Bending Graphs: Hierarchical Shape Matching using Gated Optimal Transport

Mahdi Saleh¹  Shun-Cheng Wu¹⁺  Luca Cosmo²,³
Nassir Navab¹  Benjamin Busam¹  Federico Tombari¹,⁴

¹Technische Universität München  ²Ca’ Foscari University of Venice, Italy  ³USI University of Lugano, Switzerland  ⁴Google

Abstract

Shape matching has been a long-studied problem for the computer graphics and vision community. The objective is to predict a dense correspondence between meshes that have a certain degree of deformation. Existing methods either consider the local description of sampled points or discover correspondences based on global shape information. In this work, we investigate a hierarchical learning design, to which we incorporate local patch-level information and global shape-level structures. This flexible representation enables correspondence prediction and provides rich features for the matching stage. Finally, we propose a novel optimal transport solver by recurrently updating features on non-confident nodes to learn globally consistent correspondences between the shapes. Our results on publicly available datasets suggest robust performance in presence of severe deformations without the need of extensive training or refinement.

1. Introduction

Deformable surfaces have been studied extensively by both computer graphics and computer vision communities. Dense correspondence estimation is closely linked to applications such as reconstruction and human pose estimation. In a standard formulation, we want to find a function \( f : G \rightarrow H \) where \( f \) is a mapping from a shape \( G \) to another shape \( H \). A shape is usually discretized in a triangle mesh that can be expressed in the form of a graph \( G = (V, E) \) constituted by vertices \( V \) with associated 3D coordinates and edges \( E \). Among the most successful optimization methods to find correspondences between deformable shapes, there is the Functional Map framework [41]. The pointwise correspondence between two shapes is expressed as a linear map between the functional bases (i.e. eigenfunctions of the Laplace Beltrami Operator) defined on the shapes.

As a first work to pair the functional map framework with deep neural networks on deformable shapes, deep functional maps [32] present a pipeline to seek a correspondence using as input SHOT [60] handcrafted descriptors on sampled points. Despite building on top of point cloud features, FMNet [32] finds correspondences densely on deformable shapes. Recent works [22, 61] also use learned point cloud features [59] to describe local regions. Other methods also suggest integrating spectral manifold wavelets [26] or iterative spectral upsampling into functional maps [37]. While these methods investigate dense correspondences, many require numerous initial keypoints or are sensitive to initial sparse matches.
Though meshes are commonly represented and stored as undirected lattice graphs, limited works explore graph deep learning frameworks to extract features from meshes [21, 39, 64]. Litany et al. use graph convolutional auto-encoders for the task of shape completion and Zhou et al. use graphs for image-based deformable matching. Graph neural networks (GNNs) are successfully applied to point clouds [31, 54, 58, 68] across many tasks. This work presents a graph-based descriptor designed for meshes, capturing local surface structures by constructing edges given a mesh lattice graph. GNNs on meshes are a discretization of spectral convolutions on manifold representations [7] and therefore constitute a powerful tool to capture local deformations.

Moreover, GNNs have proven to be a great framework to construct hierarchical representations [9, 12, 18, 66, 69, 72]. During recent years GNNs have been used in many computer vision tasks, from scene graph generation [18] to holistic representation learning [76], enabling relationship definition [29] and information exchange and propagation [69, 70]. Here we incorporate a high-level shape graph to represent and interconnect shape regions, as shown in figure 1. Each shape region describes the surrounding geometry by the local graph. Using such hierarchical graphs, we can process 3D meshes efficiently and learn a rich holistic shape representation, which is able to capture local details and capture neighbor geometry information.

We address the problem of 3D deformable shape matching as an optimal transport problem. As a classic problem, optimal transport is gaining popularity in the fields of feature matching and registration [43, 47, 55, 73], where differentiable Sinkhorn algorithm [42, 55, 73] performs well with learning-based feature matching [10]. The Sinkhorn solver favors putative correspondences by iteratively applying softmax and does not work well in soft correspondence problems or when an exact matching is missing. The coarse deformable setting with sampled graph seeds does not guarantee putative hard matches. To deal with such an issue, we propose a new strategy using Gated Recurrent Units (GRUs) to propagate features using matching confidence in our shape graph leading to a more robust optimal transport solution.

In summary, our contributions are as follows:

- We propose a novel Gated Optimal Transport (GOT) module incorporating attention-based feature propagation into the Sinkhorn algorithm.

## 2. Related Work

In this section, we will briefly review the related works in the area of 3D feature description, shape registration, and dense correspondence methods which are either related to the proposed method or are compared in the evaluation section.

**Point descriptors** Being able to produce a robust and descriptive point descriptor is at the core of many 3D Computer Vision tasks, especially when corresponding points have to be found between different objects. Unlike 2D image features, 3D features need to handle additional ambiguity introduced in the third dimension. To handle this ambiguity, some methods depend on a local reference frame, such as SHOT [60], RoPS [23] and TriSi [24], some others rely on pair-wise point description, such as PFH [53] and FPFH [52]. For non rigid meshes, spectral descriptors are often used [3, 8, 13] given their invariance to (near-)isometric deformations. Later, data-driven approaches are proposed to compress hand-crafted features into a compact yet informative representation [27] or to learn a more robust feature description directly from point clouds [15, 48]. PointNet [48] is the first approach that directly outputs feature description using points with a permutation invariant pooling, but it fails to capture geometric details and ignores neighborhood information. PointNet++ [49] is proposed to solve these issues by using multiple PointNets hierarchically to capture local details. 3DMatch [74] and Perfect Match [20] use voxel representation to compute feature descriptors with a 3D convolutional neural network, which is able to grasp the connection between voxels. The usage of 3D convolutions, however, significantly increases memory consumption and therefore limits the usability of these methods. Another line of works [5, 54, 59] benefits from the point representation while using GNNs to incorporate the information from surface and manifolds. KPConv [59] defines 3D local filters with a set of kernel points which allow efficient and flexible point description. Graphite [54] uses a Graph Neural Network to describe local patches and predict keypoints for point cloud registration.

**Shape registration** Shape registration aims to find the deformation between two given geometric shapes. A classic solution is ICP [2] which estimates a transformation matrix to minimize the distance between closest points of the given shapes iteratively. Several methods are proposed to improve the matching performance of ICP, such as GoICP [71]. PointNetLK [1] exploits learned point features [48] with the Lucas-Kanade algorithm [34] to tackle the registration problem. Deep Closest Point [67] incorporates the popular attention mechanism in the correspondence finding process.
to estimate the transformation. Other approaches reinforce correspondence prediction leveraging additional sources of knowledge, such as parametric human [4,33,44,45] or animal [77–79] models.

Dense correspondence In the correspondence problem, the goal is to find a map relating the points of an input shape to the points of a second one, possibly undergoing some deformation. Most of the methods for non-rigid shape matching [28,41,46,50] rely on intrinsic properties of the input shapes. A common drawback of these methods is that correspondences are poorly localized. Moreover, intrinsic properties are unaware of isometrics in the input shapes, resulting in misplaced matches. Several methods have been proposed to solve these issues [19,37,65], but still, the performance deteriorates in a more challenging scenario [36]. Unlike previous works that rely on axiomatic descriptors [40,50], recent methods focus on learning an optimal descriptor to have a better functional map estimation [14,16,25,32,35,51,56,61]. Marin et al. [35] propose a two-stage method to estimate an optimal linear transformation by the use of an invariant embedding network combined with a probe function network. Trappolini et al. [61] propose to estimate the transformation between two input point clouds with an auto-encoder architecture and a transformer network [63]. Groueix et al. [22] learn to estimate the transformation between two given shapes using a neural network.

3. Bending Graph Methodology

In this section, we illustrate our methodology. The overview of our method can be seen in figure 2. As a standard dense matching pipeline between two shape meshes, we first focus on each input mesh and process them independently. We start by explaining our local graph definition and how the descriptor is configured in section 3.1. Next, in 3.2, we discuss how the higher-level shape graph is constructed to represent the shape structure. In section 3.3, we focus on the matching problem where we present a GOT unit to predict the correspondences. At the end of this section in 3.4 we explain the training and loss formulations.

3.1. Local Graphs

Each of the input shapes is given in the form of a mesh. A mesh $M$ consists of vertices $V \in \mathbb{R}^3$, edges $E$, and $M = (V, E)$ which form a regular tiling. This definition is very close to that of an undirected graph, where a node represents vertices. To define local structures, we focus on sampled local graphs $G_i$, $i \in \mathbb{N}$. To sample $N$ graphs, we first apply furthest point sampling (FPS) on mesh vertices in Euclidean space. We form graphs $G_i$ around each of the sampled vertices $v_j \in V$. In order to construct the graph based on the mesh structure, we define edges using the local Dijkstra algorithm. We cut a sub-graph from a mesh graph where the vertices are below a certain shortest path value $d_{cut}$:

$$V_i = \{v_j \in G \mid d(v_i, v_j) < d_{cut}\},$$

where $d(v_i, v_j)$ defines the shortest paths between the vertices $v_i$ and $v_j$. Each node is associated by its coordinate in a local reference frame, $v_j = (x_j, y_j, z_j)$. Edges link to nearby vertices. As an alternative to KNN [5], we consider the unit ball with radius $r$ to maintain metrics around each node’s positional coordinates and connect them to other nodes when their distance falls below $r$. To induce some weight to the edges, we attribute a scalar value $e_{i,j} \in \mathbb{R}$ to it as the vertex-to-vertex distance.

Local Mesh Description We now introduce our graph neural network (GNN) architecture. We build graphs consisting of edges and nodes to represent a local mesh. In contrast to classic and learned point cloud descriptors, which only define a certain number of points [15,54,60], here we want to describe dynamically-sized graphs based on how the local mesh is structured.

The node and edge features, connections are fed into a GNN architecture to estimate a descriptor under variable deformations. Output descriptors should be invariant to permutations, meshing variations, and applied deformations. Following Graphite [53] we use a topology adaptive graph (TAG) [17] convolutional operator, which combines node and edge feature propagation inside the graph. We use multiple layers of TAG function with increasing hops (K=1,2,3). Hops define how many nodes the information propagates inside the graph and provides a multi-scale feature extraction by message passing [54]. Every graph convolutional module takes into account the adjacency matrix $A \in \mathbb{R}^{n \times n}$ and its diagonal degree matrix $D \in \mathbb{R}^{n \times n}$ to propagate node features across the graph. As in [17], we update node-level information $v_j'$ by propagating features as follows:

$$v_j' = \sum_{k=0}^{K} D^{-1/2} A^k D^{-1/2} v_j \Theta_k.$$  

Following the multi-scale message passing in each local graph, we apply a global max pooling operator to extract the local feature descriptor $D_i$. The features are then passed to a Multi-Layer Perception (MLP) and normalized to bring the final feature on a unit sphere. The local features $D_i$, $i \in N$ are trained as a triplet to encourage more robustness of features as further explained in 3.4.

3.2. Shape Graphs

In this section, we describe our higher-level shape graph. The intention of the shape graph is to build a coarse-level
representation of the shape and facilitate feature propagation between the local graphs. In this way, we can incorporate more global features into the node descriptor and provide better grounds for correspondence matching.

A shape graph $S$ consists of $N$ nodes, where each node is associated with a local descriptor vector $D_i \in \mathbb{R}^d$ and seed point $v_i \in \mathbb{R}^3$, initially sampled in 3.1 using FPS. Furthermore, we define edges using a unit ball in the geodesic space. Keeping the shape into a unit ball, we further define a shape radius $r_{shape}$ to construct edges between shape nodes. Similarly to local graphs, we add an edge weight based on the geodesic distance of the nodes. Figure 4 shows sample shape graphs generated from the MPI FAUST [4] dataset. The graph structure remains somewhat similar in presence of local deformations.

**Positional Encoding** In order to aggregate the node information, we need to present the position in a high-dimensional embedding. This process is commonly performed using layers of MLP similar to [48]. The absolute input position $v_i$ should be encoded to a feature of size $d$. To capture the fine details and inspired by NeRF positional encoding [38], we use a Fourier feature mapping [57]. We map input coordinates into a higher dimensional Fourier space before passing them through the network with

$$
\gamma(v_i) = [\ldots, \cos(2\pi \sigma^i/m v_i), \sin(2\pi \sigma^i/m v_i), \ldots]^T.
$$

(3)

We extract $m - 1$ log-linear spaced frequencies for each positional element. Afterwards, we pass them to a shallow MLP network (MLP$_p$) to create an embedding of size $d$. The embedded positional encoding is added to $D_i$, followed by another MLP network (MLP$_p$) to obtain the node feature $f_i$. The entire process can be expressed as follows:

$$
f_i = \text{MLP}_p (D_i + \text{MLP}_e (\gamma(v_i))).
$$

(4)

### 3.3. Gated Optimal Transport

To find the matching between two 3D meshes, we formulate the task as a linear assignment problem, which tries to maximize the total score of $\sum_{i,k} C_{i,k} P_{i,k}$ with an assignment $P$ and a score matrix $C \in \mathbb{R}^{M \times N}$. As in [55], the score for each match is calculated as a simple inner product of their descriptors:

$$
C_{i,k} = \langle f_i^A, f_k^B \rangle, \forall (i,k) \in A \times B,
$$

(5)

where $\langle \cdot, \cdot \rangle$ is the inner product, $A$ and $B$ represent a source and a target.

The above optimization problem can be efficiently solved with the Sinkhorn algorithm. This algorithm estimates bipartite joint probabilities by iteratively normalizing $\exp(C)$ along rows and columns with a given number of iterations. Since the entire operation is differentiable, the whole process can be trained end-to-end by minimizing the negative log-likelihood of $P$. The loss formulation will be explained in 3.4.

**Gated Feature Propagation** Although solving the optimal transport with Sinkhorn provides a fast and direct way to solve the bipartite matching, it focuses on finding the exact match on each given input without considering the mesh topology. Two nearby vertices on a mesh may be assigned to different regions after the matching process. To tackle this issue, we propose to utilize the confidence in the Sinkhorn operation to effectively propagate the feature with high confidence to its nearby low confidence features along with the shape graph, which we call GOT. This enforces the nearby point features to be matched to a nearby locations on the target region, as shown in Fig. 3.

Node feature propagation on the shape graph with the confidence value estimated by the similarity score can be
considered as conditional random fields (CRFs) [30], which is typically the final optimization step in a pipeline. Previous work showed that similar processes are achievable at training time using recurrent neural networks (RNNs) [70, 75]. Inspired by these, we formulate the node feature as a mean-field approximation with the connectivity as the shape graph. Similarly, we use mean-field to perform approximate inference. The feature of a node on each update step can then be formulated as:

\[ h_i^t = Q \left( h_i^{t-1}, \max_{l \in N(i)} (w_l h_j^{t-1}) \right), \]  

(6)

where \( h_i^t \) is the hidden state of node \( i \) at time \( t \), \( N(i) \) is all the neighbors of node \( i \), \( Q \) is an RNN model, for which we used a GRU [11] unit and \( w_l \) is the weight of node \( l \). The initial hidden state of each node \( h_i^0 \) is initialized as \( f_i \).

As each node may receive multiple features from its neighbors, we weight each input by the log likelihood estimated in the score matrix \( C \). The weight value of each node is estimated in the Sinkhorn assignment, which can be obtained from the score matrix \( C \) by:

\[ w_i = \max_l (C_{i,l}), \quad w_j = \max_l (C_{i,l}). \]  

(7)

The confidence values are kept in log space in the message passing process, since we found that it results in better performance. The usage of the shape graph allows us to control the number of hop neighbors to be considered in the message passing operation. Furthermore, the entire GOT process can also be applied iteratively to reinforce the optimal transport solution.

### 3.4. Loss Functions

In this section, we describe our training strategy and loss functions. As expressed initially in section 3.1, we are given pairs of shapes as meshes, and we suggest a mapping for each vertex in \( V_a \) to a vertex in \( V_b \). To enable efficient and coarse-level matching, we sample \( N \) points from each shape. To train our descriptor using a triplet loss, we also require a negative sample. As a common practice in metric learning and to increase feature distinctiveness, we add a hard-negative sample from shape \( B \). To mine a negative sample, we use the Dijkstra algorithm to draw a graph in the vicinity of the target seed vertex.

**Local descriptor** Learned descriptors or features are trained by self-supervision of pose or deformation variations [15, 22, 54]. The estimated feature vector can be trained using contrastive or metric learning. We make sure a feature pair \( D(G_a) \) and \( D(G_b) \), describing respectively the local graphs of \( G_a \) and \( G_b \), are closed in a feature space. In a triplet setting, we furthermore use negative \( D(G_n) \) to describe negative graph sample as explained below:

\[ L_D = \max \left( \sum_{i=1}^{N} \text{dist}(G_{a,i}, G_{b,i}) - \text{dist}(G_{a,j}, G_{a,i}) + \alpha, 0 \right) \]  

(8)

where \( \text{dist}(G_{a,i}, G_{b,i}) = ||D(G_{a,i}) - D(G_{b,i})||_2 \) and \( \alpha \) defines a small margin to reduce zero values in the loss.

**Matching** Following our GOT module, described in 3.3, we yield a cost matrix of size \( N \times N \), where each index \( C_{i,l} \) defines the softmax confidence values. During training, we have a bipartite distance matrix \( M \), where each \( M_{i,l} \) defines the shortest path in the graph from node \( i \) from shape \( A \) to node \( l \) from shape \( B \). Close matches are defined as minimum entries in the bipartite distance matrix. Our final goal is to have maximum score values on such entries.

In order to increase the log-likelihood of the entries softly based on the bipartite distance matrix, we define a weighting matrix, \( M' \) based on the matrix \( M \) where

\[ M'_{i,l} = \begin{cases} r_d M_{i,l} & \text{if } M_{i,l} \leq r_d \\ r_d & \text{otherwise} \end{cases} \]  

(9)

where \( M'_{i,l} \) defines an element of matrix \( M'_d \). We use this weight to induce some distance softly into the matching loss. Our loss is minimising the negative log-likelihood of correct matches on the nonzero elements of \( M' \).

\[ L_m = - \sum \log (v_{i,l} \cdot M'_{i,l}) \]  

(10)

**Regularization term** The predicted match for node \( i \) of \( A \) in \( B \) is where \( max_v v_{i,l} \). This is how we define our matching loss in the previous section. To regularize the predictions further we use the global shape structure to assert shape consistencies. We first apply a softpooling operator on predicted positions. This way, we can have a differentiable
operation for the loss calculation. We first calculate
\[
\hat{s}_i = \frac{1}{N} \sum_j^N C_{i,j} \cdot v_{j,j}
\]
(11)
and then define a Laplace operator \(\Delta(V)_i\) on the original
shape \(V\) with source positions \(v_i \in \mathbb{R}^3\) and predicted soft-
pool positions \(\hat{V}\) as
\[
\Delta(V)_i = \sum_{j:(i,j) \in E} |v_j - v_j|,
\]
(12)
where \(E\) is the set of shape graph edges.

Finally, we propose our regularization loss as follows:
\[
\mathcal{L}_R = \sum_{i=1}^N \Delta(\hat{V})_i - \Delta(V)_i
\]
(13)

Our total loss is a summation of the discussed loss functions:
\[
\mathcal{L}_{total} = \gamma_D \cdot \mathcal{L}_D + \gamma_M \cdot \mathcal{L}_M + \gamma_R \cdot \mathcal{L}_R
\]
(14)

4. Experiments

4.1. Training Setup

In this section, we demonstrate the performance of
our method by showing two major evaluations on popular
datasets used by the state-of-the-art methods for deformable
3D shape correspondences. The first experiment is to eval-
uation our network on the task of human shape registra-
tion (sec. 4.3), and the second one is on the task of ani-
mal shape registration (sec. 4.4). We further ablate each of
the proposed modules both qualitatively and quantitatively
(sec. 4.5).

4.2. Experimental Settings

In the first experiment, we use the FAUST dataset [4]
with the same testing split used in [35,61]. This dataset con-
tains 100 human shapes with a per-vertex correspondence
which allows us to evaluate the dense correspondence qual-
ity of our method. For training, we generate 500 samples
using the SURREAL dataset [62], which consists of human
SMPL [33] models with a set of parameters to control the
deformation and poses of the models. In the second exper-
iment, we use the animal shapes provided in TOSCA [6].
The TOSCA dataset provides several synthetic models in
different poses and classes. For testing, we consider all
pairs composed of the T pose of each class with all other
poses in the same class. For training, we generate 100 ran-
don models from SMAL [79] with a Gaussian distribution
of variance 0.15.

In all experiments, we sample 200 local graphs with far-
thest point sampling (FPS), as shown in figure 4. Note that
the sampled points on the source and target mesh may not
be in the same location. Therefore we do not have exact
and putative correspondences during inference. Therefore,
dense correspondence matching is achieved by using the
functional map method [41]. The error metric in all ex-
periments follows [28] which uses the average geodesic er-
ror. For ablation study, we further report the Bijectivity Rate
(BR) measures the percentage of bijective correspondences
between source and target (one-to-one consistency) over the
total number of correspondences.

For all the experiments, we implemented our method on
Pytorch and trained with an initial learning rate of 0.001, de-
creasing to 0.0001 after 30 epochs with ADAM optimizer.

4.3. Human Shape Registration

We compare our method with two dense correspondence
methods using an auto-encoder architecture, i.e. 3DC [22]
and SRT [61], a functional map method with learned linear
high dimensional basis, i.e. LinInv [35], and a method that
directly learns optimal descriptors with the functional map
framework, i.e. DGFM [16]. The dense correspondence re-
sult from our method is generated by first estimating the
coarse correspondences on the 200 patches sampled with
FPS, and then giving this correspondences as input to the
functional map algorithm in form of corresponding delta
functions. The result is shown in table 2. It can be seen
that our method outperforms previous methods by a sig-
nificant margin. We also report the number of parameters
used in each method. Due to the use of a lightweight GNN
module and our hierarchical design, our method needs only
100k trainable parameters, which is 0.7% used in [16] and
7.1% of the parameter used in [35]. The qualitative result is
shown in figure 5. Our method can match patches correctly
in locations that undergo significant deformations.
Figure 5. The result of dense shape matching on MPI-FAUST [4] trained on a few samples of SURREAL [62]. (a) we have a target shape to which we are finding correspondences. Coarse matches (b) show robust matched patches to the target shape. In (c) we visualize the final confidence values predicted. The maps suggest more confidence in less deformed regions. In (d) we can see our fine and dense correspondences by passing our coarse matches to the FM algorithm in form of corresponding delta functions. Finally, we have the Ground correspondence map in (e) and error map in (f) which highlights our good performance for inter-subject cases without the need for refinement.

4.4. Animal Shape Registration

To show our method can be used across different shapes, we evaluate our method on the TOSCA dataset. For this evaluation, we trained a model with 100 random samples on the horse class from SMAL and tested it on horse, cat, centaur and david from TOSCA. The results are shown in figure 6 and table 1. The dense correspondence results suggest that our method can also be applied to other shapes. The matching on unseen classes is more challenging since the class was unseen during training. Thanks to our hierarchical design, the model can generalize to unseen shapes.

4.5. Ablation Studies

To prove the functionality and performance of each proposed module, we design ablation studies as a further experiment. Table 3 contains the comparative results of the modules by looking at bijectivity rate (%) and average geodesic error.
Figure 6. The dense matching result of our model on TOSCA dataset. Our method is able to learn robust features that work on an unseen class. a) target shape. b) source shape with the predicted correspondences c) ground-truth correspondences, d) error map.

<table>
<thead>
<tr>
<th>Method</th>
<th>geodesic error parameters (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DC [22]</td>
<td>0.0776 3.9</td>
</tr>
<tr>
<td>DGFM [16]</td>
<td>0.0656 14.1</td>
</tr>
<tr>
<td>LinInv [35]</td>
<td>0.0942 1.4</td>
</tr>
<tr>
<td>SRT [61]</td>
<td>0.0513 1.7</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.0230</strong> 0.1</td>
</tr>
</tbody>
</table>

Table 2. We trained all models on SURREAL [62] models and tested them on MPI-FAUST synthetic [4]. Contrary to previous methods, we only train our model on 5% of the available dataset.

error normalized to the shape area. We follow the experiment setting in 4.3 by training our pipelines on 100 samples of SURREAL and testing on 20 samples of MPI-FAUST. Ablation #1 shows matching with Vanilla OT and only using local description (LG) does not provide reliable correspondences. By adding the shape graph (SG) in ablation #2, we observe a performance gain in terms of geodesic error. Here we use a Sinkhorn algorithm with 100 iterations. In this case, the correspondences contain a few bijective error. Adding a GOT layers in ablation study #3 yields more bijective correspondences, and improved final error. Note that the error is calculated on all the points without any outlier removal.

Next, in ablation #4 we deactivate the shape graph, which weakens our graph features. Nevertheless, compared to #1, we see slightly better performance. This ablation proves the effectiveness of our fine Fourier-based positional encoding. Experiment #5 illustrates the role of local graphs and descriptors and proves the local descriptor would be richer with global holistic knowledge. Ablation #6 shows the full pipeline. By comparing #3 and #6, we can observe less error and higher bijectivity which proves the efficacy of the regularization loss.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>LG</th>
<th>SG</th>
<th>GOT</th>
<th>Reg</th>
<th>BR</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 only local desc., w. OT</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42.1 0.58</td>
</tr>
<tr>
<td>#2 hier desc., w. OT</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>23.3 0.42</td>
</tr>
<tr>
<td>#3 w/o graph regularization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>40.0 0.19</td>
</tr>
<tr>
<td>#4 w/o shape graphs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>21.1 0.43</td>
</tr>
<tr>
<td>#5 w/o local desc.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>47.0 0.16</td>
</tr>
<tr>
<td>#6 full pipeline</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>49.6 0.13</td>
</tr>
</tbody>
</table>

Table 3. Ablation study trained on SURREAL [62] and evaluated on MPI-FAUST [4]. LG stands for activation of Local Graph unit 3.1; SG for Shape Graph module 3.2; GOT for our Recurrent Graph Propagation unit and Reg. for activation of our regularization loss. For each ablation, we provide the percentage of bijective correspondences (Bij. rate) and average geodesic error.

5. Limitation and failure cases

To learn and estimate correspondences, we suppose a pair of mesh with similar density. When the number of vertices varies, we need to reconstruct the mesh as a preprocessing stage. Moreover, in our training and experiments, we always normalize the meshes to the unit spheres. Although this is common practice in deformable registration, we do not provide a scale-invariant representation. The reason for normalization is setting a fixed radius for ball query operations across different datasets. Finally we only learn the coarse matches using our representation and rely on further non-learning refinements to provide dense matches.

6. Conclusion

In this paper, we propose Bending Graph, an end-to-end pipeline to learn deformable shapes in a hierarchical form. Hierarchical graphs can represent the shape flexibly and efficiently. With the use of Local and Shape Graphs, we learn object representation in a holistic way that can integrate into a matching pipeline. Furthermore, we look at the problem of dense matching using Optimal Transport. We propose a solution to learning-based Optimal Transport using Gated Recurrent Network. Using our representation, we can propagate and reinforce our features through the shape graphs. Finally, we demonstrate our pipeline and prove its effective design by providing robust results without the need for large-scale training and computationally. Our representation and matching framework can be used for multiple problems in computer vision and graphics.
References


[34] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Proc 7th Intl Joint Conf on Artificial Intelligence (IJ CAI), 1981. 2
[54] Mahdi Saleh, Shervin Dehghani, Benjamin Busam, Nassir Navab, and Federico Tombari. Graphite: Graph-induced feature extraction for point cloud registration. In 2020 In-


Giovanni Trappolini, Luca Cosmo, Luca Moschella, Ricardo Marin, Simone Melzi, and Emanuele Rodolà. Shape registration in the time of transformers. Advances in Neural Information Processing Systems, 34, 2021. 1, 3, 6, 8


Danfei Xu, Yue Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5410–5419, 2017. 2, 5

Jiaqi Yang, Zhiguo Cao, and Qian Zhang. A fast and robust local descriptor for 3d point cloud registration. Information Sciences, 346:163–179, 2016. 2


Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6365–6373, 2017. 3, 6