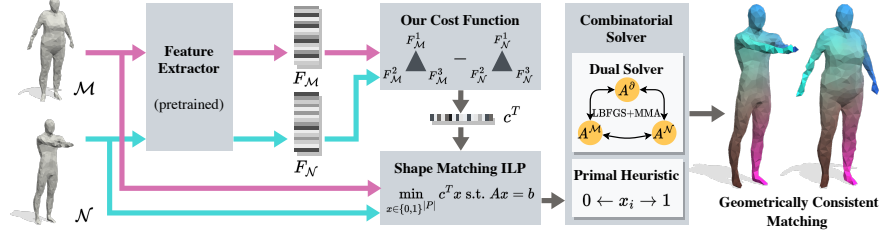


Fig. 3: Illustration of our shape matching pipeline. We use a pretrained feature extractor to define the cost function of ILP (P), which we solve using our combinatorial solver to find a geometrically consistent matching between shapes. The dual solver performs second-order quasi-Newton updates on top of the first-order min-marginal averaging scheme (MMA) of [1]. For primal recovery we adapt the approach of [1] by replacing their dual optimisation procedure by ours leading to faster and better-quality solutions.

on GPUs. To tackle this issue we propose a technique which splits a subproblem into multiple smaller ones that can be solved in parallel.

Quasi-Newton Updates. We incorporate second-order information in optimising the dual problem (D) through quasi-Newton updates. To this end we interleave L-BFGS [35, 40] updates with the first-order min-marginal averaging (MMA) scheme of [1]. An iteration of our scheme is given in Algorithm 1.

Algorithm 1: Quasi-Newton powered MMA (D)

Input: Dual variables λ , Hessian inverse estimate \hat{H} , Previous step size γ

- 1 $g^j(\lambda) := \arg \min_{s \in S_j} s^\top \lambda^j, \forall j \in \mathcal{J}$ // Subgradient of dual obj. (D)
- 2 $\hat{d} = \hat{H}g(\lambda)$ // Compute update direction
- 3 $d_i^j = \hat{d}_i^j - \frac{1}{|\mathcal{J}_i|} \sum_{k \in \mathcal{J}_i} \hat{d}_i^k, \forall i \in \mathcal{I}, j \in \mathcal{J}_i$ // Ensure dual feasibility (D)
- 4 $\gamma \leftarrow \text{FindStepSize}(\lambda, d, \gamma)$
- 5 $\lambda \leftarrow \lambda + \gamma d$ // Update Dual variables
- 6 $\lambda \leftarrow \text{MMA}(\lambda)$ // First-order update of [1]
- 7 $\hat{H} \leftarrow \text{L-BFGS}(\hat{H}, g(\lambda))$ // Update inverse Hessian estimate
- 8 **return** λ, \hat{H}, γ

In detail, we first compute a subgradient of the dual objective by finding a minimising assignment for each subproblem. Next, we calculate the search direction \hat{d} by multiplying the subgradient with an estimate of the inverse Hessian. Naively updating the dual variables in this search direction however, can violate dual feasibility (D). To address this, we project \hat{d} to a feasible search direction d such that $\sum_j d_i^j$ is zero for all i . Subsequently, a suitable step size is determined that provides sufficient increase in the objective (D). Using this step size, we perform the quasi-Newton update, followed by a min-marginal averaging (MMA) update of [1]. Finally, we update the estimate of the inverse Hessian for use in

subsequent iterations. More detailed information about step size selection and inverse Hessian updates can be found in the Appendix. Note that due to its general nature our second-order update scheme is applicable to all problems which fall under the framework of [1].

In essence, our algorithm combines two distinct update schemes for optimising the dual problem. It employs a first-order update through min-marginal averaging (MMA), which is fast and guarantees non-decreasing objective in its updates. Our L-BFGS scheme leverages second-order information across MMA iterations and facilitate faster convergence towards the optimum. Empirically, we observe this hybrid scheme to be significantly faster than either L-BFGS or MMA alone.

Constraint Splitting. Similar to [1] we represent each constraint as a binary decision diagram (BDD) [7], a general purpose data structure for encoding functions of binary variables. Essentially, BDDs are directed acyclic graphs and optimal solutions can be found by computing a shortest path. For an example of a BDD see Figure A.5 in the Appendix. Shortest paths and other needed computations require sequentially going through nodes in each of the BDDs. These computations can be parallelized across BDDs on GPUs. However, when a few long BDDs are present, the full parallelism of GPUs can often not be exploited. For this reason we propose a BDD splitting technique that takes a BDD as input, together with the desired smaller chunk size, and outputs a number of sub-BDDs of chunk size length that together are equivalent to the original BDD.

The high level idea is as follows: Let us assume we have a constraint of the form $\sum_{i=1}^k x_i = 2$ and chunk size $k/2$. By introducing auxiliary variables $y_1, y_2 \in \{0, 1\}$ we can split the constraint into two equal-sized sub-constraints $\sum_{i=1}^{\lfloor k/2 \rfloor} x_i - y_1 - y_2 = 0$ and $y_1 + y_2 + \sum_{i=\lceil k/2 \rceil}^k x_i = 2$. The technical procedure to do this for BDDs representing arbitrary (also non-arithmetic) constraints is detailed in the Appendix, Section A.3.

4.2 Energy Adaption

We use a pretrained feature extractor to predict vertex-wise features $F_{\mathcal{M}}$ and $F_{\mathcal{N}}$ for both shapes. These features are used in our adaption of the cost function of (P) as

$$c_p = \sum_{v=1}^3 (A_{m_v}^{\mathcal{M}} + A_{n_v}^{\mathcal{N}}) \|F_{m_v}^{\mathcal{X}} - F_{n_v}^{\mathcal{Y}}\|. \quad (1)$$

Here $m_v \in V_{\mathcal{M}}$, $n_v \in V_{\mathcal{N}}$ and $((m_1, n_1), (m_2, n_2), (m_3, n_3))$ forms the p -th product triangle in P . $A_{m_v}^{\mathcal{M}}$ and $A_{n_v}^{\mathcal{N}}$ are the mixed areas according to [39] at vertices m_v and n_v on shapes \mathcal{M} and \mathcal{N} , respectively. Using our learned cost function allows to infuse more descriptive information into the problem compared to the previously used hand designed costs based on elastic energy from [68]. Empirically this yields not only in improved matching accuracy but also reduces ILP solver runtimes due to easier optimisation.

5 Experiments

In this section we experimentally analyse on four shape matching datasets the quality of shape matching as well as runtime of our method and finally show how shapes with higher resolutions can be tackled.

Datasets. We consider the challenging remeshed version FAUST_r [14, 43] of FAUST [6], consisting of 100 human shapes (190 pairs in test set). Furthermore, we choose the dataset SMAL [70] with 49 animal shapes of eight species (136 pairs in test set). We also consider the SHREC’20 dataset [15] which consists of 10 highly non-isometric deformed animal shapes. Lastly, we consider DT4D-H [36], which contains nine classes of humanoid shapes sampled from the DeformingThings 4D [33] dataset. We randomly pick 100 intra-class (DT4d-Intra) and 100 inter-class (DT4d-Inter) pairs for evaluation. For all datasets we follow the train test splits mentioned in [11] and consequently evaluate only on unseen shapes (except for SHREC’20). We evaluate all shape matching methods by downsampling shapes to 450 faces using [25], unless stated otherwise. We compute features on high-resolution shapes and transfer them to the low-resolution shapes using nearest-neighbour search.

Furthermore, we evaluate all methods only on shape pairs which have the same genus, since otherwise there does not exist a solution to (P), i.e., the constraints of the optimisation problem cannot be fulfilled. We repair any defects (non-manifoldness, open boundaries) in the meshes using [4, 59].

Methods.

LP-Relax [68]: The relaxation based approach for solving the ILP (P). Here, binary constraints are relaxed continuously and the largest element in x is iteratively set to one until the constraints are fulfilled. Since this approach does not scale to larger problems we consider it only in some of the experiments.

Sm-Comb [45]: We compare to the shape matching solver for solving the ILP formulation (P). Here, the dual is optimised to obtain improved costs on the primal which drive a custom primal heuristic.

FastDOG [1]: A general purpose solver for binary optimisation problems including the ILP (P). It shows good scalability on problems from vision and machine learning due to massively parallel dual and primal optimisation routines.

URSSM [11]: The current state-of-the-art method for 3D shape matching based on an unsupervised learning-based scheme which, however, does not enforce geometric consistency. We compare against this method to also evaluate if imposing geometric consistency can further improve shape matching performance.

KrnLM [65]: Relaxes a permutation matrix to the set of bi-stochastic matrices while using so-called kernel matrices to efficiently compute solutions of a quadratic assignment problem. For a fair comparison we replace the originally used features with the features computed with the pretrained feature extractor of [11].

DiscoMatch: The ILP formulation with our improved cost function (1) based on the pretrained feature extractor of [11]. For solving the dual (D) we use our quasi-Newton scheme along with the BDD splitting approach from Sec. 4.1. For recovering a primal solution we modify the approach of [1] by replacing their dual solver by ours.

We will also utilize our improved cost function (1) in evaluation of all existing ILP based solvers for a fair comparison unless state otherwise. For CPU-based solvers (LP-Relax Sm-Comb) we use an Intel Core i9-12900K CPU and 64GB RAM. For GPU-based solvers we use an NVIDIA A40 GPU with 48GB DRAM.

Metrics. We evaluate shape matching methods w.r.t. the following metrics

Geodesic Distances: We follow the Princeton protocol [27] for evaluation matching quality. To do so, we compute geodesic distances on the meshes and normalise them by the square-root of the area of shape (cf. [27, Sec. 8.2]).

Conformal Distortion: We show conformal distortion errors [23] to evaluate geometric consistency or rather smoothness of the computed point map. Intuitively, the conformal distortion error evaluates local consistency of triangles after matching (cf. [23, Eq. 3]).

Primal, Dual Objectives: To evaluate performance of ILP solvers for (P) we measure normalised values of primal (P) and dual objectives (D). For dual objective d_t at time t we report the relative dual gap $(d^* - d_t)/d^*$, where d^* is the best known objective for a given instance. To quantify the optimality of a primal solution with objective p we report the primal dual gap $(p - d_{max})/p$, where d_{max} is the maximum dual objective by the respective solver. Note that the primal dual gap cannot be lower than the integrality gap, since (D) is a relaxation of the original ILP (P). We consider a solution to be globally optimal if the primal-dual gap is less than 10^{-2} .

5.1 Results

In Fig. 4, we compare shape matching results w.r.t. mean geodesic errors, conformal distortion errors, and computation times. Results show that the geometric consistency constraints help to decrease mean geodesic errors by sometimes over 50%. At the same time smoothness (by means of conformal distortion errors) of the resulting matching increases consistently across all datasets. Only on SMAL dataset our approach lags behind by a small margin w.r.t. mean geodesic error but still produces a smoother matching. Note that mean geodesic error does not directly quantify geometric consistency and even though results might have small error values, solutions can still be visually implausible (which can also be seen from qualitative results, cf. Fig. 5, Fig. A.3).

From the runtime perspective in Fig. 4 (*bottom*) our method is by far the fastest method among the ILP-based ones (Sm-Comb, FastDOG, DiscoMatch). Moreover our dual optimisation scheme also helps in solving the primal problem (P). It produces most amount of globally optimal solutions, and fails the least number of times as compared to other ILP solvers, cf. Tab. A.1. Note that

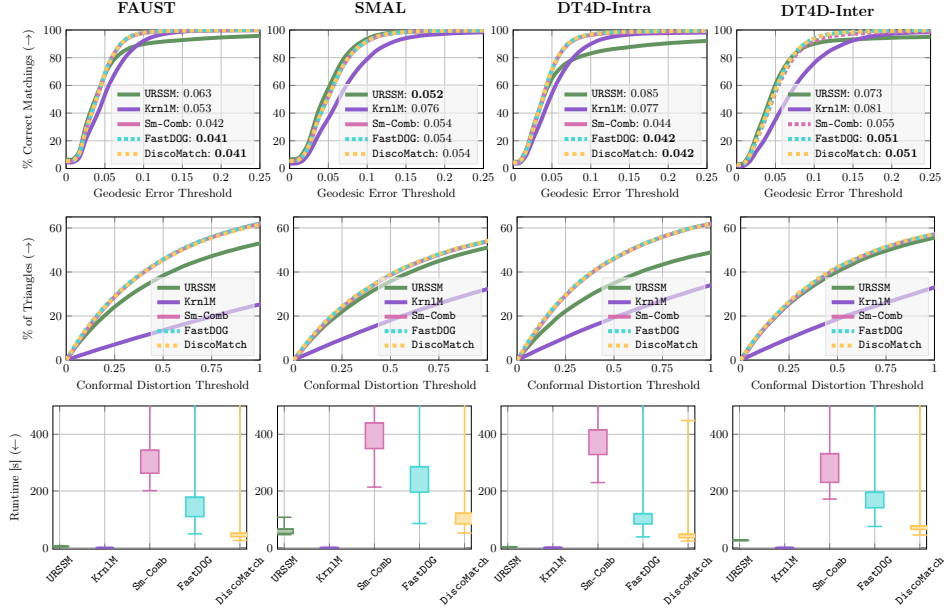


Fig. 4: *Top:* Percentage of correct keypoints w.r.t. geodesic error thresholds on FAUST, SMAL, DT4D-Intra and DT4D-Inter datasets. The numbers in the legends are mean geodesic errors (↓). *Middle:* Conformal distortion errors on FAUST, SMAL, DT4D-Intra and DT4D-Inter datasets. *Bottom:* Statistics of runtime of all methods.

mean geodesic errors of all ILP methods yield very similar matching quality since these methods solve the same optimisation problem.

Furthermore, we observe in Fig. 4 that **KnnLM** struggles on all datasets, likely because it searches over the space of permutation matrices to find shape correspondence which is not a valid assumption for shapes with different discretisation. Thus, we do not include **KnnLM** in further comparisons.

In Fig. 5 and Fig. A.3 we visualize shape matching solutions. For showcasing the effect of geometric consistency we transfer triangulation from one shape to the other via the computed matching. We observe that geometric consistency constraints drastically improve solution quality in our approach as compared to **URSSM** which does not impose geometric consistency.

Ablation: Faster Optimisation. In Fig. 6 we compare the convergence of our solver to the fastest existing approach **FastDOG** for solving the ILP formulation (P). We observe that our algorithm produces much better primal and dual objectives (in significantly less time) compared to **FastDOG**. Moreover, in Tab. A.1 we compare the number of globally optimal and infeasible solutions found by each solver. Our solver finds globally optimal solutions in most cases and the smallest number of infeasible solutions. This shows that our improved dual optimisation also facilitates in primal recovery. In Fig. A.2 we compare runtimes of all ILP-based methods on different problem sizes by varying the res-

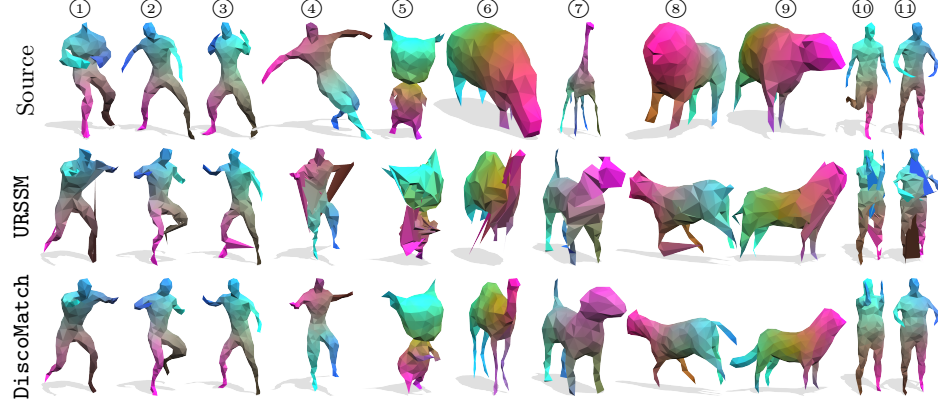


Fig. 5: Qualitative Comparison on DT4D-Inter (①-③), DT4D-Intra (④,⑤), SHREC'20 (⑥,⑦), SMAL (⑧,⑨) and FAUST (⑩,⑪). To better visualise the quality of matchings we transfer triangulation and the color from source to target shapes.

olution (number of triangles) for 5 pairs from the FAUST dataset. We observe that LP-Relax does not scale well with growing problem sizes. While FastDOG is faster than all other existing methods, we outperform it as well.

Note that our solver related improvements from Sec. 4.1 are applicable to ILPs beyond shape matching task as well. To this end we evaluate our solver on some of the difficult datasets used in [1] and obtain 5–10 times runtime improvement. We refer to Sec. A.1 for more details.

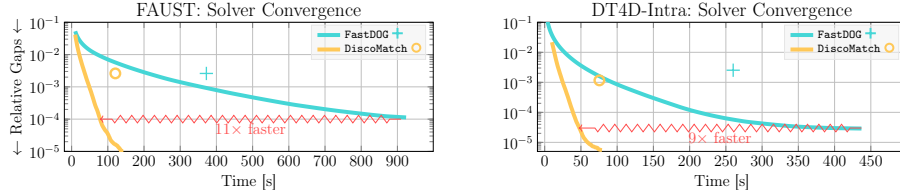


Fig. 6: Convergence plots averaged over a total of 55 instances with varying shape resolutions in $\{500, 550, \dots, 1000\}$. The curves (—) and markers (+, o) represent quality of dual objectives (by relative dual gaps) and primal objectives (by primal-dual gaps), respectively. Our combined solver improvements (Sec 4.1) of quasi-newton updates and constraint splitting give up to $11\times$ improvement in runtime of dual optimisation which also aids in faster primal solutions.

Ablation: Energy Adaptation. In Tab. 2, we compare runtimes and geodesic errors when the ILP (P) is solved with the elastic energy of [69] or with our proposed energy (1). Our solver is the fastest with both energies and furthermore our energy plays a crucial role in decreasing matching errors and runtimes.

	Elastic Energy			Our Energy		
	Time [s.]	Geo. Error	# Inf.	Time [s.]	Geodesic Error	# Inf.
Sm-Comb	648	0.087	12	376	0.042	9
FastDOG	925	0.083	36	327	0.041	4
DiscoMatch	423	0.084	8	134	0.041	2

Table 2: Comparison of solvers for (P) using elastic energy of [69] and our energy (1). We report mean runtime in seconds (Time [s.]), mean geodesic errors (Geo. Error), and number of infeasible solutions (# Inf.) on FAUST dataset (with 450 faces).

5.2 Scaling to Higher Resolutions

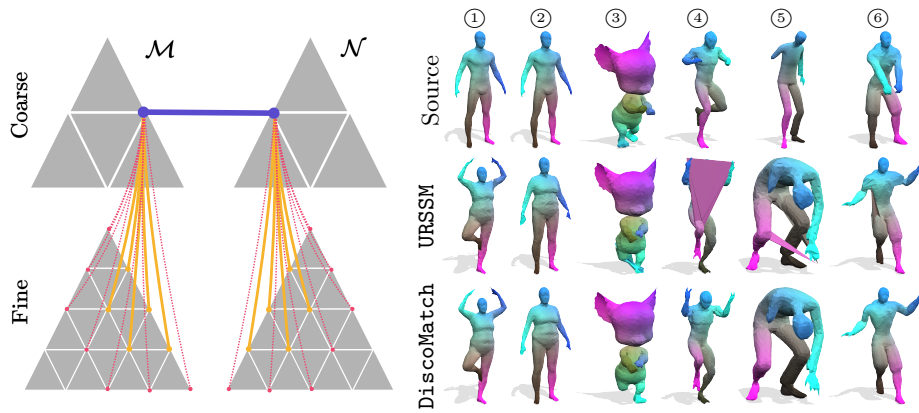


Fig. 7: *Left:* Illustration of our coarse-to-fine pruning strategy. For two matched vertices on the coarse level (—) we only keep potential matchings on the fine level which belong to one-ring (●) and two-ring neighbourhood (●) of respective vertex on coarse level. *Right:* Qualitative matching comparison via triangulation transfer on FAUST (①-②) and DT4D-Intra (③-⑥) datasets for shapes with $4k$ triangles.

As a proof of concept we evaluate our approach on larger shape resolutions by employing a coarse-to-fine strategy inspired by [69]. To this end, we use solutions from coarser shape resolutions to prune (P) on finer shape resolutions. Hence, we start with a resolution of 500 triangles and prune the subsequent ILPs with underlying shape resolutions of $1k$, $2k$, $3k$ up till $4k$ triangles.

For pruning the ILPs we only allow matching between triangles which contain vertices of one-ring and two-ring neighbourhood of a given vertex-vertex matching on the coarser resolution, see Fig. 7 (left).

We evaluate on datasets of FAUST, DT4D-Intra (by subsampling shapes to 4000 triangles) with resulting mean geodesic errors of 0.0193/0.0161 for ours and 0.0197/0.0165 for URSSM respectively. Note that scale of the geodesic error decreases due to lesser discretisation noise on higher resolutions. The average

computation time of the whole coarse-to-fine pipeline is around 20 minutes from which approximately 30% was spent on dual optimisation. In Fig. 7 (right), we show the corresponding qualitative results. Similarly to lower resolution results, our approach leads to better matching quality as compared to URSSM.

6 Discussion & Limitations

Overall, our proposed solver empirically leads to favourable results compared to competing methods: we obtain the largest portion of globally optimal solutions (90%) the smallest amount of infeasible solutions (2%, cf. Tab. A.1), and our solver is up to $11\times$ faster compared to previous approaches (Fig. 6). Despite these positive aspects, there is also a shortcoming: we build upon the product space formalism [68] and also inherit its quadratic space complexity (w.r.t. number of triangles). We found that for our GPU-based solver the memory of modern graphics cards constitutes the main bottleneck when scaling to higher shape resolutions. With a 48GB GPU, we can solve shape matching problems with up to 1000 triangles per shape, which is an order of magnitude smaller compared to purely learning-based shape matching methods (e.g., [11]). As a proof of concept to circumvent the memory bottleneck, we employ a coarse-to-fine scheme allowing us to compute geometrically consistent matchings of shapes with larger resolutions (4000 triangles). Yet, our runtimes for the coarse-to-fine scheme (20 minutes) are still higher than the ones of deep learning methods (few seconds). Nevertheless, we consider our approach to be a large leap forward, especially because our matchings ensure geometric consistency.

7 Conclusion

Our work is an important step for bridging the gap between practical learning-based shape matching and purely combinatorial optimisation formulations. While learning-based methods are fast and robust, they do not allow to enforce solutions with certain structural or geometric properties. In contrast, optimisation-based approaches allow for taking such properties into account, but they are generally slower and lead to challenging non-convex problems. To address these limitations, we pursue a two-fold strategy: (i) we infuse learning-based feature embeddings into the cost function of a combinatorial problem, and (ii) we propose strategies for better optimisation of the Lagrangian dual (D) by quasi-Newton updates and effectively utilising parallelism in modern GPUs. We experimentally demonstrate that this allows us to solve practically relevant 3D shape matching problems faster compared to previous approaches, often even to global optimality. We believe that on the one hand our approach is an important contribution for the field of 3D shape analysis per-se. On the other hand, many downstream applications in computer vision and computer graphics that build upon correspondence problems (e.g. autonomous driving and medical image analysis) can benefit from geometrically consistent matchings.

References

1. Abbas, A., Swoboda, P.: FastDOG: Fast discrete optimization on GPU. In: CVPR (2022) [2](#), [6](#), [7](#), [8](#), [9](#), [10](#), [12](#), [19](#), [20](#), [23](#), [27](#), [28](#)
2. Attaiki, S., Ovsjanikov, M.: Understanding and improving features learned in deep functional maps. In: CVPR (2023) [3](#)
3. Attaiki, S., Pai, G., Ovsjanikov, M.: DPFM: Deep partial functional maps. In: 2021 International Conference on 3D Vision (3DV). pp. 175–185. IEEE (2021) [3](#)
4. Attene, M.: A lightweight approach to repairing digitized polygon meshes. *The visual computer* **26**, 1393–1406 (2010) [9](#)
5. Bernard, F., Suri, Z.K., Theobalt, C.: MINA: Convex mixed-integer programming for non-rigid shape alignment. In: CVPR (2020) [2](#), [4](#)
6. Bogo, F., Romero, J., Loper, M., Black, M.J.: FAUST: Dataset and evaluation for 3D mesh registration. In: CVPR (2014) [9](#)
7. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on* **100**(8), 677–691 (1986) [8](#)
8. Cao, D., Bernard, F.: Unsupervised deep multi-shape matching. In: ECCV (2022) [3](#)
9. Cao, D., Bernard, F.: Self-Supervised Learning for Multimodal Non-Rigid 3D Shape Matching. In: CVPR (2023) [3](#)
10. Cao, D., Eisenberger, M., El Amrani, N., Cremers, D., Bernard, F.: Spectral meets spatial: Harmonising 3d shape matching and interpolation. In: CVPR (2024) [3](#)
11. Cao, D., Roetzer, P., Bernard, F.: Unsupervised learning of robust spectral shape matching. *ACM Transactions on Graphics (TOG)* (2023) [1](#), [2](#), [9](#), [10](#), [14](#), [20](#), [21](#)
12. Coughlan, J., Yuille, A., English, C., Snow, D.: Efficient deformable template detection and localization without user initialization. *Computer Vision and Image Understanding* **78**(3), 303–319 (2000) [4](#)
13. Donati, N., Corman, E., Ovsjanikov, M.: Deep orientation-aware functional maps: Tackling symmetry issues in shape matching. In: CVPR (2022) [3](#)
14. Donati, N., Sharma, A., Ovsjanikov, M.: Deep geometric functional maps: Robust feature learning for shape correspondence. In: CVPR (2020) [3](#), [9](#)
15. Dyke, R.M., Lai, Y.K., Rosin, P.L., Zappalà, S., Dykes, S., Guo, D., Li, K., Marin, R., Melzi, S., Yang, J.: SHREC’20: Shape correspondence with non-isometric deformations. *Computers & Graphics* **92**, 28–43 (2020) [9](#)
16. Ezuz, D., Ben-Chen, M.: Deblurring and denoising of maps between shapes. In: *Computer Graphics Forum*. vol. 36, pp. 165–174. Wiley Online Library (2017) [3](#)
17. Ezuz, D., Heeren, B., Azencot, O., Rumpf, M., Ben-Chen, M.: Elastic correspondence between triangle meshes. In: *Computer Graphics Forum*. vol. 38, pp. 121–134. Wiley Online Library (2019) [3](#)
18. Felzenszwalb, P.F.: Representation and detection of deformable shapes. *TPAMI* **27**(2), 208–220 (2005) [4](#)
19. Gasparetto, A., Cosmo, L., Rodola, E., Bronstein, M., Torsello, A.: Spatial maps: From low rank spectral to sparse spatial functional representations. In: 2017 International Conference on 3D Vision (3DV). pp. 477–485. IEEE (2017) [4](#)
20. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: 3d-coded: 3d correspondences by deep deformation. In: European Conference on Computer Vision (2018) [3](#)
21. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), <https://www.gurobi.com> [28](#)

22. Halimi, O., Litany, O., Rodola, E., Bronstein, A.M., Kimmel, R.: Unsupervised learning of dense shape correspondence. In: CVPR (2019) [3](#)
23. Hormann, K., Greiner, G.: MIPS: An efficient global parametrization method. *Curve and Surface Design: Saint-Malo 1999* pp. 153–162 (2000) [10](#)
24. Huang, Q.X., Guibas, L.: Consistent shape maps via semidefinite programming. In: *Computer graphics forum*. vol. 32, pp. 177–186. Wiley Online Library (2013) [2](#)
25. Jacobson, A., Panozzo, D., et al.: libigl: A simple C++ geometry processing library (2018), <https://libigl.github.io/> [9](#)
26. Jiang, P., Sun, M., Huang, R.: Neural Intrinsic Embedding for Non-rigid Point Cloud Matching. In: CVPR (2023) [3](#)
27. Kim, V.G., Lipman, Y., Funkhouser, T.: Blended intrinsic maps. *ACM transactions on graphics (TOG)* **30**(4), 1–12 (2011) [10](#)
28. Knuth, D.E.: *The Art of Computer Programming: Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, 1st edn. (2011) [23](#)
29. Löhner, Z., Rodola, E., Schmidt, F.R., Bronstein, M.M., Cremers, D.: Efficient globally optimal 2d-to-3d deformable shape matching. In: CVPR (2016) [4](#)
30. Lange, J.H., Swoboda, P.: Efficient message passing for 0–1 ilps with binary decision diagrams. In: *International Conference on Machine Learning*. pp. 6000–6010. PMLR (2021) [6](#)
31. Li, L., Donati, N., Ovsjanikov, M.: Learning multi-resolution functional maps with spectral attention for robust shape matching. NIPS (2022) [3](#)
32. Li, Q., Liu, S., Hu, L., Liu, X.: Shape correspondence using anisotropic Chebyshev spectral CNNs. In: CVPR (2020) [3](#)
33. Li, Y., Takehara, H., Taketomi, T., Zheng, B., Nießner, M.: 4dcomplete: Non-rigid motion estimation beyond the observable surface. In: ICCV (2021) [9](#)
34. Litke, N., Droske, M., Rumpf, M., Schröder, P.: An image processing approach to surface matching. In: *Symposium on Geometry Processing*. vol. 255, pp. 207–216 (2005) [3](#)
35. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Mathematical programming* **45**(1-3), 503–528 (1989) [7](#), [23](#)
36. Magnet, R., Ren, J., Sorkine-Hornung, O., Ovsjanikov, M.: Smooth non-rigid shape matching via effective dirichlet energy optimization. In: *International Conference on 3D Vision (3DV)* (2022) [9](#)
37. Marin, R., Rakotosaona, M.J., Melzi, S., Ovsjanikov, M.: Correspondence learning via linearly-invariant embedding. In: NeurIPS (2020) [3](#)
38. Maron, H., Dym, N., Kezurer, I., Kovalsky, S., Lipman, Y.: Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)* **35**(4), 73 (2016) [4](#)
39. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: *Visualization and mathematics III*. pp. 35–57. Springer (2003) [8](#)
40. Nocedal, J.: Updating quasi-newton matrices with limited storage. *Mathematics of computation* **35**(151), 773–782 (1980) [7](#)
41. Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., Guibas, L.: Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)* **31**(4), 30 (2012) [2](#), [3](#), [4](#)
42. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NIPS (2017) [3](#)
43. Ren, J., Poulenard, A., Wonka, P., Ovsjanikov, M.: Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (ToG)* **37**, 1–16 (2018) [9](#)

44. Roetzer, P., Löhner, Z., Bernard, F.: Conjugate product graphs for globally optimal 2d-3d shape matching. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2023) [4](#)
45. Roetzer, P., Swoboda, P., Cremers, D., Bernard, F.: A scalable combinatorial solver for elastic geometrically consistent 3d shape matching. In: CVPR (2022) [4](#), [5](#), [9](#), [20](#)
46. Roufousse, J.M., Sharma, A., Ovsjanikov, M.: Unsupervised deep learning for structured shape matching. In: ICCV (2019) [3](#)
47. Sahillioglu, Y.: A genetic isometric shape correspondence algorithm with adaptive sampling. ACM Transactions on Graphics (ToG) **37**(5), 1–14 (2018) [2](#)
48. Sahillioglu, Y.: Recent advances in shape correspondence. The Visual Computer **36**(8), 1705–1721 (2020) [3](#)
49. Sankoff, D., Kruskal, J.B.: Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. Addison-Wesley Publishing Company (1983) [4](#)
50. Schmidt, F.R., Töppe, E., Cremers, D.: Efficient planar graph cuts with applications in computer vision. In: CVPR (2009) [4](#)
51. Schmidt, P., Campen, M., Born, J., Kobbelt, L.: Inter-surface maps via constant-curvature metrics. ACM Transactions on Graphics (TOG) **39**(4), 119–1 (2020) [3](#)
52. Schmidt, P., Pieper, D., Kobbelt, L.: Surface maps via adaptive triangulations. Computer Graphics Forum **42**(2) (2023) [2](#), [3](#)
53. Schoenemann, T., Cremers, D.: A combinatorial solution for model-based image segmentation and real-time tracking. TPAMI **32**(7), 1153–1164 (2009) [4](#)
54. Schreiner, J., Asirvatham, A., Praun, E., Hoppe, H.: Inter-surface mapping. In: ACM SIGGRAPH 2004 Papers, pp. 870–877 (2004) [2](#), [3](#)
55. Sharma, A., Ovsjanikov, M.: Weakly supervised deep functional maps for shape matching. NIPS (2020) [3](#)
56. Sharma, A., Horaud, R., Cech, J., Boyer, E.: Topologically-robust 3d shape matching based on diffusion geometry and seed growing. In: CVPR 2011. pp. 2481–2488. IEEE (2011) [3](#)
57. Sharp, N., Attai, S., Crane, K., Ovsjanikov, M.: Diffusionnet: Discretization agnostic learning on surfaces. ACM Transactions on Graphics (TOG) (2022) [3](#)
58. Solomon, J., Peyré, G., Kim, V.G., Sra, S.: Entropic metric alignment for correspondence problems. ACM Transactions on Graphics (ToG) **35**(4), 1–13 (2016) [2](#)
59. Sullivan, C.B., Kaszynski, A.: PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). Journal of Open Source Software **4**(37), 1450 (may 2019). <https://doi.org/10.21105/joss.01450>, <https://doi.org/10.21105/joss.01450> [9](#)
60. Takayama, K.: Compatible intrinsic triangulations. ACM Transactions on Graphics (TOG) **41**(4), 1–12 (2022) [2](#), [3](#)
61. Tam, G.K., Cheng, Z.Q., Lai, Y.K., Langbein, F.C., Liu, Y., Marshall, D., Martin, R.R., Sun, X.F., Rosin, P.L.: Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. IEEE transactions on visualization and computer graphics **19**(7), 1199–1217 (2012) [2](#)
62. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: ICCV (2019) [3](#)
63. Trappolini, G., Cosmo, L., Moschella, L., Marin, R., Melzi, S., Rodolà, E.: Shape registration in the time of transformers. Advances in Neural Information Processing Systems (2021) [3](#)

64. Van Kaick, O., Zhang, H., Hamarneh, G., Cohen-Or, D.: A survey on shape correspondence. In: *Computer graphics forum*. vol. 30, pp. 1681–1707. Wiley Online Library (2011) [2](#), [3](#)
65. Vestner, M., Löhner, Z., Boyarski, A., Litany, O., Slossberg, R., Remez, T., Rodola, E., Bronstein, A., Bronstein, M., Kimmel, R., et al.: Efficient deformable shape correspondence via kernel matching. In: *2017 international conference on 3D vision (3DV)*. pp. 517–526. IEEE (2017) [4](#), [9](#)
66. Vestner, M., Litman, R., Rodola, E., Bronstein, A., Cremers, D.: Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 3327–3336 (2017) [3](#)
67. Wiersma, R., Eisemann, E., Hildebrandt, K.: Cnns on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (ToG)* **39**(4), 92–1 (2020) [3](#)
68. Windheuser, T., Schlickewei, U., Schmidt, F.R., Cremers, D.: Geometrically consistent elastic matching of 3d shapes: A linear programming solution. In: *ICCV* (2011) [2](#), [4](#), [5](#), [8](#), [9](#), [14](#)
69. Windheuser, T., Schlickewei, U., Schimdt, F.R., Cremers, D.: Large-scale integer linear programming for orientation preserving 3d shape matching. In: *Computer Graphics Forum*. vol. 30, pp. 1471–1480. Wiley Online Library (2011) [4](#), [5](#), [12](#), [13](#)
70. Zuffi, S., Kanazawa, A., Jacobs, D.W., Black, M.J.: 3D menagerie: Modeling the 3D shape and pose of animals. In: *CVPR* (2017) [9](#)

A Fast Discrete Optimisation for Geometrically Consistent 3D Shape Matching - Appendix

A.1 Evaluation on other ILP benchmark problems

Although in this work we focus mainly on shape matching problems, our quasi-Newton solver is generally applicable in all cases which [1] can solve, i.e., 0–1 integer linear programs. Moreover, our constraint splitting scheme is also generally applicable however, it is useful only in certain scenarios, i.e., a few constraints containing a larger amount of variables as compared to other constraints.

To check efficacy of our contribution we evaluate our solver on the most difficult instances of cell tracking and graph matching datasets as used in [1]. We only utilise Alg. 1 in our solver and do not use our constraint splitting scheme since both datasets contain constraints of roughly equal size. The results are reported in Figure A.1.

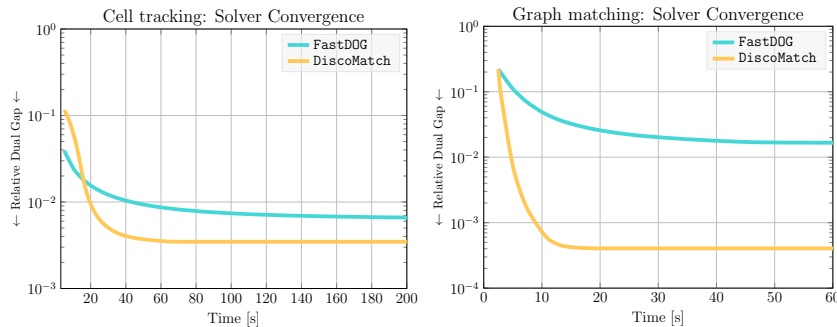


Fig. A.1: Convergence comparison of our quasi-Newton based solver via Alg. 1 versus FastDOG [1] on problems unrelated to shape matching. We report results on the datasets which are most difficult to solve in [1] i.e., the largest instance of cell tracking problem (left) and 20 instances of graph matching for developmental biology (right). In both datasets we reach better objectives of the Lagrangean relaxation (D) measured via gap to the optimal solution (lower values are better).

We observe that on both datasets our solver improves upon [1] by reaching better objectives as measured by gap to the optimum. It also matches the performance of [1] in significantly less time.

A.2 Additional Analysis

In Tab. A.1, we show number of infeasible and optimal solutions per dataset. We observe that our method produces the most of amount of optimal and the least amount of infeasible solutions on all datasets.

Dataset (#Inst.)	FAUST(190)		DT4D-Intra(100)		DT4D-Inter(100)		SMAL(136)		SHREC'20(18)		Total(544)	
	# Opt.	# Inf.	# Opt.	# Inf.	# Opt.	# Inf.	# Opt.	# Inf.	# Opt.	# Inf.	# Opt.	# Inf.
Sm-Comb	149	9	82	2	68	11	101	6	7	2	407	30
FastDOG	170	4	93	1	80	12	119	3	9	0	471	20
DiscoMatch	174	2	97	0	81	9	128	0	12	0	492	11

Table A.1: We compare number of **global optimal** (\uparrow) and **infeasible** (\downarrow) solutions of the individual ILP solvers per dataset. Our solver consistently yields most global optimal and least infeasible solutions. In total, our approach solves 90% of instances to global optimality (FastDOG 87%, Sm-Comb 75%) and produces only 2% of infeasible solutions (FastDOG 4%, Sm-Comb 6%).

Moreover in Fig. A.2 we observe that our solver offers much better scaling to growing instances sizes and finds solutions with lower optimality gaps as compared to other ILP solvers.

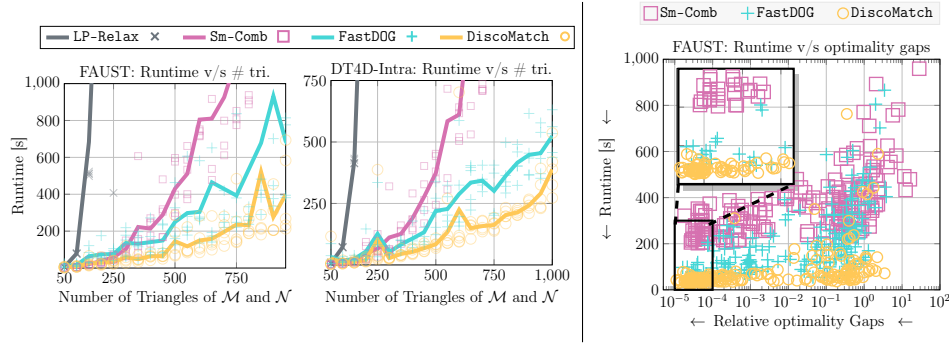


Fig. A.2: *Left:* Runtime comparison w.r.t. shape resolution. Thick lines are mean runtimes and markers show individual experiments. Note that the problem size grows quadratically with the number of triangles of the individual shapes, i.e., the search space size is approximately $21 \cdot |T_{\mathcal{M}}| |T_{\mathcal{N}}|$. *Right:* Our novel combinatorial solver \circ is faster and finds tighter optimality gaps compared to previous combinatorial approaches Sm-Comb [45] and FastDOG [1].

In Fig. A.3, we show more qualitative results on the datasets as well as individual mean geodesic errors for respective pairs. In addition to our solutions being smooth, we can also observe that low mean geodesic error is not necessarily an indicator for geometrically consistent matchings. Some results computed with the method [11] yield lower error scores than ours despite visually obvious geometric inconsistencies. This is due to discretisation artifacts stemming from the *triangle-based* formalism (cf. Fig. 2) that we use to compute matchings. In the datasets that we consider, ground-truth correspondences are given as point-wise maps and thus we need to convert triangle-based matchings into point-wise matchings (e.g., triangle-based matching $((m_1, n_1), (m_2, n_1), (m_3, n_1))$ con-

verts to point-wise matchings (m_1, n_1) , (m_2, n_1) , (m_3, n_1) , where $m_v \in V_{\mathcal{M}}$ and $n_v \in V_{\mathcal{N}}$. Given the ground-truth point-wise matching (m_1, n_1) , our approach does not only capture this point-wise matching, but also includes matchings (m_2, n_1) , (m_3, n_1) which leads to geodesic errors (because (m_2, n_1) , (m_3, n_1) are not part of ground-truth matchings). Consequently, our approach can result in larger mean-geodesic errors due to local discretisation artifacts. Nevertheless, the formulation we use (P) is the only one which guarantees *geometrically consistent* solutions.

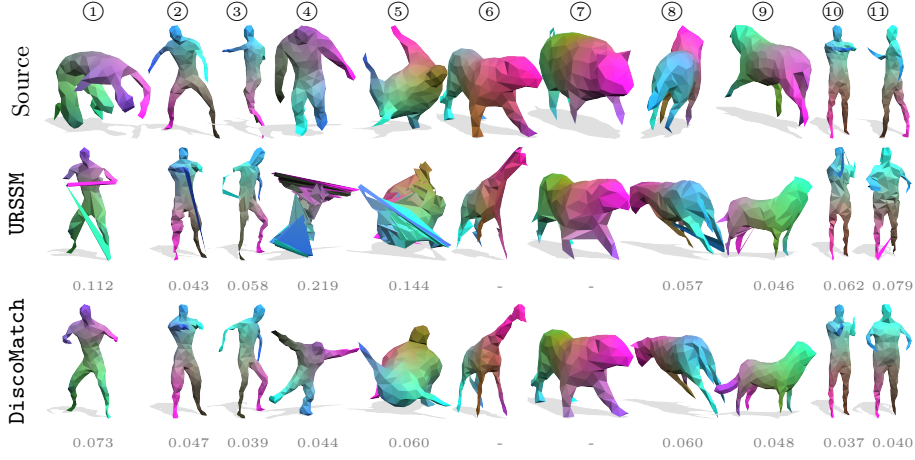


Fig. A.3: Qualitative results on DT4D-Inter (①,②), DT4D-Intra (③-⑤), SHREC’20 (⑥,⑦), SMAL (⑧,⑨) and FAUST (⑩,⑪). We transfer colour and triangulation from source to target shape via computed matching. Numbers are mean geodesic errors for respective instances. Note that SHREC’20 dataset does not come with dense ground truth correspondences. Consequently, we cannot evaluate geodesic errors for these pairs. Furthermore, we observe in columns ②, ⑥ and ⑨ that mean geodesic errors are smaller for [11] even though matchings contain obvious geometric inconsistencies (cf. also Appendix A.2).

Furthermore, we investigate geometric consistency by comparing a version of Dirichlet energies [11, Eq. 10] of the respective matchings. For that, we first rigidly align source and target shape with ground-truth matching. Then we compute the matching deformation field $D \in \mathbb{R}^{|V_{\mathcal{M}}| \times 3}$ as $D = V_{\mathcal{M}} - \Pi_{\mathcal{MN}} V_{\mathcal{N}}$ where $\Pi_{\mathcal{MN}} \in \{0, 1\}^{|V_{\mathcal{M}}| \times |V_{\mathcal{N}}|}$ is the point-wise map, i.e., the computed matching. Finally, we compute the respective Dirichlet energy $E_{\text{Dirichlet}}$ for a matching deformation field D as follows

$$E_{\text{Dirichlet}} = \text{trace}(D^T L_{\mathcal{M}} D) \quad (2)$$

where $L_{\mathcal{M}}$ is the Laplacian of shape \mathcal{M} . In Fig. A.4 we show commulative Dirichlet energies on datasets FAUST, SMAL and DT4D (intra and inter). Considering the shown Dirichlet energies, we can see that Sm-Comb, FastDOG and

DiscoMatch (i.e., all methods solving problem **P** and thus optimising for a geometrically consistent matching) lead to less distorted matchings, i.e., matchings with less Dirichlet energy compared to the non-geometrically consistent method URSSM.

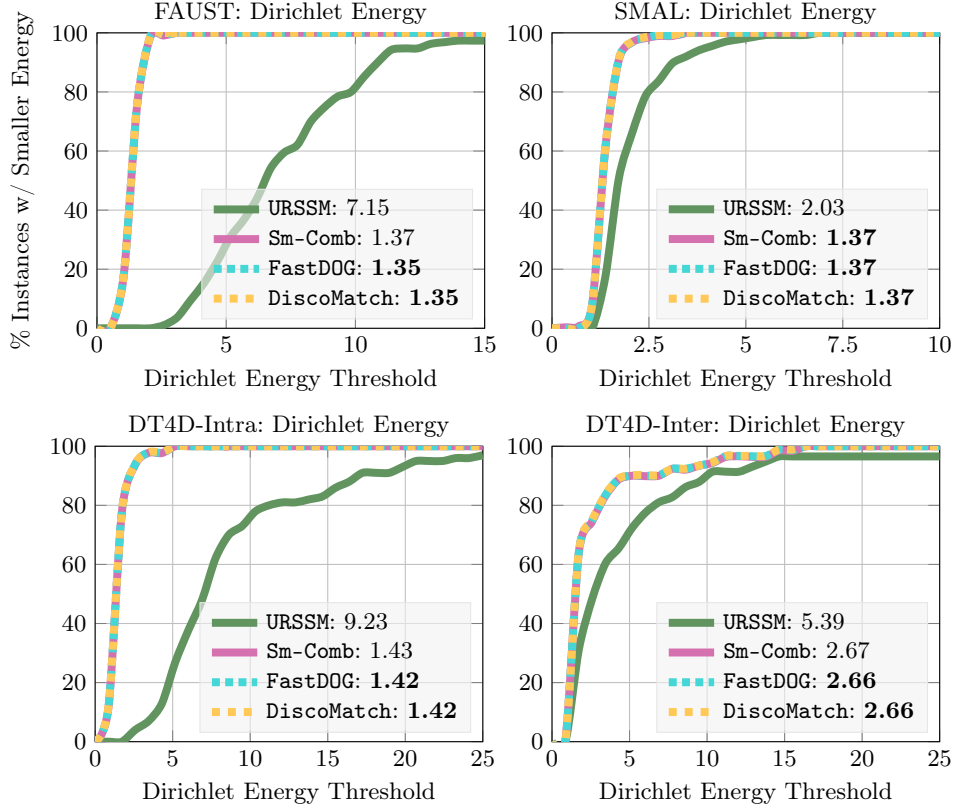


Fig. A.4: Geometric consistency evaluation by comparing Dirichlet energies on FAUST, SMAL, DT4D-Intra and DT4D-Inter. Values in legends are mean Dirichlet energies across all instances. Methods solving problem **P** (i.e., Sm-Comb, FastDOG and DiscoMatch) and thus providing geometric consistency guarantees lead to smaller Dirichlet energies compared to non-geometrically consistent method by URSSM across all datasets.

A.3 Constraint Splitting

A BDD is a directed acyclic graph $G = (V, A^0, A_1)$ that represents Boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$ as follows:

- Each regular node v has an associated variable $\text{var}(v)$. We call all nodes that belong to a variable a layer and $\text{idx}(v) = l$ if v is the l -th node of layer.
- Additionally, there are two terminal nodes, the true terminal \top and the false terminal \perp .
- Each regular node v has two outgoing arcs: a zero-arc $v, v^0 \in A^0$ and a one-arc $v, v^1 \in A^1$. They correspond to assigning the associated variable the respective value.
- Each arc leads either from variable x_i to x_{i+1} or to a terminal.
- There exists exactly one root node associated with x_1 .
- All paths from the unique node associated with x_1 to the true terminal \top correspond to variable assignments x with $f(x) = 1$, all paths to terminal \perp to $f(x) = 0$ (this is actually a small departure from the original BDD definition which allows skipping arcs).

Additionally, BDDs are required to be reduced, i.e., no isomorphic subgraphs are present. For a good introduction to BDDs we refer to [28]. See the top-part of Figure A.5 for a BDD of the constraint $\sum_{i=1}^8 x_i = 2$.

The constraint splitting is described in Algorithm 2. It proceeds as follows: Let a variable x_i be specified after which to split the constraint involving variables x_1, \dots, x_n . Let k be the number of nodes associated with variable x_{i+1} . Then auxiliary variables y_1, \dots, y_k are introduced. The BDD is split into two sub-BDDs, which we call left and right sub-BDD. The left contains variables $x_1, \dots, x_i, y_1, \dots, y_k$, the right one $y_1, \dots, y_k, x_{k+1}, x_n$. The arcs are arranged so that $y_j = 1 \Leftrightarrow$ the j -th node associated to variable x_{i+1} is used.

In Figure A.5 we illustrate the BDD splitting for a simple cardinality constraint.

In Fig. A.6, we show the sparsity pattern of the constraint matrix A of the optimisation problem (P). It visualises the amount of variables involved in individual constraints and especially that the equalities $A^M x = \mathbf{1}$ as well as $A^N x = \mathbf{1}$ involve many variables. The latter motivates our improved parallelism schemes.

Evaluation In Figure A.7 we evaluate performance of our solver (with quasi-Newton updates) with or without the BDD splitting strategy. We observe that constraint splitting gives runtime improvement on top of our contribution of quasi-Newton updates.

In Figure A.8 we report GPU utilisation and number of (sequential) iterations required for a complete BDD traversal in Alg 1. Due to BDD splitting the number of iterations is reduced and GPU utilisation is increased.

A.4 Quasi-Newton Updates

Here we provide implementation details of our Algorithm 1. For quasi-Newton updates we directly use L-BFGS [35] without modifications. For parallel min-marginal averaging we use the dual optimisation algorithm of [1]. For completeness we describe our procedure for quasi-Newton updates below by introducing

Algorithm 2: BDD Splitting

Input: BDD $G = (V, A^0, A^1)$
 split variables x_i
Result: left BDD $G_l = (V_l, A_l^0, A_l^1)$,
 right BDD $G_r = (V_r, A_r^0, A_r^1)$
 // Determine number of auxiliary variables
 1 $k := \max\{\text{idx}(v) : v \in V, \text{var}(v) = i + 1\}$
 // Initialize left and right sub-BDD with left and right part of
 the nodes and arcs of the original BDD
 2 $V_l = \{v \in V : \text{var}(v) \leq i\}$
 3 $A_l^\beta = \{vw \in A^\beta : \text{var}(w) \leq i\}, \beta \in \{0, 1\}$
 4 $V_r = \{v \in V : \text{var}(v) > i\}$
 5 $A_r^\beta = \{vw \in A^\beta : \text{var}(v) > i\}, \beta \in \{0, 1\}$
 // Add auxiliary nodes for left BDD
 6 $V_l = V_l \cup \{aux_{1,1}, \dots, aux_{1,k}\}$
 7 $\text{var}(aux_{1,l}) := y_j \forall l$
 8 **for** $j = 2, \dots, k$ **do**
 9 $V_l = V_l \cup \{aux_{j,1}, \dots, aux_{j,k-j+2}\}$
 10 $\text{var}(aux_{j,l}) := y_j \forall l$
 // connect left BDD part to auxiliary nodes
 11 $A_l^\beta = \cup\{v aux_{1,\text{idx}(w)} : vw \in A^\beta, \text{var}(v) = i\}, \beta \in \{0, 1\}$
 // connect left BDD auxiliary nodes
 12 **for** $j = 1, \dots, k - 1$ **do**
 13 **for** $l = 1, \dots, k - j + 1$ **do**
 14 **if** $l = k - j$ **then**
 15 $A_l^1 = A_l^1 \cup \{aux_{j,k-j} aux_{j+1,k-j}\}$
 16 **else if** $l = k - j + 1$ **then**
 17 $A_l^0 = A_l^0 \cup \{aux_{j,k-j+1} aux_{j+1,k-j}\}$
 18 **else**
 19 $A_l^0 = A_l^0 \cup \{aux_{j,l} aux_{j+1,l}\}$
 20 $A_l^1 = A_l^1 \cup \{aux_{k,1} \top\}$
 21 $A_l^0 = A_l^0 \cup \{aux_{k,2} \top\}$
 // Add auxiliary nodes for right BDD
 22 **for** $j = 1, \dots, k$ **do**
 23 $V_r = V_r \cup \{aux'_{j,1}, \dots, aux'_{j,j}\}$
 24 $\text{var}(aux'_{j,l}) := y_j$
 // Connect right BDD auxiliary nodes
 25 **for** $j = 1, \dots, k - 1$ **do**
 26 $A_r^1 = A_r^1 \cup \{aux'_{j,1} aux'_{j+1,2}\}$
 27 $A_r^0 = A_r^0 \cup \{aux'_{j,l} aux'_{j+1,l+1}\}, l \in \{2, j\}$
 // Connect auxiliary nodes to right BDD part
 28 $A_r^1 = A_r^1 \cup \{aux'_{k,1} w : w \in V, \text{var}(w) = i + 1, \text{idx}(w) = 1\}$
 29 **for** $l=2, \dots, k$ **do**
 30 $A_r^0 = A_r^0 \cup \{aux'_{k,1} w : w \in V, \text{var}(w) = i + 1, \text{idx}(w) = l\}$
 31 Add all missing arcs such that endpoint goes to \perp .

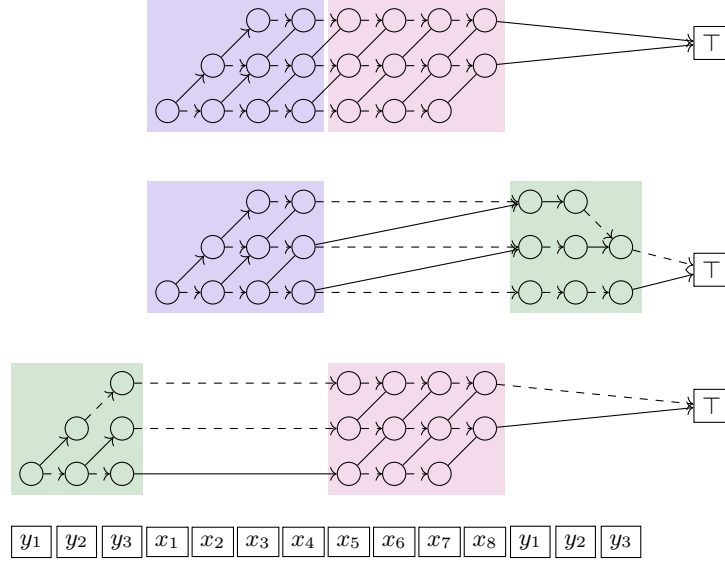


Fig. A.5: BDD splitting for constraint $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 2$ (top) with chunk size 4, resulting in two sub-BDDs (middle and bottom). Three additional variables y_1, y_2, y_3 are introduced. \top denotes true terminal node, solid arcs denote variable assignment to 1 and dashed ones to 0. The false terminal node \perp is not shown for readability. Non-shown arcs all lead to the false terminal \perp .

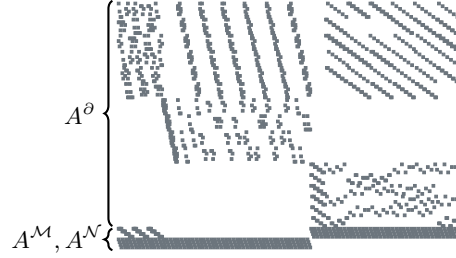


Fig. A.6: Sparsity pattern of constraint matrix A for a pair of shapes with 4 triangles each. We can see that the A^θ block only couples few variables (non-zeros per row ≈ 20), while the A^M and A^N blocks couple many variables (non-zeros per row $\approx 20 \cdot |T_N|$ or rather $\approx 20 \cdot |T_M|$) which motivates the constraint splitting technique for enhanced parallelism.

the following notation

$$\begin{aligned}
 s_k &= \lambda_{k+1} - \lambda_k, \\
 y_k &= g(\lambda_{k+1}) - g(\lambda_k), \\
 \rho_k &= \frac{1}{\langle s_k, y_k \rangle},
 \end{aligned} \tag{3}$$

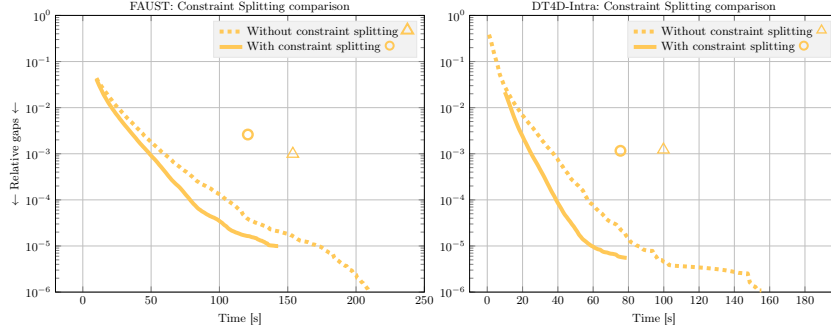


Fig. A.7: Convergence plots with and without constraint splitting. The results are averaged over 5 instances with different shape resolutions from $\{500, 550, \dots, 1000\}$. Curves depict relative dual gaps and markers denote quality of primal solutions through primal-dual gaps. Both solvers use our quasi-Newton updates. Our contribution of constraint splitting Sec 4.1 yields improvement in convergence although with slightly worse dual objectives.

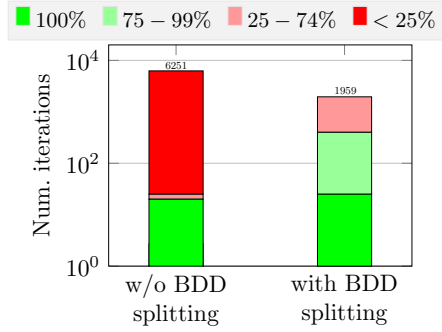


Fig. A.8: Comparison of number of iterations and GPU utilisation in Algorithm 1 with and without BDD splitting on a shape matching problem with 500 triangles. The y-axis denotes number of iterations required for traversing all BDDs (in log-scale) and bar colors indicate GPU utilisation. Due to BDD splitting the number of iterations is reduced and GPU utilisation is increased.

where $g(\lambda_k)$ is a (super-)gradient of (\mathbf{D}) at λ_k .

The L-BFGS algorithm maintains an approximation of the Hessian inverse by storing information of at most m past iterates. The Algorithm 3 describes the procedure for storing such iterates (used in Alg. 1 line 7). Given this information about previous iterates Algorithm 4 computes ascent direction for maximising (\mathbf{D}) which is used in Alg. 1 line 2.

By denoting the dual objective (\mathbf{D}) as $E(\lambda)$ in Algorithm 5 we provide our strategy to find a step size with sufficient improvement. In detail, we increase the step size by a factor $\bar{\alpha} > 1$ if we find non-negative improvement in the objective which is below the threshold Δ_{min} and decrease the step size by $0 <$

$\underline{\alpha} < 1$ if we find non-improvement in the objective. The step size search is done for at most K many iterations for efficiency reasons. If the final step size does not yield an improvement in the objective we do not perform the LBFGS update in the current iteration. For ensuring sufficient ascent we set $\Delta_{min} = 10^{-6}(E(\lambda^1) - E(\lambda^0))$ where λ^0 denotes Lagrange variables at the start of dual optimisation and λ^1 after first invocation of Alg. 1. Rest of the parameters are set as $\epsilon = 10^{-8}$, $\bar{\alpha} = 1.1$, $\underline{\alpha} = 0.8$ and $K = 5$ for all experiments.

Algorithm 3: Inverse Hessian Update (Alg. 1 line 7)

Input: current Lagrange variables and subgradient λ_i, g_i ,
Previous diff. of Lagrange vars. $S = (s_{k-1}, \dots, s_{k-m})$,
Previous diff. of subgradients $Y = (y_{k-1}, \dots, y_{k-m})$
// Check curvature condition
1 **if** $s_i^\top y_i \geq \epsilon$ **then**
2 $S \leftarrow (s_i, s_{k-1}, \dots, s_{k-m-1})$
3 $Y \leftarrow (y_i, y_{k-1}, \dots, y_{k-m-1})$
4 **return** S, Y

Algorithm 4: L-BFGS Direction (Alg. 1 line 2)

Input: current subgradient g ,
Diff. of Lagrange variables $S = (s_k, \dots, s_{k-m+1})$,
Diff. of subgradients $Y = (y_k, \dots, y_{k-m+1})$,
1 **for** $j = k, k-1, \dots, k-m+1$ **do**
2 $\alpha_j = \rho_j s_j^\top g$
3 $g = g - \alpha_j y_j$
4 $d = (s_k^\top y_k / y_k^\top y_k) g$
5 **for** $j = k-m+1, k-m+2, \dots, k$ **do**
6 $\beta = \rho_j y_j^\top r$
7 $d = d + s_j(\alpha_j - \beta)$
8 **return** d

Lastly we observe that the Algorithms 4 and 5 contain trivially parallelisable operations. We perform these operations on GPU and thereby maintain the massively parallel nature of the base solver [1].

A.5 Coarse-to-fine solving strategy

Although we use the primal heuristic of [1] for solving our full shape matching problems, it does not produce feasible solutions in many cases of pruned ILPs. To circumvent this we aim to fix a large amount of confident variables to reduce

Algorithm 5: FindStepSize (Alg. 1 line 4)

Input: Lagrange variables λ , Dual feasible ascent direction d , Previous step size γ

```

1   $\gamma_m = \gamma$ 
2   $E_{init} = E(\lambda + \gamma d)$ 
3  for  $t = 1, \dots, K$  do
4      if  $E(\lambda + \gamma d) \leq E_{init}$  then
5           $\gamma = \underline{\alpha}\gamma$  // Decrease step size
6      else
7           $\gamma = \bar{\alpha}\gamma$  // Increase step size
8      if  $E(\lambda + \gamma d) \geq E(\lambda + \gamma_m d)$  then
9           $\gamma_m = \gamma$ 
10     if  $E(\lambda + \gamma d) - E_{init} \geq \Delta_{min}$  then
11         break
12 return  $\gamma_m$ 

```

problem sizes and solve the rest by Gurobi [21]. To this end, given dual variables λ after convergence of (D) we compute min-marginal differences [1] defined for all variables i and subproblems j as

$$M_i^j = \left[\min_{s \in \mathcal{S}_j, s_i=1} s^\top \lambda^j \right] - \left[\min_{s \in \mathcal{S}_j, s_i=0} s^\top \lambda^j \right]. \quad (4)$$

Note that $M_i^j > 0$ implies the corresponding variable preferring an assignment of 0 and viceversa. Moreover if all subproblems j have min-marginal differences with the same sign the corresponding variable *agrees* on that assignment. For our purposes we compute $\sum_j M_i^j$ of all agreeing variables and fix top scoring 90% of them to their preferable values. The resulting smaller problem is then solved by Gurobi [21]. Although such strategy can still fail, empirically it provides a solution in most cases.