Simulated Annealing for 3D Shape Correspondence

Benjamin Holzschuh
TU Munich
benjamin.holzschuh@tum.de

Zorah Lähner
TU Munich
zorah.laehner@tum.de

Daniel Cremers
TU Munich
cremers@tum.de

Abstract

We propose to use Simulated Annealing to solve the correspondence problem between near-isometric 3D shapes. Our method gains efficiency through quickly upsampling a sparse correspondence by minimizing the embedding error of new samples on the surfaces and applying simulated annealing to refine the result. The algorithm alternates between sampling additional points on the surface and swapping points within the current solution according to Simulated Annealing theory. Simulated Annealing is a probabilistic method and less prone to get stuck in local extrema which allows us to obtain good results on the NP-hard quadratic assignment problem (QAP). Our method can be used as a stand-alone correspondence pipeline through an initial seed generator as well as to densify a set of sparse input matches. Furthermore, the use of locality sensitive hashing to approximate geodesic distances reduces the computational complexity and memory consumption significantly. This allows our algorithm to run on meshes with over $100k$ *points, an accomplishment that few approaches tackling the QAP directly achieve. We show convincing results on datasets like TOSCA and SHREC'19 Connecitvity.*

1. Introduction

Shape correspondence problems occur in a great variety of 2D and 3D vision and graphic processing tasks. They can be applied in many applications, e.g. texture transfer, recognition or statistical shape models. These applications become more and more relevant with the rise of VR and AR, and the need for scalable algorithms increases with the precision of acquisition hardware. In its essence, the shape correspondence problem aims to find a semantically meaningful mapping between the points on two compact two-dimensional Riemannian manifolds $\mathcal{X}$ and $\mathcal{Y}$, i.e. a function $\varphi : \mathcal{X} \to \mathcal{Y}$. The definition of semantically meaningful can vary depending on the application but it is common to choose pair-wise features, e.g. distance values, to be preserved. In case of a rigid transformation between $\mathcal{X}$ and $\mathcal{Y}$ this translates to preserving the Euclidean distance between points and the problem has six degrees of freedom, which makes it efficiently solvable. In more general cases and with discretized shapes this can be formulated as a version of the NP-hard Quadratic Assignment Problem (QAP):

$$\Pi^* = \underset{\Pi \in \mathcal{P}_n}{\arg\max} \sum_{x,y \in \mathcal{X}} k_{\mathcal{X}}(x, y) \cdot k_{\mathcal{Y}}(\Pi(x), \Pi(y)). \quad (1)$$

Here, $\mathcal{P}_n$ denotes the set of n-permuations assuming both shapes are discretized with n points and $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ denote an arbitrarily chosen measure of the closeness between two points on $\mathcal{X}$ and $\mathcal{Y}$. A wide variety of relaxations for this formulation exist (see Section 2) but are often still not feasible for a large number of vertices. Another problem with solving Eq. (1) exactly is the underlying assumption that φ is an bijection. While this is reasonable in the continuous formulation, it requires the same amount of vertices on both shapes. This assumption is often not met in real-world data and needs to be artificially enforced through subsampling which adds additional pre- and postprocessing and might distort the result, for example for partial shapes.

Contribution In this paper, we propose to compute a correspondence between two 3D shapes by approximating the solution to Eq. (1) using a simulated annealing strategy [16]. Our main contributions are the following:

- We propose the first scalable application of Simulated Annealing to the 3D non-rigid correspondence problem.
- Although QAPs are a NP-hard problem, we approximate (1) in $O(n \log(n)\sqrt{n})$ runtime where n is the number of vertices.
- We propose to use a variant of locality sensitive hashing to reduce the memory requirement to $O(n\sqrt{n})$.
- In numerous experiments, we demonstrate that the proposed algorithm can be used both as a stand-alone framework with a seed generator as well as complete and denoise a set of sparse input matches. It provides state-of-the-art results and scales to over 100K vertices.

2. Background & Related Work

In this section we summarize the general background needed to understand the rest of the paper and related work most relevant to our method. A more general survey of recent shape correspondence methods can be found in [26].

We denote the input triangular meshes as $\mathcal{X}$ and $\mathcal{Y}$ and assume that they are proper discretizations of Riemannian 2-manifolds embedded in 3D. The set of vertices of each is denoted by $\{x_1, ..., x_n\}$ and $\{y_1, ..., y_m\}$ respectively.

2.1. Quadratic Assignment Problem

Modeling the correspondence problem in variants of the Quadratic Assignment Problem has a long history [3]. If both shapes have the same number of points, then the correspondence ϕ can be represented by a permutation π, which maps the vertices $\{x_1, ..., x_n\}$ of the triangluar mesh $\mathcal{X}$ to the vertices $\{y_1, ..., y_n\}$ of the triangluar mesh $\mathcal{Y}$. Following the approach of [32], the optimal permutation π can be described as the solution to a quadratic assignment problem (QAP) with the objective

$$\max_{\pi} \sum_{1 \leq i,j \leq n} k_{\mathcal{X}}(x_i, x_j) k_{\mathcal{Y}}(y_{\pi(i)}, y_{\pi(j)}). \tag{2}$$

This objective is also referred to as the *Koopman-Beckman* version of a QAP. However, as the original formulation, all variants are NP-hard and in general not tractable for instances with more than a few dozen points. This also holds for the Quadratic Assignment Matching (QAM) [8]. It was even shown that finding a ϵ-approximate in polynomial time for any ϵ is only possible if P=NP [27].

Relaxations of the permutation matrix constraint are a popular way to reduce the computational complexity of (1). Spectral relaxations as introduced in [17] replace permutations with a Frobenius norm constraint which reduces the optimization to an eigenvector problem. Other popular relaxations consider doubly-stochastic (DS) matrices instead [5, 9] which preserves the optimum for concave energies and on specific graph matching cases [1] but not for general non-convex energies. Other relaxations can be shown to be tight but are still too computationally demanding to apply to high-resolution scans [4, 13].

The Product Manifold Filter (PMF) [33] solves the same optimization problem as our work with a series of Linear Assignment Problems (LAPs). However, it cannot work as a stand-alone method, and only works on full shapes with the same resolution and requires a (possibly sparse) initialization. Additionally, the size of the problem is restricted by the size of LAP that can be solved, usually not more than a few thousand vertices. PMF has been extended to work with features as initialization and on much higher resolutions with a multi-scale approach in [32], but cannot densify sparse inputs anymore. Both [33] and [32] are prone to get stuck in local optima without chance of recovering whereas our framework starts with a variety of initializations and applies a probabilistic approach which makes it more flexible.

2.2. Approximation Algorithms

Approaches that do not have any guarantees on being close to the optimal solution can still work well in practice and actually be more efficient. One class of algorithms, that our method also falls into, looks for small step improvements over the current solution. A famous member of this class is [12] solving graph isomorphism. If the improvement step has a probabilistic condition it is possible to escape local optimal in very non-convex problems.

Genetic algorithms [21] fall under probabilistic optimization with an idea based on evolution theory, namely mutation and selection. Different than Simulated Annealing genetic algorithms maintain multiple solutions throughout the optimization. This is more efficient in exploring the solution space but increases the computational complexity. [25] explored this direction for 3D correspondences, but due to the aforementioned complexity it is not efficient enough to produce a dense correspondence on high resolution shapes. [25] is similar to our pipeline in that its starts with a very sparse set of correspondences that are refined and expanded iteratively but only generates a fixed sized sparse solution where we can sample indefinitely. [10] also applied a genetic algorithm for 3D shape correspondence but operates on maps. This scales to high resolutions but relies on a reliable method to convert the map back to a pointwise correspondence.

Simulated Annealing, which we use in this work, is a variant of the Metropolis algorithm [20] that can approximate the global solution for complex functions and solution spaces that defy conventional optimization techniques [23]. A solution to a discrete optimization problem is identified by the physical state of a set of atoms. Starting with an initial state and temperature t, a random generator produces displacements of the atoms which change the energy E (objective function) so that the system of atoms remains admissible. A displacement du is applied to a randomly chosen atom and then the change in energy dE is assessed. If $dE < 0$ the displacement is accepted. Otherwise

$$P(dE) := e^{-\frac{dE}{kt}} \tag{3}$$

is evaluated, where k is the Boltzmann constant. $P(dE)$ is compared with a random variable X that is uniformly distributed in $(0, 1)$ and we accept the displacement if $X < P(dE)$. The temperature t controls the flexibility to accept changes increasing the objective function value and is gradually lowered over the course of the optimization. In the

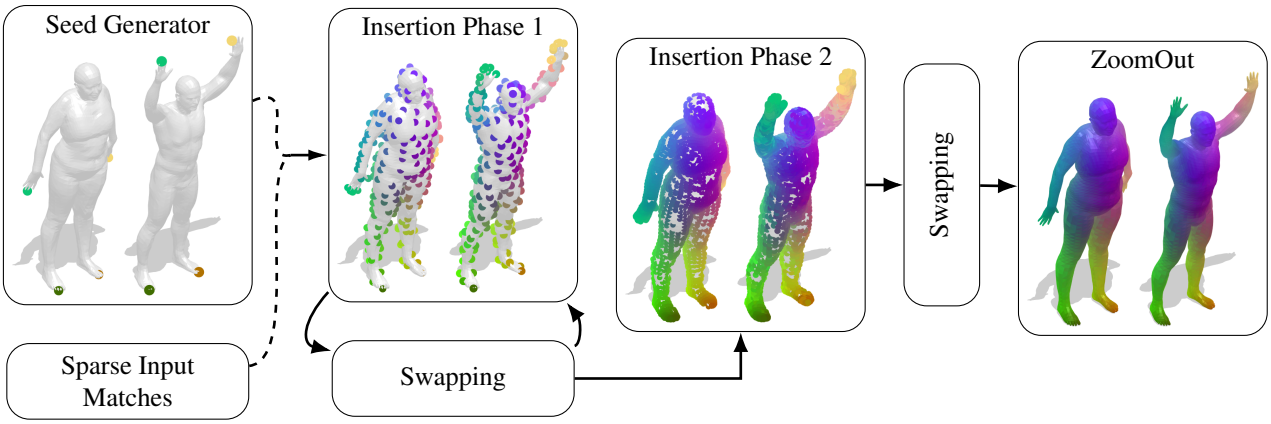


Figure 1: Overview over our pipeline. The initialization can either be a set of sparse input matches or produced by our proposed seed generator. The next phase alternates between inserting new points and swapping the current (sparse) solution according to Simulated Annealing strategies until 250 points are filled in. After that the solution is filled to 70% of all points with a final round of swapping. The final step is a round of post-processing with ZoomOut [19] to densify the solution.

case of the QAP, the random displacements are transpositions and the corresponding change in energy can be computed in $O(n)$. Simulated Annealing has been used to find point correspondences for stereo vision [29] or protein prediction [28]. However, these algorithms do not scale to large candidate sets but instead operate on special interest points or return a sparse set of correspondences.

3. Method

We propose to use Simulated Annealing (SA) to optimize for the optimal permutation Π^* in this QAP

$$\Pi^* = \underset{\Pi \in \mathcal{P}_n}{\arg\max} \sum_{x,y \in \mathcal{X}} k_{\mathcal{X}}(x,y) \cdot k_{\mathcal{Y}}(\Pi(x), \Pi(y)). \quad (4)$$

We propose a specific choice for $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$:

$$k_{\mathcal{X}}(x,y) := e^{-d_{\mathcal{X}}(x,y)} \quad \text{and} \quad k_{\mathcal{Y}}(x,y) := e^{-d_{\mathcal{Y}}(x,y)}, \quad (5)$$

where $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ denote the geodesic distance on $\mathcal{X}$ and $\mathcal{Y}$. This definition tries to incentivize smoothness because a single point x_i on $\mathcal{X}$ contributes the most to the optimal objective (2) if all points in its neighborhood have correspondences on $\mathcal{Y}$ that are also in the neighborhood of $\Pi(x_i)$. In SA the current solution and a neighboring one are evaluated in terms of energy (Eq. (4)), and the neighbor is accepted with a certain probability based on the energy change and current temperature (see Section 2.2 for details). In case of the correspondence problem the current solution is a (sub)permutation Π and a neighbor is a subpermutation, where two matches $\Pi(x) = x', \Pi(y) = y'$ are switched such that $\Pi(x) = y', \Pi(y) = x'$. If at some point no more random displacements are accepted, the system is 'frozen'.

Due to the complexity of the problem for large number of vertices n, we propose several adjustments to make SA more efficient in both memory consumption and runtime. We use locality sensitive hashing instead of calculating and storing all geodesic distances, see Section 3.1. Furthermore, we introduce a seed generator to produce a sparse initialization well suited for our algorithm in Section 3.5. Based on the sparse initialization our algorithm gradually adds more points (Section 3.3) while already refining the solution with SA (Section 3.4). We focus on refining when the solution is not dense because this allows to correct errors with less swapping operations.

3.1. Locality Sensitive Hashing

We use locality sensitive hashing [11] instead of calculating the entire geodesic distance matrices for $\mathcal{X}, \mathcal{Y}$. Calculating the entire distance matrix is slow and not feasible for high resolutions, since it contains n^2 elements.

For a set of points $\mathcal{S} = \{s_1, ..., s_n\}$ and an arbitrary distance function $d : \mathcal{S} \to \mathbb{R}$ locality sensitive hashing approximates the distances by selecting a suitable subset $Z \subset \mathcal{S}$ with $|Z| \ll n$ and considers the inequalities obtained from the elementary triangle inequality:

$$\max_{z \in Z} |d(z,s) - d(z,s^*)| \leq d(s,s^*) \leq \min_{z \in Z} d(z,s) + d(z,s^*), \quad (6)$$

which holds for all $s, s^* \in \mathcal{S}$. The tightness of these inequalities is determined by $\mathcal{S}$, d and Z. We demonstrate how well we can approximate the geodesic distance d on a sphere in Appendix B. Effectively, each $z \in Z$ serves as a hash function which projects all points $\{s \in \mathcal{S} \,|\, d(s,z) = t\}$ to a single number $t \in \mathbb{R}_+$. If two points s_1 and s_2 have a similar profile, i.e. $d(s_1, z) \approx d(s_2, z)$ for all $z \in Z$, then s_1 and s_2 are likely to be very close on $\mathcal{S}$. We call the points contained in Z *basis points*.

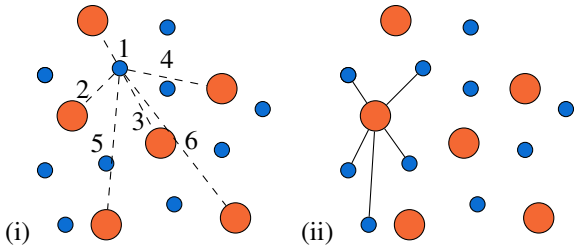


Figure 2: Connections between basis points (orange) and their surrounding vertices (blue). (i) For each x, basis points are sorted by their distance as labeled on the dashed lines. (ii) The environment of a basis point b are all vertices such that b is one of their three closest basis points (solid lines).

Basis Points For $\mathcal{X}$ we choose $\sqrt{n}$ many basis points by farthest point sampling and precompute the order of nearest basis points with decreasing $k_{\mathcal{X}}(x, z)$ for each vertex x on $\mathcal{X}$. Most $k_{\mathcal{X}}(x, z)$ will be close to 0 because $k_{\mathcal{X}}(x, z)$ decays exponentially with increasing distance. Assuming that the points x and the basis points z are evenly distributed across the shape's surface, only few basis points will be close to x, and we define the environment of a basis points z as all points x to which z is among the three closest basis points. As a result, the environments of close basis points will be overlapping and each point on X will be contained in the environment of exactly three basis point environments. Close points to x can be computed by merging the environments of the basis points that are close to x. Depending on how many close points we want, we can change the number of close basis points that we consider. This behavior will be important in Section 3.3. Note that if $|Z| = \Theta(\sqrt{n})$, then the cardinality of the environments of the basis points will be $\Theta(\sqrt{n})$ on average. Ordering the basis points' distances for each vertex on $\mathcal{X}$ can be done with an efficient implementation in $O(n|Z|\log(|Z|))$. The construction and analysis on $\mathcal{Y}$ is equivalent.

3.2. Optimization

Our SA process has five stages that depend on the number of vertices k already added to the solution. First, we start on several sparse subpermutations π_k produced by the seed generator as the initial seed to which we add gradually more matches. SA is applied regularly during insertion until a sufficient number of points are matched in Π_k. In our experiments we found that the final matching was already well characterized by approx. 250 points and [33] has shown that a well distributed subpermutation defines the dense solution well enough [33]. Each system is 'frozen' by another application of SA with temperature 0 and each Π_k is evaluated according to the QAP objective (4). The Π_k with the best score is finalized by inserting the remaining points into the matching until 70% of points are matched. We do not match all points through the insertion process because the last insertions are prone to produce outliers. A final SA phase with zero temperature is applied and in the end ZoomOut [19] is used to produce a dense solution. We focus on applying SA early because the amount of swaps needed to escape a random, dense solution is very high.

Inconsistent Mesh Resolution. A challenging nature of the shape correspondence problem can be observed, when the number of vertices on the mesh $\mathcal{X}$ is less than the number of vertices on $\mathcal{Y}$. For instance, let the same Riemannian manifold be discretized with two different resolutions, one of which is much finer than the other. We cannot require that an optimal correspondence in this case is bijective anymore but if the number of vertices on $\mathcal{X}$ is much smaller, injectivity is still possible. However, an optimal solution will result in a matching in which all points on $\mathcal{Y}$ are clustered together instead of being evenly distributed. This is a direct result of the formulation of objective (2), as points can only contribute to the sum if they are close together. The Cauchy-Schwartz inequality, i.e. for two vectors $x, y \in \mathbb{R}^n$

$$\langle x, y\rangle = \left(\sum_{i=1}^{n} x_i y_i\right)^2 \leq \left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i^2\right) \quad (7)$$

implies that $\langle x, y\rangle$ is maximal, if $x_i = y_i$. This suggests that objective (2) is near optimal, if $k_{\mathcal{X}}(x_i, x_j) \approx k_{\mathcal{Y}}(y_{\pi(i)}, y_{\pi(j)})$. Based on this, we propose a surrogate function for adding new correspondences to a partial matching which aims to preserve the measure of closeness between two points on $\mathcal{X}$ and their matchings on $\mathcal{Y}$ and can be optimized greedily. Let $\hat{X}, \hat{Y}$ be the sets of already matched points on $\mathcal{X}$ and $\mathcal{Y}$ respectively and let $y(\hat{x})$ be the correspondence of $\hat{x} \in \hat{X}$ on $\mathcal{Y}$. We match a new point $x \in \mathcal{X}$ to

$$\hat{y}^* = \underset{\hat{y}\in\mathcal{Y}\setminus\hat{Y}}{\arg\min} \sum_{\hat{x}\in\hat{X}} |k_{\mathcal{X}}(\hat{x}, x) - k_{\mathcal{Y}}(y(\hat{x}), \hat{y})|. \quad (8)$$

$\hat{y}^*$ is the point on $\mathcal{Y}$ whose embedding with respect to $\hat{Y}$ is the closest to how $\hat{x}$ is embedded in relation to $\hat{X}$.

3.3. Point Insertion

Point insertion refers to the extension of the existing subpermutation by a a single pair of points $(x, y) \in \mathcal{X} \times \mathcal{Y}$, which is done repeatedly during the optimization process. As discussed in the previous subsection, for a given point x on $\mathcal{X}$ we want to find a yet unmatched point $y \in \mathcal{Y}$ which minimizes (8). There are two stages of how we choose the point x based on how many points were already matched.

3.3.1 First stage:

The goal of the first stage is to achieve an evenly distributed cover of both surfaces with the inserted points. We add a

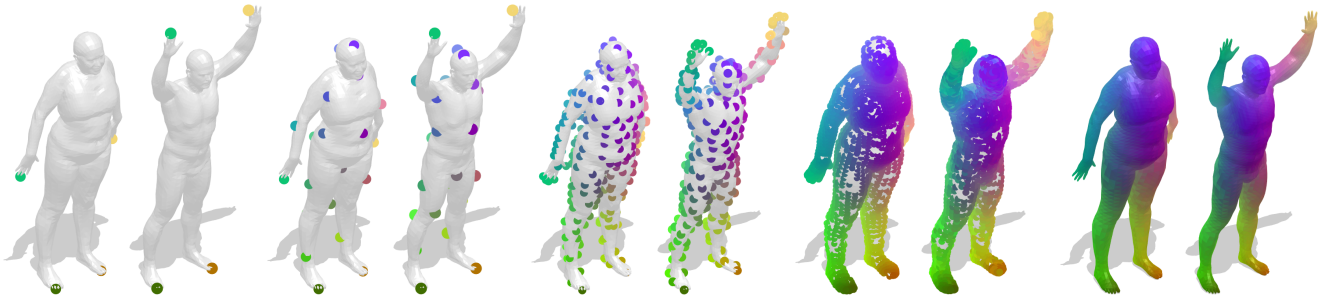


Figure 3: Insertion process of the algorithm visualized on two shapes from the FAUST dataset. From left to right: 1. The 4 initial seed points generated by the seed generator are well distributed over the surface. 2. The first 25 points added within the first phase of point insertion. 3. The first phase ends when 250 points were added. Until this point swapping operations refine the solution regularly. 4. The second insertion phase stops when the majority of points were added. Some noise and outliers are visible. 5. The final solution is dense and refined.

new pair to the existing matching as follows

1. Select a point on $x \in \mathcal{X}$ by farthest point sampling with respect to the already matched points.
2. Find the best k best matching basis points according to Eq. (8) on $\mathcal{Y}$ which we call $\hat{B}$. We choose $k = 3$.
3. Among the points contained in the environments of every basis point in $\hat{B}$, match the optimal unmatched point according to Eq. (8) with x.

Pre-selecting the basis points reduces the search space significantly. We alternate between sampling a point from $\mathcal{X}$ and finding a matching candidate on $\mathcal{Y}$, and sampling a point on $\mathcal{Y}$ and finding the best matching candidate on $\mathcal{X}$. This guarantees that the matched points are evenly distributed on both surfaces after the first phase terminates. In our experiments we stop the first phase after 250 points have been matched. The first phase of the insertion process is visualized in Figure 3 and already indicates that 250 matches are dense enough to infer the rest of the correspondence. Therefore, we switch to the second insertion stage which efficiently interpolates the current result.

3.3.2 Second stage:

In the second phase, we sample a new point x randomly from all yet unmatched points on $\mathcal{X}$. During this phase it might happen that we can no longer find an unmatched point on $\mathcal{Y}$ in the environments of the best matching basis points. To still enforce the smoothness condition whenever possible, we use the following algorithm:

1. Find all already matched points that are close to x. This can be efficiently done by looking up the top m closest basis points around x and merging their environments. We denote the obtained set by $\mathrm{E}_{\mathcal{X}}(x)$. Let $\mathrm{UE}_{\mathcal{X}}(x) \subset \mathrm{E}_{\mathcal{X}}(x)$ contain only unmatched points and $\mathrm{ME}_{\mathcal{X}}(x)$ only matched points.
2. For each element $\hat{x}$ in $\mathrm{ME}_{\mathcal{X}}(x)$ find $\mathrm{E}_{\mathcal{Y}}(\hat{y}(\hat{x}))$ analogously to the step above where $\hat{y}(\hat{x})$ denotes the correspondence of x on $\mathcal{Y}$. The set of candidates is now $\hat{E} := \hat{E}(x) = \bigcup_{\hat{x} \in \mathrm{ME}_{\mathcal{X}}(x)} \mathrm{UE}_{\mathcal{Y}}(\hat{y}(\hat{x}))$.
3. If $\hat{E} = \emptyset$ we gradually increase m by 3. Once m is greater or equal to the number of basis points, $\hat{E}$ is identical to the set of unmatched points on $\mathcal{Y}$ and we are guaranteed to find a match.
4. The matching candidate $y \in \hat{E}$ for x is chosen according to Eq. (8). When evaluating this score, we only sum over $\mathrm{ME}_{\mathcal{X}}(x)$ instead of all matched points $\hat{X}$.

Note that we stop expanding the sets $\mathrm{ME}_{\mathcal{X}}(x)$ and $\hat{E}(x)$ in step 1 and 2 if they contain too many elements to guarantee the time bounds of the algorithm. See Appendix B for a derivation of the algorithmic complexity. We effectively search for a candidate y only among the points that are close to the correspondences of the matched points on $\mathcal{X}$ that are close to x and score it according to them as well. Note that this only works well as both shapes are sufficiently covered after the first phase of the insertion process.

However, the matches that are inserted last often suffer from not finding a good unmatched correspondence on Y since most points on Y are already matched (this happens after approx. 90% of all points were inserted). See Section 4.2 for experiments showing the evolution of the quality of our results during insertion.

3.4. Swapping

The swapping according to SA strategy is applied multiple times during the algorithm (see Section 3.2). During the first insertion phase SA is applied repeatedly after a fixed number of particles (in our experiments we used 10) has been inserted using the same initial temperature. Since we evaluate multiple seeds, we do not want a too high temperature because this would impair the diversity of the seeds. On

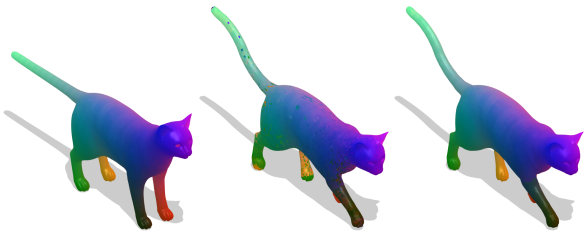


Figure 4: Correspondence example of our method from an input shape (left) with our method without ZoomOut (middle) and with ZoomOut (right). Due to the sequential adding the last points can often not be placed correctly, leading to some extreme outliers. These can be easily removed by using ZoomOut as post-processing.

the other hand, the temperature should not be too low so that meaningful improvements to the subpermutation are possible. In our experiments we used a temperature of $t = 0.01$.

We pick possible transpositions randomly between matched points on both shapes. The improvements are calculated only with respect to the points on $\mathcal{X}$ that have a correspondence on $\mathcal{Y}$ according to (8). Depending on the temperature and the improvement the transposition is accepted. This is repeated 2.500 times. In the last SA phase of the first insertion stage, we decrease the temperature to $t = 0$ to move the system to a local optimum.

At the end of the algorithm, after all points were inserted, we evaluate how much each match in the current (sub)correspondence π contributes to the objective (2) and the worst 20% of contributors are marked as candidates for refinement in a final SA phase with temperature $t = 0$. The local contribution of $x \in \mathcal{X}$ can be calculated as:

$$\sum_{x' \in \mathrm{ME}_{\mathcal{X}}(x)} k_{\mathcal{X}}(x, x') k_{\mathcal{Y}}(y(x), y(x')). \tag{9}$$

We iterate through all refinement candidates x and check if there are other refinement candidates $y \in \hat{E}(x)$ (i.e the environments of the points on $\mathcal{X}$ that correspond to points on $\mathcal{Y}$ which are close to the correspondence of x on $\mathcal{Y}$, see Section 3.3) and check if a transposition of those pairs improves their overall contribution.

3.5. Seed Generator

Our method needs an initial set of sparse seeds to start the insertion process as described in Section 3.3. These can either be given by the user or come from the seed generator.

The seed generator produces and evaluates a series of k-submatchings (sets of k matches that are locally bijective) called *seeds*. The best seed according to Eq (4) is used as the initialization for the rest of the pipeline. The embedding of new points, Eq. (8), relies on the fact that the initial points are well distributed over the surface and therefore we sample potential seeds with farthest point sampling. For instance, when matching shapes resembling human bodies, a promising seed should match points located on the limbs of one body to the similar counterpart on the other. These property should be kept in mind when running the rest of the pipeline on arbitrary input matches. We produce all seeds by sampling m distinct points via farthest point sampling from each of the input shapes. These are called $M_{\mathcal{X}}$ and $M_{\mathcal{Y}}$. There are $m!$ possible correspondences between $M_{\mathcal{X}}$ and $M_{\mathcal{Y}}$. However, it is unlikely that any of them is completely meaningful because the point sets were sampled independently of each other and the perfect match might not have been sampled. To be more robust to inconsistent samplings we only keep $k < m$ of the candidates. There are $\binom{m}{k}$ many different k-sized subsets of an m-sized set. Hence we can generate $\binom{m}{k}\binom{m}{k}k!$ many different k-sized seeds from the candidate point sets on $\mathcal{X}$ and $\mathcal{Y}$. This works well because points at the tip of extremities are the the furthest away from many subsets and sampled early with a very high probability. In general, the difference between m and k should not be too large, as it drastically increases the number of seeds that need to be evaluated. In our experiments, we obtained good matchings for seeds constructed with $m = 4$ and $k = 3$. See Figure 3 for an example of the final seed.

4. Experiments

We show experiments evaluating the matching error of our method in comparison to state-of-the-art methods on popular data sets in Section 4.1 and an ablation study in Section 4.2, as well as some qualitative examples. We evaluate our algorithm according to the Princeton benchmark protocol, see [14]. If the matching produced by our algorithm contains the pair (x, y), then we plot its accumulated error $\epsilon(x) := d_{\mathcal{Y}}(x, \hat{y})/\mathrm{diam}(\mathcal{Y})$, where $\mathrm{diam}(\mathcal{Y})$ is the diameter of $\mathcal{Y}$ and the pair $(x, \hat{y})$ is given by a known optimal matching. In the quantitative results we choose only non-learning methods to evaluate against to keep the results comparable.

4.1. Quantitative Results

We evaluate our algorithm quantitatively on the TOSCA dataset [7], the FAUST dataset [6] and the SHREC Connectivity dataset [18]. The ground-truth correspondences for all these datasets are known. The TOSCA dataset consists of 8 classes of triangular meshes resembling animals and humans in different positions and ranging from 3.000 to 50.000 vertices. The results can be seen in Figure 6. The FAUST registration dataset contains 100 shapes from 10 people in different poses. See Figure 7 for our results including results of the ablation study. The SHREC Connectivity dataset contains 44 human shapes with a wide range of resolutions and inconsistent meshings even within the shape. This is a very challenging setup for many algorithms.

Figure 5: Matching between a (downsampled) cat and a centaur from the TOSCA dataset. From left to right: original coloring on the cat, our method before ZoomOut without any swapping (notice one wrong leg), with only greedy swapping (zero temperature), with our proposed swapping (high to low temperature), final result after ZoomOut, ZoomOut on only 20 input matches. There are some discontinuities in the final solution due to very different shapes but our dense solution helps guide ZoomOut a lot in this difficult case.

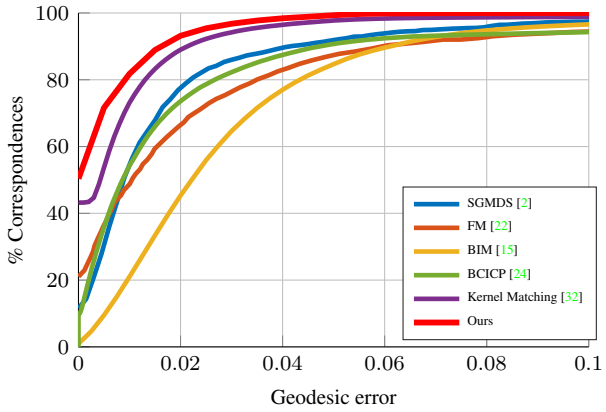


Figure 6: Correspondence accuracy on the TOSCA dataset. We compare against Functional maps, Blended Intrinsic Maps, BCICP, Kernel Matching and SGMDS.

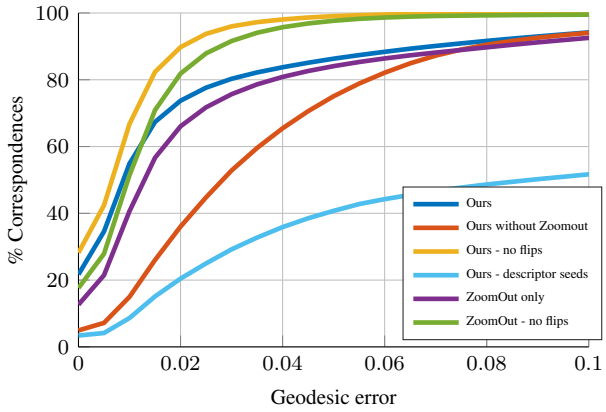


Figure 7: Correspondence accuracy on the FAUST registrations. We test the impact of ZoomOut post-processing on our method. Without any (red) our method includes a lot of outliers in the last iterations. Both ZoomOut and our method are purely intrinsic and cannot separate intrinsic symmetry flips. The yellow and green curves show the results where symmetry mistakes were excluded.

See Figure 8 for our results. We have used the seed generator with $k = 3$ and $m = 4$ in all evaluations. Some exemplary matchings produced by our algorithm can be seen and are briefly discussed in Figure 3 and Figure 4.

4.2. Ablation Study

We test the influence of different parts of our framework on the FAUST registrations [6]. In particular, we show how removing the seed generator, the swapping steps and post-processing with ZoomOut [19] changes the result.

Seed Generator. Our seed generator produces a sparse initial seed for the framework, usually four matches. We replace these four matches with the top four similar matches according to Heat Kernel Signature (HKS) [30] and SHOT [31]. As seen in Figure 7 the result is considerably worse. A possible reason is that while the seed generator ensures that the matches are well distributed *and* accurate, the descriptor matches are only accurate but sometimes cluttered together which is not beneficial for our insertion process.

Swapping. The swapping operations are performed with a certain probability depending on the set temperature, see Eq. (3). We test the influence of swapping and the temperature on our result. For that, we repeat the evaluation on the Faust registration without any swapping, swapping at zero temperature (basically a greedy improvement step) and with high temperature. The results are shown in Table 1.

Postprocessing. We use ZoomOut [19] to post-process our results. The main reason is that during the last few insertion operations (when the majority of the shape is al-

	No swapping	Greedy	Ours
w/o ZoomOut	0.047	0.042	**0.041**
with ZoomOut	0.042	0.0405	**0.038**

Table 1: Comparison of swapping strategies on the FAUST registration dataset.

ready filled) the optimal places according to Eq. (8) might already be filled and a suboptimal choice has to be made. See Figure 11 in the Appendix to see how the performance drops when adding the last 10% of matches. Clusters of wrong matches can only be rearranged with an unreasonable amount of swapping operations but are easily fixed by a few iterations of ZoomOut. See Figure 4. To show that our framework does not solely rely on ZoomOut, we show ZoomOut directly applied on the initial matches from the seed generator and an example of non-isometric shapes where ZoomOut makes large mistakes without a really dense input from our framework (see Figure 5).

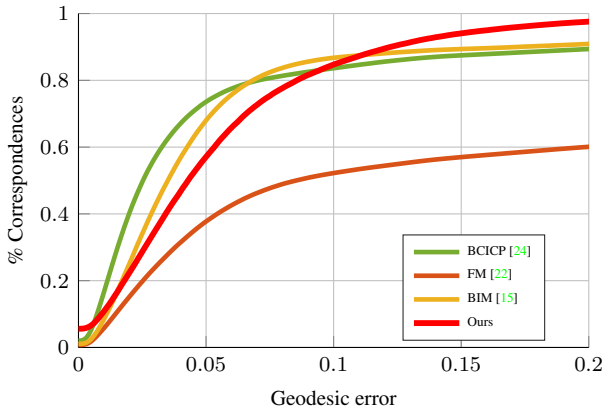


Figure 8: Correspondence accuracy on the SHREC'19 Connectivity dataset. We compare against Blended Intrinsic Map, Functional Maps and BCICP. Kernel Matching and SGMDS do not perform well on this kind of data. We reach full performance (100%) long before the competitors.

5. Conclusion

We presented a probabilistic algorithm that can solve the 3D non-rigid correspondence problem on high-resolution meshes and, to the best of our knowledge, the first application of Simulated Annealing to QAPs on 3D meshes. Our algorithm includes a novel seed generator that can produce a very sparse initial correspondence but can also process a given set of sparse input matches. Based on the sparse initialization we add new matches by optimizing the embedding on the surface with respect to the matches in the current solution. While this would usually requires the entire geodesic distance matrix, we utilize locality sensitive hashing to make the operations efficient. The intermediate and final solutions are refined by applying Simulated Annealing, a local and probabilistic optimization that is also fast and can recover from local optima. Finally, we post-process our results with ZoomOut to remove outliers that appear in the later insertion operations. Our experiments outperform several recent 3D correspondence methods and even scale to shapes with more than $100k$ vertices. Additionally, we show that we can handle shapes with very different resolution and vertex density due to our sampling strategy, a case in which methods that optimize Eq. (4) normally return clustered solutions.

Figure 9: (Left) Example of a failure case of our method on the FAUST registrations. The front and back side are flipped. (Right) Failure when initializing our method with seeds from descriptor similarity. The upper and lower part of the body are not oriented the same way. This can happen if the seeds are not well distributed on the surface.

Figure 10: (Left) Example of our results on a FAUST scan. The shapes have over $100k$ vertices, a size that is normally unfeasible for QAPs. (Right) Texture transfer between two cats of the TOSCA dataset.

Acknowledgement

We would like to thank Matthias Vestner for useful discussions. We gracefully acknowledge the support of the Collaborative Research Center SFB-TRR 109 'Discretization in Geometry and Dynamics'.

References

[1] Y. Aflalo, A. Bronstein, and R. Kimmel. On convex relaxations of graph isomorphism. In *Proceedings of the National Academy of Sciences PNAS*, 2015.

[2] Y. Aflalo, A. Dubrovina, and R. Kimmel. Spectral generalized multi-dimensional scaling. *IJCV*, 118(3):380–392, 2016.

[3] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[4] F. Bernard, Z. K. Suri, and C. Theobalt. MINA: convex mixed-integer programming for non-rigid shape alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[5] F. Bernard, C. Theobalt, and M. Moeller. DS*: tighter lifting-free convex relaxations for quadratic matching problems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] F. Bogo, J. Romero, M. Loper, and M. J. Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014.

[7] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[8] O. Burghard and R. Klein. Efficient lifted relaxations of the quadratic assignment problem. In *Vision, Modeling & Visualization (VMV)*, 2017.

[9] N. Dym, H. Maron, and Y. Lipman. DS++: a flexible, scalable and provably tight relaxation for matching problems. *ACM Transactions on Graphics (ToG)*, 36(6), 2017.

[10] M. Edelstein, D. Ezuz, and M. Ben-Chen. Enigma: Evolutionary non-isometric geometry matching. *Transactions on Graphics (Proc. SIGGRAPH)*, 2020.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. Very Large Database (VLDB*, 1999.

[12] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(4), 1996.

[13] I. Kezurer, S. Kovalsky, R. Basri, and Y. Lipman. Tight relaxations of quadratic matching. *Computer Graphics Forum (CGF)*, 34(5), 2015.

[14] V. G. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, volume 30, page 79. ACM, 2011.

[15] V. G. Kim, Y. Lipman, and T. A. Funkhouser. Blended intrinsic maps. *Trans. Graphics*, 30(4), 2011.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.

[17] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference on Computer Vision (ICCV)*, 2005.

[18] S. Melzi, R. Marin, E. Rodolà, U. Castellani, J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov. Matching Humans with Different Connectivity. In *Eurographics Workshop on 3D Object Retrieval*, 2019.

[19] S. Melzi, J. Ren, E. Rodolà, A. Sharma, P. Wonka, and M. Ovsjanikov. Zoomout: Spectral upsampling for efficient shape correspondence. *Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2019.

[20] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Tellar, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), 1953.

[21] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[22] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. Functional maps: a flexible representation of maps between shapes. *Trans. Graphics*, 31(4):30, 2012.

[23] M. Pincus. A monte-carlo method for the approximate solution of certain types of constrained optimization problems. *Operation Research*, 18(6), 1970.

[24] J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov. Continuous and orientation-preserving correspondence via functinal maps. *Trans. Graphics*, 37(6), 2018.

[25] Y. Sahillioglu. A genetic isometric shape correspondence algorithm with adaptive sampling. *Transactions on Graphics*, 37(5), 2018.

[26] Y. Sahillioglu. Recent advances in shape correspondence. *The Visual Computer*, 2019.

[27] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 1976.

[28] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis. Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (casp13). *PROTEINS*, 87(12), 2019.

[29] P. Starink and E. Backer. Finding point correspondences using simulated annealing. *Pattern Recognition*, 28(2), 1995.

[30] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature-based on heat diffusion. *Computer Graphics Forum*, 2009.

[31] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

[32] M. Vestner, Z. Lähner, A. Boyarski, O. Litany, R. Slossberg, T. Remez, E. Rodola, A. Bronstein, M. Bronstein, R. Kimmel, et al. Efficient deformable shape correspondence via kernel matching. In *2017 International Conference on 3D Vision (3DV)*, pages 517–526. IEEE, 2017.

[33] M. Vestner, R. Litman, E. Rodolà, A. Bronstein, and D. Cremers. Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Appendix for Simulated Annealing for 3D Shape Correspondence

A. Quality during Insertion Process

We show the evolution of the error curve during the insertion process. When inserting the last 10% of points it is very likely that the immediate neighborhood of the best potential match is already taken and an unfavorable point has to be picked. This creates a lot of outliers in the end. We stop the insertion process at 70% to circumvent this. There is no significant difference in the final result between filling 70% or 90%, but a noticeable drop for 100%.

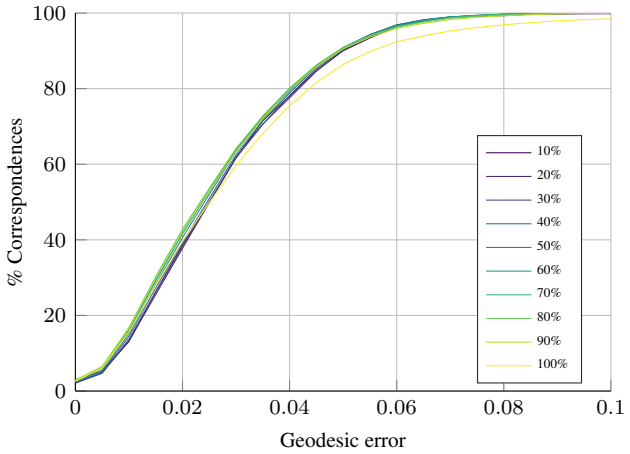
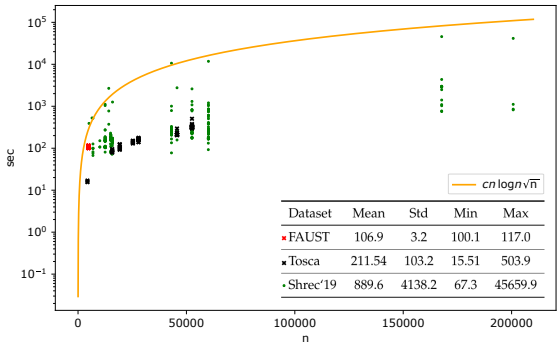


Figure 11: Evolution of the correspondence quality for one FAUST pair with percentage of points matched. Due to inserting points w.r.t. (8) the error stays mostly constant during insertion but during the last 10% the correct potential matched might already be taken so a lot of outliers appear.

B. Computational Complexity

Figure 12 gives a summary of the runtimes for the different datasets (not including the distance caluclations). We also give a short analysis of the asymptotic time complexity for the different parts of the algorithm:

- **Distance calculation:** computing the distances between a single basis point and all other points can be done in $O(E + n\log(n))$ using Dijksra's algorithm with Fibonacci heaps, Here, E is the number of edges of the mesh, which is in $O(n)$. For all basis points, the time is bounded by $O(\sqrt{n}n\log(n)))$, as there are $O(\sqrt{n})$ basis points. Ordering the basis points according to their distance for each point can be done in $O(n\log\sqrt{n}\sqrt{n})$.



Dataset	Mean	Std	Min	Max
FAUST	106.9	3.2	100.1	117.0
Tosca	211.54	103.2	15.51	503.9
Shrec'19	889.6	4138.2	67.3	45659.9

Figure 12: Runtimes on several datasets. Mean, min and max are in seconds. n is the maximum number of vertices. $c = 0.0001$ was chosen for the reference plot.

- **First insertion phase:** this phase is only repeated for 250 points and each insertion in this stage can be done in $O(n\log(n)\sqrt{n})$. The same is true for the all swapping operations done in this stage.
- **Second insertion phase:** we follow the structure of the algorithm as presented in Subsection 3.3.2 and give some further details on each step:
  1. $\mathrm{ME}_{\mathcal{X}}(x)$ contains at most $O(\sqrt{n})$ elements and is then filtered to contain only a maximum number of elements that are closest to x.
  2. We stop expanding the set $\hat{E}(x)$ after a fixed number of points independent of n.
  3. There are at most $O(\sqrt{n})$ increases of m until we find points to add to $\hat{E}(x)$.
  4. Evaluating the score can be done in $O(\sqrt{n})$ by the bound on the size of $\mathrm{ME}_{\mathcal{X}}(x)$ and the approximations of $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ with locality sensitive hashing in time $O(\sqrt{n})$. For all points, this gives $O(n\sqrt{n})$.
- **Final swapping:** the identifaction of the refinement candidates is in $O(n\sqrt{n})$, as the size of the set $\mathrm{ME}_{\mathcal{X}}(x)$ is bound and we can approximate pairwise distances in $O(\sqrt{n})$ using locality sensitive hashing. Checking if a transposition of two points improves the their overall contribution according to Eq. (9) can be done in $O(\sqrt{n})$ by the same argumentation.

Overall, we can therefore bound the running time by $O(n\log(n)\sqrt{n})$.

C. Locality Sensitive Hashing : Geodesic distances on the surface of a unit sphere

In this section, we want to demonstrate how well locality sensitive hashing can be used to approximate distances on the surface S of a unit sphere. For this, we need to generate uniformly distributed points on the sphere's surface S. A single point's position on S is fully characterized by its azimuthal angle $\theta \in [0, 2\pi]$ and its polar angle $\phi \in [0, \pi]$. See Figure 13 for reference. It is easy to check that the density f of the desired distribution of θ and ϕ is given by $f(\theta, \phi) = \frac{1}{4}\pi \sin(\phi)$, which can be marginalized into

$$f_\theta(\theta) = \tfrac{1}{2}\pi \quad \text{and} \quad f_\phi(\phi) = \frac{\sin(\phi)}{2}. \tag{10}$$

Consider two points v and w on S. Without loss of generality, we may assume that v is the north pole of the sphere, i.e. $v = (0, 0)$ and the point w is given by $(0, \phi)$ for some polar angle ϕ. Then, the distance between v and w is exactly $d(v, w) = \phi$. Let a new point u be given by the angles $(\hat{\theta}, \hat{\phi})$. This point acts as a basis point, and we want to use the distances $d(u, v)$ and $d(u, w)$ to approximate the distance $d(v, w)$. We know that $d(u, v) = \hat{\phi}$ holds. To get the distance between w and u we need the (polar) angle ψ between them, which can be obtained from the dot product

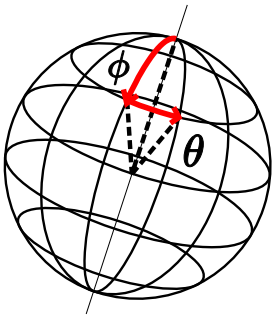


Figure 13: Sphere coordinates: polar angle ϕ and azimuthal angle θ.

$$w \cdot u = \cos(\psi) = \cos(\phi)\cos(\hat{\phi}) + \sin(\phi)\sin(\hat{\phi})\cos(\hat{\theta})$$

and then $d(w, u) = \psi$. For the geodesic distance, the triangle inequality holds. Therefore, we can write

$$\phi \leq \hat{\phi} + \cos^{-1}(\cos(\phi)\cos(\hat{\phi}) + \sin(\phi)\sin(\hat{\phi})\cos(\hat{\theta})).$$

Note that if $\hat{\theta} = 0$, i.e. u lies on the shortest line connecting w and v, then

$$\begin{aligned}
&\cos^{-1}(\cos(\phi)\cos(\hat{\phi}) + \sin(\phi)\sin(\hat{\phi})\cos(0)) \\
=&\cos^{-1}(\cos(\phi)\cos(-\hat{\phi}) - \sin(\phi)\sin(-\hat{\phi})) \\
=&\cos^{-1}(\cos(\phi - \hat{\phi})) = \phi - \hat{\phi}
\end{aligned}$$

by the addition theorem for cosine and thus the inequality is tight. Now suppose we sample u according to the density $f(\theta, \phi)$, then the random variable

$$R(\hat{\phi}, \hat{\theta}) := \cos^{-1}(\cos(\phi)\cos(\hat{\phi}) + \sin(\phi)\sin(\hat{\phi})\cos(\hat{\theta}))$$

gives the distance $d(u, w)$. We can use $R(\hat{\phi}, \hat{\theta})$ to get an upper and a lower bound on ϕ from the triangle inequality and the reverse triangle inequality, i.e.

$$L_{\text{upper}}(\phi, \hat{\phi}, \hat{\theta}) := R(\hat{\phi}, \hat{\theta}) + \hat{\phi} \geq \phi$$

and

$$L_{\text{lower}}(\phi, \hat{\phi}, \hat{\theta}) := |\hat{\phi} - R(\hat{\phi}, \hat{\theta})| \leq \phi.$$

We want to assess how close these bounds are to the actual distance ϕ between v and w w.r.t. random sampling of u. More specifically, we are interested in

$$\Pr[L_{\text{upper}}(\phi, \hat{\phi}, \hat{\theta}) - \phi \leq \tau | \phi]$$

and

$$\Pr[\phi - L_{\text{lower}}(\phi, \hat{\phi}, \hat{\theta}) \leq \tau | \phi]$$

for $\tau > 0$. We evaluate these quantities numerically as seen in Figure 14 for $\tau = 0.001$. As seen in this figure, the lower bound is good for small values of $d(v, w)$, or small ϕ, and bad for large $d(v, w)$, whereas it is exactly the opposite situation for the upper bound.

Now consider upper and lower bounds that depend on a set of multiple points Z with $|Z| = n$ sampled from the surface of the sphere

$$L^*_{\text{upper}}(\phi, Z) := \min_{z \in Z} R(\phi_z, \theta_z) + \phi_z$$

and

$$L^*_{\text{lower}}(\phi, Z) := \max_{z \in Z} |\phi_z - R(\phi_z, \theta_z)|.$$

Then

$$\begin{aligned}
&\Pr[L^*_{\text{upper}}(\phi, Z) - \phi \leq \tau | \phi] \\
=&1 - \prod_{z \in Z} (1 - \Pr[L_{\text{upper}}(\phi, \phi_z, \theta_z) - \phi \leq \tau | \phi])
\end{aligned}$$

and analogously for the lower bound. For $|Z| = 500$ numerical evaluations are shown in Figure 15. Assuming that we know which of the two bounds performs better, we can approximate $d(v, w)$ in the interval $[d(v, w) - \tau, d(v, w) + \tau]$ with probabilities at least as high as shown in Figure 16.

D. Additional Quantitative Experiments

We ran additional experiments evaluating our method against [25]. The genetic algorithm also produces a sparse

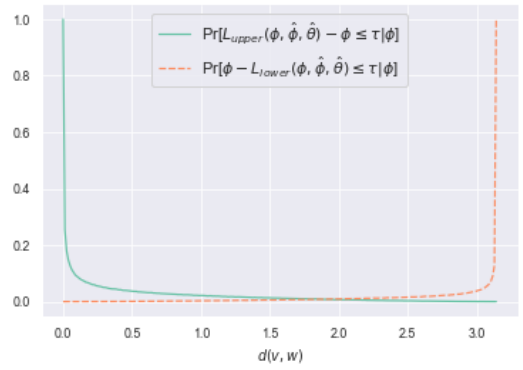


Figure 14: $\Pr[L_{\text{upper}}(\phi, \hat{\phi}, \hat{\theta}) - \phi \leq \tau|\phi]$ (upper bound) and $\Pr[\phi - L_{\text{lower}}(\phi, \hat{\phi}, \hat{\theta}) \leq \tau|\phi]$ (lower bound) with $\tau = 0.001$.

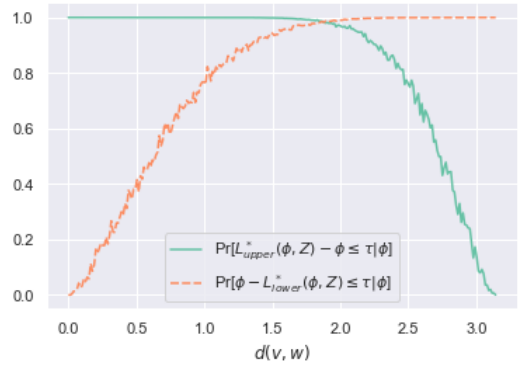


Figure 15: $\Pr[L^*_{\text{upper}}(\phi, Z) - \phi \leq \tau|\phi]$ (upper bound) and $\Pr[\phi - L^*_{\text{lower}}(\phi, Z) \leq \tau|\phi]$ (lower bound) with $\tau = 0.001$.

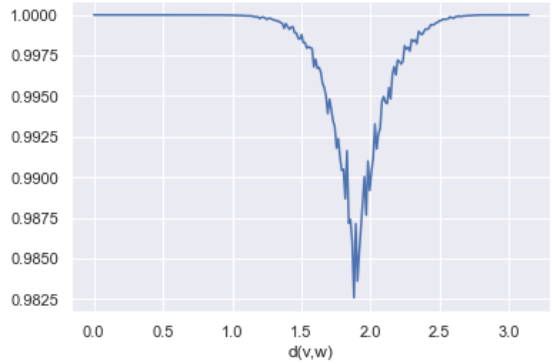


Figure 16: Maximum of $\Pr[\phi - L^*_{\text{lower}}(\phi, Z) \leq \tau|\phi]$ and $\Pr[L^*_{\text{upper}}(\phi, Z) - \phi \leq \tau|\phi])$ with $\tau = 0.001$.

set of correspondences (here 100) and for fairness we compare against the first 100 correspondences of our algorithm which are usually the most accurate. See Figure 17 for the results.

Additionally, we test how robust our method performs

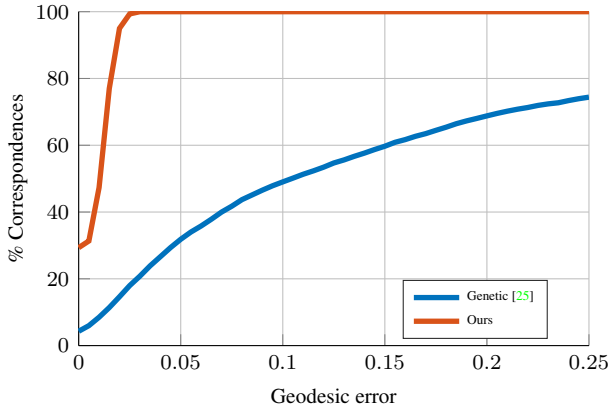


Figure 17: Comparison of our method to [25] on 100 matches on TOSCA. [25] only produces sparse correspondence, we choose our first 100 correspondences for a fair comparison.

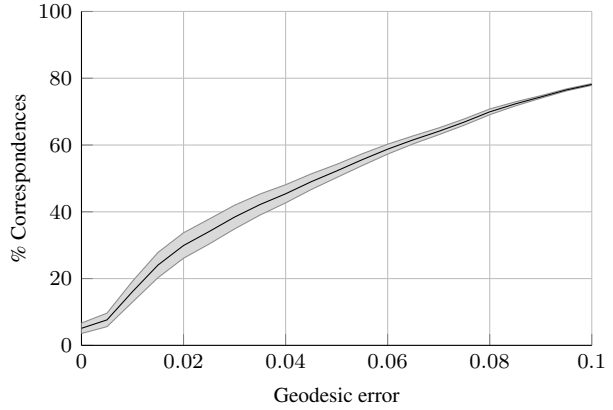


Figure 18: Robustness of our method. We ran the proposed algorithm on the same pair of FAUST shapes 50 times and give the mean and standard deviation. This is a failure case of our method where the front and back are swapped, which is found consistently throughout all runs.

when repeating the same experiment multiple times due to the probabilistic nature of our results. Figure 18 shows that the results are very reliable.