Integrating Efficient Optimal Transport and Functional Maps For Unsupervised Shape Correspondence Learning

Supplementary Material

In this supplementary, we first define some notations that are used in our main paper and supplementary in Sec. 8. We then discuss some limitations of our work and potential future directions to address them in Sec. 9. In Sec. 10, we provide detailed computation and algorithm to compute the proposed loss functions. Furthermore, we delineate the implementation details and hyperparameters used in our training process in Sec. 11. Finally, we provide additional qualitative results of our proposed approach in Sec. 12.

8. Notations

For any $d \geq 2$, we denote $\mathbb{S}^{d-1} := \{\theta \in \mathbb{R}^d \mid ||\theta||_2^2 = 1\}$ and $\mathcal{U}(\mathbb{S}^{d-1})$ as the unit hyper-sphere and its corresponding uniform distribution. We denote $\theta\sharp\mu$ as the push-forward measures of μ through the function $f : \mathbb{R}^d \to \mathbb{R}$ that is $f(x) = \theta^\top x$. Furthermore, we denote $\mathcal{P}(\mathcal{X})$ as the set of all probability measures on the set $\mathcal{X}$. For $p \geq 1$, $\mathcal{P}_p(\mathcal{X})$ is the set of all probability measures on the set $\mathcal{X}$ that have finite p-moments.

9. Limitations and discussion

Our work is the first to integrate an efficient optimal transport to functional map framework for shape correspondence, yet it is not without limitations, potentially opening new research directions. First of all, our algorithm is designed for use with clean and complete meshes. An intriguing avenue for future research would be to extend the applicability of our method to more diverse scenarios, such as dealing with partial meshes, noisy point clouds, and other forms of data representation. This expansion would enhance the versatility of our approach in handling a wider range of practical applications. Secondly, our adaptive refinement module, which utilizes an entropic regularized optimal transport for estimating the soft-feature similarity matrix, shows promise in achieving more precise refinement. However, this method is not without its drawbacks, notably a quadratic increase in memory complexity and computational demand. This presents a challenge that future research could address by developing more computationally efficient approximations, thereby making the process more feasible for larger datasets or more resource-constrained environments. Overall, these potential research directions could significantly contribute to the evolution of shape correspondence methodologies.

10. Detailed algorithms and discussion

Sliced Wasserstein distance. The unidirectional sliced Wasserstein distance version of Eq. 10 is given by:

$$\mathcal{L}_{uniSW} = (\mathbb{E}_{\theta\sim\mathcal{U}(\mathbb{S}^{d-1})}\mathrm{W}_p^p(\theta\sharp\mathcal{F}_x, \theta\sharp\hat{\mathcal{F}}_y))^{\frac{1}{p}}, \tag{15}$$

where $\hat{\mathcal{F}}_y = \hat{\Pi}_{xy}\mathcal{F}_y$. The unidirectional sliced Wasserstein distance given in Eq. 15 is computed by using L Monte Carlo samples $\theta_1, ..., \theta_L$ from the unit sphere:

$$\widehat{\mathcal{L}_{uniSW}} = \left(\frac{1}{L}\sum_{l=1}^{L}\mathrm{W}_p^p(\theta_l\sharp\mathcal{F}_x, \theta_l\sharp\hat{\mathcal{F}}_y)\right)^{\frac{1}{p}}, \tag{16}$$

where $\mathrm{W}_p^p(\theta\sharp\mathcal{F}_x, \theta\sharp\hat{\mathcal{F}}_y) = \int_0^1 |F^{-1}_{\theta\sharp\mathcal{F}_x}(z) - F^{-1}_{\theta\sharp\hat{\mathcal{F}}_y}(z)|^p dz$ denotes the closed form solution one-dimensional Wasserstein distance of two probability measures $\mathcal{F}_x$ and $\hat{\mathcal{F}}_y$. Here, $F_{\theta\sharp\mathcal{F}_x}$ and $F_{\theta\sharp\hat{\mathcal{F}}_y}$ are the cumulative distribution function (CDF) of $\theta\sharp\mathcal{F}_x$ and $\theta\sharp\hat{\mathcal{F}}_y$ respectively.

Similarly, the bidirectional sliced Wasserstein distance in Eq. 10 is also estimated by using L Monte Carlo samples $\theta_1, ..., \theta_L$ from the unit sphere:

$$\begin{aligned}\widehat{\mathcal{L}_{biSW}} = (\frac{1}{L}\sum_{l=1}^{L}[&\mathrm{W}_p^p(\theta_l\sharp\mathcal{F}_x, \theta_l\sharp\hat{\mathcal{F}}_y) \\ &+ \mathrm{W}_p^p(\theta_l\sharp\mathcal{F}_y, \theta_l\sharp\hat{\mathcal{F}}_x)])^{\frac{1}{p}},\end{aligned} \tag{17}$$

where $\hat{\mathcal{F}}_x = \hat{\Pi}_{yx}\mathcal{F}_x$ and $\hat{\mathcal{F}}_y = \hat{\Pi}_{xy}\mathcal{F}_y$. We provide a pseudo-code for computing the unidirectional and bidirectional sliced Wasserstein distance in Algorithm 1 and Algorithm 2, respectively.

Energy-based sliced Wasserstein distance. The unidirectional sliced Wasserstein distance version of Eq. 11 is defined as:

$$\mathcal{L}_{uniEBSW} = \left(\frac{\mathbb{E}_{\theta\sim\sigma_0(\theta)}[\mathrm{W}_{\theta,\mathcal{X}}w(\theta)]}{\mathbb{E}_{\theta\sim\sigma_0(\theta)}[w(\theta)]}\right)^{\frac{1}{p}}, \tag{18}$$

where we denote $\mathrm{W}_{\theta,\mathcal{X}} := \mathrm{W}_p^p(\theta\sharp\mathcal{F}_x, \theta\sharp\hat{\mathcal{F}}_y)$, $w(\theta) := \frac{\exp(\mathrm{W}_{\theta,\mathcal{X}})}{\sigma_0(\theta)}$, and $\sigma_0(\theta) \in \mathcal{P}(\mathbb{S}^{d-1})$ denotes the proposed distribution. The unidirectional energy-based sliced Wasserstein distance given in Eq. 18 can be computed via importance sampling estimator L Monte Carlo $\theta_1, ..., \theta_L$ sampled from $\sigma_0(\theta)$:

$$\widehat{\mathcal{L}_{uniEBSW}} = \left(\frac{1}{L}\sum_{l=1}^{L}[\mathrm{W}_{\theta_l,\mathcal{X}}\tilde{w}(\theta_l)]\right)^{\frac{1}{p}}, \tag{19}$$

Algorithm 1 Computational algorithm of the unidirectional SW distance

Input: Features extracted from feature extractor module $\mathcal{F}_x, \mathcal{F}_y$; $p \geq 1$; soft features similarity $\hat{\Pi}$ from Eq. 9; and the number of projections L.

Compute $\hat{\mathcal{F}}_y = \hat{\Pi}_{xy}\mathcal{F}_y$
for $l = 1$ to L **do**
  Sample $\theta_l \sim \mathcal{U}(\mathbb{S}^{d-1})$
  Compute $v_l = \mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_x, \theta_l \sharp \hat{\mathcal{F}}_y)$
end for
Compute $\widehat{\mathcal{L}_{uniSW}} = \left(\frac{1}{L}\sum_{l=1}^{L} v_l\right)^{\frac{1}{p}}$
Return: $\widehat{\mathcal{L}_{uniSW}}$

Algorithm 2 Computational algorithm of the bidirectional SW distance

Input: Features extracted from feature extractor module $\mathcal{F}_x, \mathcal{F}_y$; $p \geq 1$; soft features similarity $\hat{\Pi}$ from Eq. 9; and the number of projections L.

Compute $\hat{\mathcal{F}}_x = \hat{\Pi}_{yx}\mathcal{F}_x$ and $\hat{\mathcal{F}}_y = \hat{\Pi}_{xy}\mathcal{F}_y$
for $l = 1$ to L **do**
  Sample $\theta_l \sim \mathcal{U}(\mathbb{S}^{d-1})$
  Compute $v_l = \mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_x, \theta_l \sharp \hat{\mathcal{F}}_y) + \mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_y, \theta_l \sharp \hat{\mathcal{F}}_x)$
end for
Compute $\widehat{\mathcal{L}_{biSW}} = \left(\frac{1}{L}\sum_{l=1}^{L} v_l\right)^{\frac{1}{p}}$
Return: $\widehat{\mathcal{L}_{biSW}}$

Algorithm 3 Computational algorithm of the unidirectional EBSW distance

Input: Features extracted from feature extractor module $\mathcal{F}_x, \mathcal{F}_y$; $p \geq 1$; soft features similarity $\hat{\Pi}$ from Eq. 9; and the number of projections L.

Compute $\hat{\mathcal{F}}_y = \hat{\Pi}_{xy}\mathcal{F}_y$
for $l = 1$ to L **do**
  Sample $\theta_l \sim \mathcal{U}(\mathbb{S}^{d-1})$
  Compute $v_l = \mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_x, \theta_l \sharp \hat{\mathcal{F}}_y)$
  Compute $w_l = f(\mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_x, \theta_l \sharp \hat{\mathcal{F}}_y))$
end for
Compute $\widehat{\mathcal{L}_{uniEBSW}} = \left(\frac{1}{L}\sum_{l=1}^{L} v_l \frac{w_l}{\sum_{i=1}^{L} w_i}\right)^{\frac{1}{p}}$
Return: $\widehat{\mathcal{L}_{uniEBSW}}$

where $\tilde{w}(\theta_l) := \frac{w(\theta_l)}{\sum_{l'=1}^{L} w(\theta_{l'})}$. When $\sigma_0(\theta) = \mathcal{U}(\mathbb{S}^{d-1}) = \frac{\Gamma(d/2)}{2\pi^{d/2}}$ (a constant of θ) [38], we substitute $w(\theta_l)$ with $f(\mathrm{W}_{\theta_l,\mathcal{X}})$. We can choose the energy function $f(x) = e^x$, then the normalized importance weights become the Softmax function of $\mathrm{W}_{\theta,\mathcal{X}}$ as follows:

$$\tilde{w}(\theta_l) = \mathrm{Softmax}(\mathrm{W}_{\theta_l,\mathcal{X}}) = \frac{\exp(\mathrm{W}_{\theta_l,\mathcal{X}})}{\sum_{l'=1}^{L}\exp(\mathrm{W}_{\theta_{l'},\mathcal{X}})}$$

Based on the computation of unidirectional energy-based sliced Wasserstein distance, we can compute the bidirectional energy-based sliced Wasserstein distance, i.e. $\mathcal{L}_{biEBSW}$, in Eq. 11 as follows:

$$\widehat{\mathcal{L}_{biEBSW}} = \left(\frac{1}{L}\sum_{l=1}^{L}[(\mathrm{W}_{\theta_l,\mathcal{X}} + \mathrm{W}_{\theta_l,\mathcal{Y}})\hat{w}(\theta_l)]\right)^{\frac{1}{p}}, \quad (20)$$

where we denote $\mathrm{W}_{\theta,\mathcal{Y}} := \mathrm{W}_p^p(\theta \sharp \mathcal{F}_y, \theta \sharp \hat{\mathcal{F}}_x)$, and $\hat{w}(\theta_l) := \frac{\exp(\mathrm{W}_{\theta_l,\mathcal{X}}+\mathrm{W}_{\theta_l,\mathcal{Y}})}{\sum_{l'=1}^{L}\exp(\mathrm{W}_{\theta_{l'},\mathcal{X}}+\mathrm{W}_{\theta_{l'},\mathcal{Y}})}$. It is worth noting that the importance weights of $\widehat{\mathcal{L}_{biEBSW}}$ in Eq. 20 are different from that

Algorithm 4 Computational algorithm of the bidirectional EBSW distance

Input: Features extracted from feature extractor module $\mathcal{F}_x, \mathcal{F}_y$; $p \geq 1$; soft features similarity $\hat{\Pi}$ from Eq. 9; and the number of projections L.

Compute $\hat{\mathcal{F}}_x = \hat{\Pi}_{yx}\mathcal{F}_x$ and $\hat{\mathcal{F}}_y = \hat{\Pi}_{xy}\mathcal{F}_y$
for $l = 1$ to L **do**
  Sample $\theta_l \sim \mathcal{U}(\mathbb{S}^{d-1})$
  Compute $v_l = \mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_x, \theta_l \sharp \hat{\mathcal{F}}_y) + \mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_y, \theta_l \sharp \hat{\mathcal{F}}_x)$
  Compute $w_l = f(\mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_x, \theta_l \sharp \hat{\mathcal{F}}_y) + \mathrm{W}_p^p(\theta_l \sharp \mathcal{F}_y, \theta_l \sharp \hat{\mathcal{F}}_x))$
end for
Compute $\widehat{\mathcal{L}_{biEBSW}} = \left(\frac{1}{L}\sum_{l=1}^{L} v_l \frac{w_l}{\sum_{i=1}^{L} w_i}\right)^{\frac{1}{p}}$
Return: $\widehat{\mathcal{L}_{biEBSW}}$

Algorithm 5 Algorithm of the adaptive refinement

Input: Pair shapes $\mathcal{X}, \mathcal{Y}$ with their Laplace-Beltrami operators Φ_x, Φ_y . Trained model with parameter $\mathcal{G}_\Theta$. Number of refinement steps T.

while reach T **do**
  Compute $\mathcal{F}_x = \mathcal{G}_\Theta(\mathcal{X}, \Phi_x)$ and $\mathcal{F}_y = \mathcal{G}_\Theta(\mathcal{Y}, \Phi_y)$. ▷ Extract features.
  Compute $C_{xy}, C_{yx} = \mathrm{FMSolver}(\mathcal{F}_x, \mathcal{F}_y, \Phi_x, \Phi_y)$. ▷ Find functional map via FM solver.
  Compute $\tilde{\Pi}_{xy}, \tilde{\Pi}_{yx} = \mathrm{Sinkhorn}(\mathcal{F}_x, \mathcal{F}_y)$. ▷ Estimate pseudo similarity matrix via Sinkhorn.
  Compute unsupervised losses $\mathcal{L}_{total}(\mathcal{F}_x, \mathcal{F}_y, C_{xy}, C_{yx}, \tilde{\Pi}_{xy}, \tilde{\Pi}_{yx})$.
  Update features and soft similarity matrix by minimizing $\mathcal{L}_{total}$.
end while
Compute $P = NN(\mathcal{F}_x, \mathcal{F}_y)$ ▷ Compute point-to-point correspondence via nearest neighbor search.
Return: P

of $\widehat{\mathcal{L}_{uniEBSW}}$ in Eq. 19, since the slicing distribution here is shared and affected by both one-dimensional Wasserstein distances, thus providing a more expressive projecting features for computing sliced Wasserstein distance. We provide a pseudo-code for computing the unidirectional and bidirectional energy-based sliced Wasserstein distance in Algorithm 3 and Algorithm 4, respectively.

Adaptive refinement. As discussed in Sec. 4.5, we refine our correspondence result by estimating the pseudo-soft correspondence via entropic regularized optimal transport. The pseudo-code for our adaptive refinement is given in Algorithm 5.

11. Implementation details

All experiments are implemented using Pytorch 2.0, and executed on a system equipped with an NVIDIA GeForce RTX GPU 2080 Ti and an Intel Xeon(R) Gold 5218 CPU. We employ DiffusionNet [50] as the feature extraction mechanism, with wave kernel signatures (WKS) [6] serving as the input features. The dimension of the WKS is set to 128 for all of our experiments. Regarding spectral resolution, we opt for the first 200 eigenfunctions derived from the Laplacian matrices to form the spectral embedding. The output features of the feature extractor are set to 256. During training, the value of the learning rate is set to $1e-3$ with cosine annealing to the minimum learning rate of $1e-4$. The network is optimized with Adam optimizer with batch size 1. About adaptive refinement, the number of refinement iterations is empirically set to 12.

Regarding the loss functions, as stated in Eq. 13, we empirically set $\lambda_1 = \lambda_3 = 1.0, \lambda_2 = 100.0$. About the weight for each component of $\mathcal{L}_{fmap}$ in Eq. 8, we set $\alpha_1 = \alpha_2 = 1.0$. Regarding Sliced Wasserstein distance and energy-based sliced Wasserstein distance, we set $p = 2, L = 200$ for all of our experiments.

12. Additional visualizations

In this section, we provide additional visualizations of our proposed approach on multiple datasets.

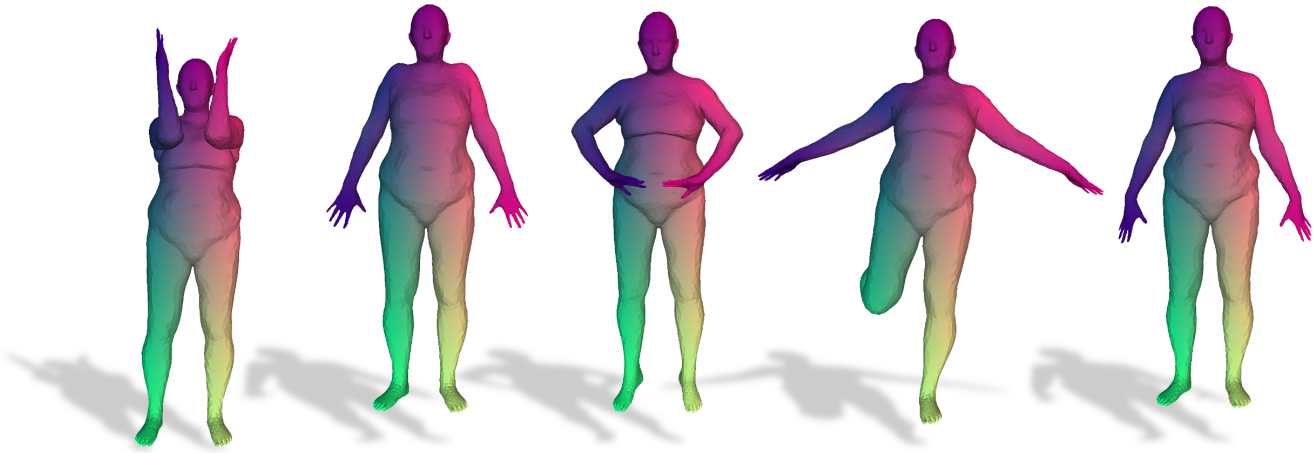

Figure 5. Qualitative results of our method on FAUST dataset.

Figure 6. Qualitative results of our method on SCAPE dataset.

Figure 7. Qualitative results of our method on SHREC dataset.

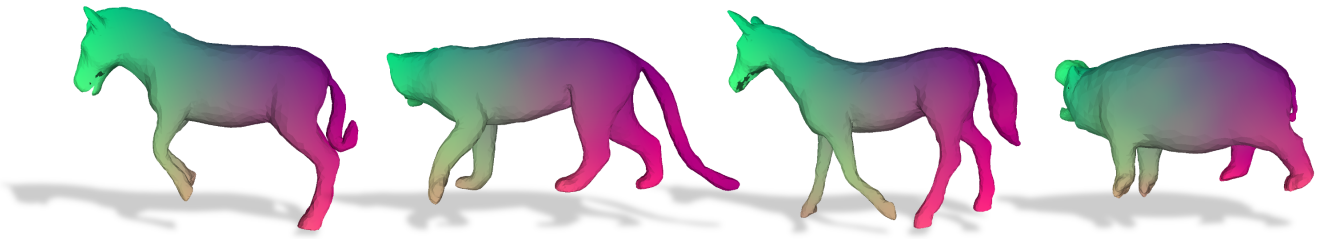

Figure 8. Qualitative results of our method on SMAL dataset.

Figure 9. Qualitative results of our method on DT4D-H dataset.

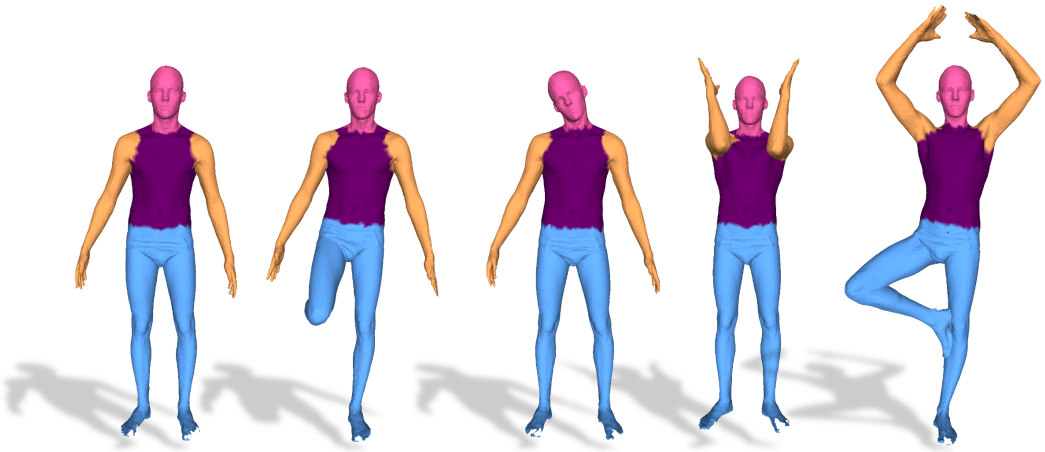

Figure 10. Qualitative results of our method on segmentation transfer coarse FAUST dataset.

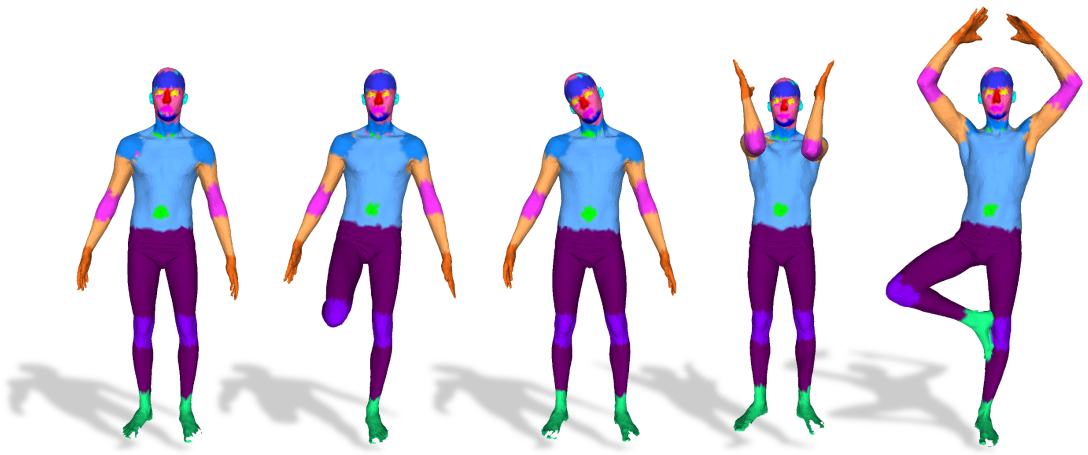

Figure 11. Qualitative results of our method on segmentation transfer fine-grained FAUST dataset.