FilterReg: Robust and Efficient Probabilistic Point-Set Registration using Gaussian Filter and Twist Parameterization Supplemental Document

Wei Gao
Massachusetts Institute of Technology
weigao@mit.edu

Russ Tedrake
Massachusetts Institute of Technology
russt@mit.edu

Abstract

This supplemental document is organized as follows:

- *Sec. 1 presents the complete derivation of the EM procedure of the proposed probabilistic model.*
- *Sec. 2 describes the technical details about the permutohedral lattice filter for the E step.*
- *Sec. 3 describes the twist parameterization for the node-graph [5] deformable kinematic model. The algorithm details and GPU-implementation are also discussed.*
- *Sec. 4 reports the parameters ans setup for several experiments.*

1. Derivation of the EM procedure

Recall that we use X, Y to denote the two point sets, $x_1, x_2, ..x_M$ and $y_1, y_2, ..., y_N$ are points in X and Y. The point-set X is defined as the **model** that is controlled by the **motion parameter** θ. Another point set Y is defined as the **observation**. According to the paper, We assume the following factorization schemes

$$\begin{aligned} p(X,\theta|Y) &\propto \phi_{\text{kinematic}}(X,\theta)p_{\text{geometric}}(X|Y) \\ &\propto \phi_{\text{kinematic}}(X,\theta)\prod_{i=1}^{M} p_{\text{geometric}}(x_i|Y) \end{aligned} \tag{1}$$

and the geometric distribution of each model point is

$$p_{\text{geometric}}(x_i|Y) = \sum_{j=1}^{N+1} p(y_j)p(x_i|y_j) \tag{2}$$

where $p(x_i|y_j) = \mathcal{N}(x_i; y_j, \Sigma_{xyz})$ is the Probability Density Function (PDF) of Gaussian distribution, $\Sigma_{xyz} = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ is the diagonal covariance matrix. An additional uniform distribution $p(x_i|y_{N+1}) = \frac{1}{M}$ is added to account for the noise and outliers. Similar to [8], we use equal membership probabilities $p(y_j) = \frac{1}{N}$ for all GMM components, and introduce a parameter $0 \leq w \leq 1$ to represent the weight of uniform outlier distribution. Thus, the geometric distribution of each model point is

$$p_{\text{geometric}}(x_i|Y) = \frac{w}{M} + (1-w)\frac{1}{N}\sum_{j=1}^{N} p(x_i|y_j) \tag{3}$$

We use maximum likelihood estimation to estimate the motion parameter θ and model geometric X. The log-likelihood function that we want to maximize is

$$L = \log(\phi_{\text{kinematic}}(X,\theta)) + \sum_{i=1}^{M} \log(\sum_{j=1}^{N+1} p(y_j)p(x_i|y_j)) \tag{4}$$

This optimization has sum terms within the log function, which is not easy to deal with. Thus, we use the EM [3] algorithm to solve it. We introduce the latent indicator variable Z, where $z_{ij} \in \{0, 1\}$ represents whether model point x_i is responsible for observation point y_j.

The EM algorithm exploits the following relationship, while holds for an arbitrary distribution of the latent variable $q(Z)$:

$$\begin{aligned} \log p(X,\theta|Y) = \mathcal{L}_q(X) + KL(q|p) + \log(\phi_{\text{kinematic}}(X,\theta)) \\ + \text{constant} \end{aligned} \tag{5}$$

where

$$\begin{aligned} \mathcal{L}_q(X) &= \sum_Z q(Z)\log(\frac{p_{\text{geometric}}(X,Z|Y)}{q(Z)}) \\ KL(q|p) &= -\sum_Z q(Z)\log(\frac{p(Z|X,Y)}{q(Z)}) \end{aligned} \tag{6}$$

here we have used the factorization in (1), and $KL(q|p)$ is the KL-divergence between $q(Z)$ and $p(Z|X,Y)$. The abstract EM procedure is

E step:

$$q(Z) \leftarrow p(Z|X^{\text{old}}, Y) \tag{7}$$

M step: solve the following maximization problem with respect to X and θ given $q(Z)$ in the abstract E step (7)

$$\mathcal{L}_q(X) + \log(\phi_{\text{kinematic}}(X, \theta)) \tag{8}$$

Following these procedures, the log-likelihood function $\log p(X, \theta|Y)$ is guaranteed not to decrease.

The posterior probability of the correspondence variable z_{ij} can be computed as

$$\begin{aligned} p(z_{ij} = 1|X, Y) &= \frac{p(z_{ij} = 1, x_i|Y)}{p(x_i|Y)} \\ &= \frac{\frac{1-w}{N}\mathcal{N}(x_i|y_j, \Sigma_{xyz})}{\frac{1-w}{N}\sum_{k=1}^{N}\mathcal{N}(x_i|y_k, \Sigma_{xyz}) + \frac{w}{M}} \\ &= \frac{\mathcal{N}(x_i|y_j, \Sigma_{xyz})}{\sum_{k=1}^{N}\mathcal{N}(x_i|y_k, \Sigma_{xyz}) + c} \end{aligned} \tag{9}$$

where $c = \frac{w}{1-w}\frac{N}{M}$ is a constant. The $\mathcal{L}_q(X)$ can be calculated as:

$$\begin{aligned} \mathcal{L}_q(X) &= \sum_Z p(Z)\log(p(X|Y, Z)) \\ &= \sum_Z p(Z)\log(\prod_i\prod_j p(x_i|y_j, z_{ij} = 1)^{z_{ij}}) \\ &= \sum_Z p(Z)\sum_i\sum_j z_{ij}\log(\mathcal{N}(x_i|y_j, \Sigma_{xyz})) \\ &= \sum_i\sum_j \alpha_{ij}\log(\mathcal{N}(x_i|y_j, \Sigma_{xyz})) \\ &= -\frac{1}{2}\sum_i\sum_j \alpha_{ij}(x_i - y_j)^T\Sigma_{xyz}^{-1}(x_i - y_j) \\ &\quad - \frac{\log(|\Sigma_{xyz}|)}{2}\sum_i\sum_j \alpha_{ij} + \text{constant} \end{aligned} \tag{10}$$

where $\alpha_{ij} = E(z_{ij}|X, Y) = p(z_{ij} = 1|X, Y)$. Note that Y is fixed during registration, we have

$$\begin{aligned} &\sum_i\sum_j \alpha_{ij}(x_i - y_j)^T\Sigma_{xyz}^{-1}(x_i - y_j) \\ &= \sum_i\sum_j \alpha_{ij}(x_i^T\Sigma_{xyz}^{-1}x_i - 2x_i^T\Sigma_{xyz}^{-1}y_j) + \text{constant} \\ &= \sum_i((\sum_j \alpha_{ij})(x_i^T\Sigma_{xyz}^{-1}x_i) - 2x_i^T\Sigma_{xyz}^{-1}(\sum_j \alpha_{ij}y_j)) \\ &+ \text{constant} \end{aligned} \tag{11}$$

here the constant means they don't depend on X. By inserting $\alpha_{ij} = p(z_{ij} = 1|X, Y)$ and the posterior (9), we get the concrete EM procedure

E step: For each x_i, compute

$$\begin{aligned} M_{x_i}^0 &= \sum_{y_k}\mathcal{N}(x_i^{\text{old}}|y_k, \Sigma_{xyz}) \\ M_{x_i}^1 &= \sum_{y_k}\mathcal{N}(x_i^{\text{old}}|y_k, \Sigma_{xyz})y_k \end{aligned} \tag{12}$$

M step: minimize the following objective function with respect to X and θ using $M_{x_i}^0$ and $M_{x_i}^1$ from the E step (12)

$$\begin{aligned} &-\log(\phi_{\text{kinematic}}(X, \theta)) \\ &+\sum_{x_i}\frac{M_{x_i}^0}{M_{x_i}^0 + c}(x_i - \frac{M_{x_i}^1}{M_{x_i}^0})^T\Sigma_{xyz}^{-1}(x_i - \frac{M_{x_i}^1}{M_{x_i}^0}) \end{aligned} \tag{13}$$

If the kinematic model encoded by $\phi_{\text{kinematic}}(X, \theta)$ is $X = X(\theta)$, the M step becomes

$$\sum_{x_i}\frac{M_{x_i}^0}{M_{x_i}^0 + c}(x_i(\theta) - \frac{M_{x_i}^1}{M_{x_i}^0})^T\Sigma_{xyz}^{-1}(x_i(\theta) - \frac{M_{x_i}^1}{M_{x_i}^0}) \tag{14}$$

1.1. Optimized Variance

In the derivation above, we assume the diagonal convariance matrix Σ_{xyz} is a fixed parameter. Similar to [8], the variance can be interpreted as a decision variable in (10). Furthermore, if the $\Sigma_{xyz} = \text{diag}(\sigma^2, \sigma^2, \sigma^2)$, the optimization with respect to Σ_{xyz} can be decoupled with the optimization with respect to X and θ. The optimization with respect X and θ to becomes,

$$\begin{aligned} \text{minimize:}\ &-\log(\phi_{\text{kinematic}}(X, \theta)) \\ &+\sum_{x_i}\frac{M_{x_i}^0}{M_{x_i}^0 + c}(x_i - \frac{M_{x_i}^1}{M_{x_i}^0})^T(x_i - \frac{M_{x_i}^1}{M_{x_i}^0}) \end{aligned} \tag{15}$$

and the optimization with respect to σ becomes,

$$\begin{aligned} \text{minimize:}\ &\frac{1}{2\sigma^2}\sum_i\sum_j \alpha_{ij}(x_i - y_j)^T(x_i - y_j) \\ &+\frac{3}{2}\log(\sigma^2)\sum_i\sum_j \alpha_{ij} \end{aligned} \tag{16}$$

where the x_i in (16) is obtained by solving (15). The optimization (16) can be solved analytically and the optimal value of σ is

$$\begin{aligned} \sigma^2 &= \frac{\sum_i\sum_j \alpha_{ij}(x_i - y_j)^T(x_i - y_j)}{3\sum_i\sum_j \alpha_{ij}} \\ &= (\Sigma_i\frac{M_{x_i}^0 x_i^T x_i - 2x_i^T M_{x_i}^1 + M_{x_i}^2}{M_{x_i}^0 + c})/(\Sigma_i\frac{M_{x_i}^0}{M_{x_i}^0 + c}) \end{aligned} \tag{17}$$

where

$$M_{x_i}^2 = \sum_{y_k}\mathcal{N}(x_i^{\text{old}}|y_k, \Sigma_{xyz})y_k^T y_k \tag{18}$$

should be inserted into the E step (12).

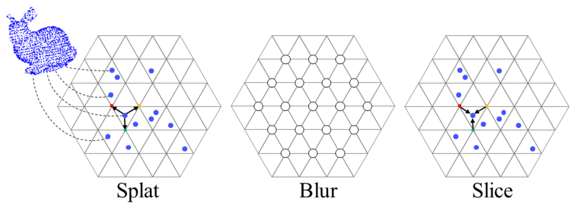

Figure 1. An illustration of the permutohedral lattice filter [1]. **Splat**: The input features are interpolated to permutohedral lattice using barycentric weights. **Blur**: lattice points exchange their values with nearby lattice points. **Slice**: The filtered signal is interpolated back onto the input signal.

2. Permutohedral Filter for the E step

We briefly summarize the filtering process of [1], an illustration is shown in Fig. 1. In permutohedral filter [1], the d-dimension feature f is first embedded in $(d+1)$-dimensional space, where the permutohedral lattice lives. In the embedded space, each input value v **Splats** onto the vertices of its enclosing simplex with barycentric weights. Next, lattice points **Blur** their values with nearby lattice points. Finally, the space is **Sliced** at each input position using the same barycentric weights to interpolate output values.

The permutohedral filter approximates the Gaussian filter with two barycentric filters (the Splat and Slice) and a Gaussian filter on the lattice (the Blur). From the analysis of [1], for filtering with d-dimension feature, the variance of the barycentric filter is $d(d+1)^2/12$, and the variance of the lattice point Gaussian filter is $d(d+1)^2/2$. The total variance is $2d(d+1)^2/3$. As the feature is in d dimensional feature space, the total variance on each dimension is $2(d+1)^2/3$. To approximate a Gaussian filter with unit variance, the feature space should be scaled by $\sqrt{\frac{2}{3}}(d+1)$ in the feature embedding stage.

As mentioned in the paper, the performance of the permutohedral filter decreases along with the variance value. To resolve it, we propose to omit the Blur stage if the relative variance is small. This approximation is less accurate than the full pipeline for large variance, but it is sufficient for point-set registration with small variance in practice. Additionally, the lattice index built without the Blur stage doesn't need rebuilding among EM iterations, as only the Blur stage involves the model point X which will be updated with motion parameter θ.

By omitting the Blur stage, we are approximating the Gaussian filter with two barycentric filters, and the total variance is $d(d+1)^2/6$. To approximate the Gaussian filter with unit variance on the normalized feature, we need to scale the feature space by $\sqrt{\frac{1}{6}}(d+1)$ in the feature embedding stage. We approximately characterize the relative value of variance Σ_f using the number of the lattice vertices

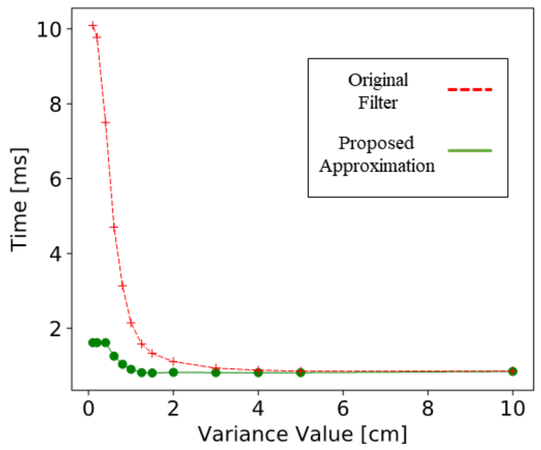

Figure 2. The time to build the index for the E step at different variance values. The proposed approximation is more efficient than the original permutohedral filter [1] if the variance is small.

and the size of the input cloud. Let N_{lattice} be the number of lattice vertices, and N_{cloud} be the number of points in the input point cloud. If the variance Σ_f is large, N_{lattice} would be much smaller than N_{cloud}. On the contrary, N_{lattice} would be comparable to N_{cloud} for small variance. Thus, we use full pipeline if $N_{\text{lattice}} < \alpha N_{\text{cloud}}$, else we omit the Blur stage. The value of α is determined empirically, we found $0.01 < \alpha < 0.1$ works well in practice and $\alpha = 0.015$ is used for all the presented numerical experiments.

We compare the performance of index building using the original permutohedral filter and our approximation. The time to build the index is shown in Fig. 2. Both the original permutohedral filter and the proposed approximation are tested on an outlier-corrupted bunny model with 10000 points. The diameter of the bunny model is about 10 [cm]. From Fig. 2, if the variance is small, the Blur stage dominates the overall performance and the original permutohedral filter is less efficient than our approximation.

3. Deformable Kinematic Model with Twist

In this section, we present the twist parameterization for the node-graph deformable kinematic model. Recall that the node graph is defined as a set $\{[p_j \in R^3, T_j \in SE(3)]\}$, where j is the node index, p_j is the position of the jth node, and T_j is the $SE(3)$ defined on the jth node. T_j is represented by DualQuaternion [6] $\hat{q}_j$ for better interpolation. The motion parameter $\theta = [\hat{q}_1, \hat{q}_2, ..., \hat{q}_{N_{\text{node}}}]$. The kinematic equation can be written as

$$T_i(\theta) = \text{normalized}(\Sigma_{k \in N_i(x_{i\_\text{reference}})} w_{ki} \hat{q}_k) \quad (19)$$

where $N_i(x_{i\_\text{reference}})$ is the nearest neighbor nodes of model point x_i, and w_{ki} is the skinning weight.

Similar to the case of the articulated registration, the computation of the A matrix for the node-graph deformable kinematic can be factored as,

$$A = \sum_{\text{node-pair}_{kl}} (\frac{\partial \zeta_k}{\partial \theta})^T (\sum_{x_i \text{ in node-pair}_{kl}} w_{ki} w_{li} \frac{\partial \text{residual}_i}{\partial \zeta_i}^T \frac{\partial \text{residual}_i}{\partial \zeta_i}) (\frac{\partial \zeta_l}{\partial \theta}) \quad (20)$$

where ζ_k and ζ_l are the twist of node k and l, and we have exploited $\frac{\partial \zeta_i}{\partial \zeta_k} = w_{ki} I_{6\times 6}$. The Jacobian $\frac{\partial \zeta_k}{\partial \theta}$ is a blocked matrix

$$\frac{\partial \zeta_k}{\partial \theta} = [0_1, 0_2, ..., 0_{k-1}, I_{6\times 6}, 0_{k+1}, ..., 0_{N_{\text{node}}}] \quad (21)$$

where 0_i is a 6×6 matrix. Here we again parameterize the change of θ with twists of nodes. The algorithm to compute the A matrix is summarized in Algorithm 1.

We implement the Algorithm 1 on GPU with Nvidia CUDA. To achieve it, an inverse index from node pairs to model points is required. We use a GPU hash table to maintain this index, and the index is built by letting each model point insert its connected node in parallel. With the index, lines 1-4 of the Algorithm 1 can be easily implemented (in parallel). As the A matrix is a block sparse matrix, we represent it using the block compressed row format (BCSR). We first build the BCSR row index and column index using the map from node pairs to model points. With the BCSR column index, lines 5-7 of the Algorithm 1 can be implemented by inserting $J^T J_{\text{twist\_}kj}$ to its corresponded blocks.

Algorithm 1 The A matrix for deformable kinematic

```
1: for all node-pair$_{kj}$ do                                   ▷ Parallelizable
2:     $J^T J_{\text{twist\_}kj} = 0_{6\times 6}$
3:     for all point $x_i$ in node-pair$_{kj}$ do                ▷ Parallelizable
4:         $J^T J_{\text{twist\_}j} \mathrel{+}= w_{ki} w_{ji} \frac{\partial \text{residual}_i}{\partial \zeta_i}^T \frac{\partial \text{residual}_i}{\partial \zeta_i}$
5: ▷ $A$ is stored as a block sparse matrix
6: $A = 0_{N_{\text{joint}} \times N_{\text{joint}}}$
7: for all node-pair$_{kj}$ do
8:     ▷ $A[k, j]$ is the $6 \times 6$ block at $(k, j)$
9:     $A[k, j] = A[j, k] = J^T J_{\text{twist\_}kj}$
```

4. Parameters for Experiments

In this section, we summarize the setup and parameters for several experiments that is not presented in the main paper.

4.1. Robustness Test on Synthetic Data

The setup and parameters for the presented method and the baselines are:

CPD [8]: Our code is a C++ re-implementation of the original author's Matlab code. We modify it to enable a user-defined initial variance σ_{init}. We set the initial variance to be $\sigma_{\text{init}} = 5$ [cm] and the outlier ratio to be $w = 0.3$.

Proposed: For the proposed method with fixed variance, we use $\sigma = 2$ [cm]. For the proposed method with updated variance, we set $\sigma_{\text{init}} = 5$ [cm]. For both variants of our method, we set outlier ratio to be $w = 0.3$.

TrICP [2]: We modify the implementation of TrICP [2] inside PCL [9], and the internal correspondence search is performed by FLANN [7]. We hard-coded the correspondence filter in PCL [9] to improve the performance. We use truncated distance to be 5 [cm] and accept top 75% correspondence pairs.

4.2. Rigid Registration on Real-World Data

For the setup and parameters of baseline methods, please refer to [4]. For the proposed method with fixed variance, we use $\sigma = 8$ [cm]. For the proposed method with updated variance, we set $\sigma_{\text{init}} = 20$ [cm]. For both variants of our method, we use outlier ratio $w = 0.1$.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010. 3

[2] D. Chetverikov, D. Stepanov, and P. Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and Vision Computing*, 23(3):299–309, 2005. 4

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. 1

[4] B. Eckart, K. Kim, and J. Kautz. Fast and accurate point cloud registration using trees of gaussian mixtures. *arXiv preprint arXiv:1807.02587*, 2018. 4

[5] L. Kavan, S. Collins, C. OSullivan, and J. Zara. Dual quaternions for rigid transformation blending. 1

[6] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):105, 2008. 3

[7] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. 4

[8] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, Dec 2010. 1, 2, 4

[9] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011. 4