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

Wei Gao
Massachusetts Institute of Technology
weigao@mit.edu

Russ Tedrake
Massachusetts Institute of Technology
russt@mit.edu

Abstract

Probabilistic point-set registration methods have been gaining more attention for their robustness to noise, outliers and occlusions. However, these methods tend to be much slower than the popular iterative closest point (ICP) algorithms, which severely limits their usability. In this paper, we contribute a novel probabilistic registration method that achieves state-of-the-art robustness as well as substantially faster computational performance than modern ICP implementations. This is achieved using a rigorous yet computationally-efficient probabilistic formulation. Point-set registration is cast as a maximum likelihood estimation and solved using the EM algorithm. We show that with a simple augmentation, the E step can be formulated as a filtering problem, allowing us to leverage advances in efficient Gaussian filtering methods. We also propose a customized permutohedral filter [1] for improved efficiency while retaining sufficient accuracy for our task. Additionally, we present a simple and efficient twist parameterization that generalizes our method to the registration of articulated and deformable objects. For articulated objects, the complexity of our method is almost independent of the Degrees Of Freedom (DOFs). The results demonstrate the proposed method consistently outperforms many competitive baselines on a variety of registration tasks. The video demo and source code are available on our project page.

1. Introduction

Point-set registration is the task of aligning two point clouds by estimating their relative transformation. This problem is an essential component for many practical vision systems, such as SLAM [27], object pose estimation [20], dense 3d reconstruction [40], and interactive tracking of articulated [24] and deformable [12] objects.

The ICP [3] algorithm is the most widely used method for this task. ICP alternatively establishes nearest-neighbor correspondences and minimizes the point-pair distances. With spatial indices such as the KD-tree, ICP provides relatively fast performance. The literature contains many variants of the ICP algorithm; [28] and [29] provide a thorough review and comparison.

Despite its popularity, the ICP algorithm is susceptible to noise, outliers and occlusions. These limitations have been widely documented in the literature [9, 25, 15]. Thus, a great deal of research has been done on the use of probabilistic models for point-set registration [25, 14, 13], which can in principle provide better outlier-rejection. Additionally, if each point is given a Gaussian variance, the point cloud can be interpreted as a Gaussian Mixture Model (GMM). Most statistical registration methods are built on the GMM and empirically provide improved robustness [25, 15, 10]. However, these methods tend to be much slower than the ICP and can hardly scale to large point clouds, which severely limits their practical usability.

In this paper, we present a novel probabilistic registration algorithm that achieves state-of-the-art robustness as well as substantially faster computational performance than modern ICP implementations. To achieve it, we propose a computationally-efficient probabilistic model and cast the registration as a maximum likelihood estimation, which can be solved using the EM algorithm. With a simple augmentation, we formulate the E step as a filtering problem and solve it using advances in efficient Gaussian filters [1, 5, 2]. We also present a customized permutohedral filter [1] with improved efficiency while retaining sufficient accuracy for our task. Empirically our method is as robust as state-of-the-art GMM-based methods, such as [25]. In terms of the speed, our method with CPU is 3-7 times faster than modern ICP implementations and orders of magnitude faster than typical robust GMM-based methods. Furthermore, the proposed method can be GPU-parallelized and is 7 times faster than the CPU implementation.

Additionally, we propose a simple and efficient twist parameterization that extends our method to articulated and node-graph [17] deformable objects. Our method is easy to implement and achieves substantial speedup over direct parameterization. For articulated objects, the complexity of our method is almost independent of the DOFs, which makes it highly efficient even for high-DOF systems. Combining these components, we present a robust, efficient and general registration method that outperforms many competitive baselines on a variety of registration tasks. The video demo, supplemental document and source code are available on our project page.

2. Related Work

The problem of point set registration is extensively pursued in computer vision and an exhaustive review is prohibitive. In the following text, we limit our discussion to GMM-based probabilistic registration and review them roughly according to their underlying probabilistic models.

The earliest statistical methods [30, 21, 23, 14] implicitly assumed the model points, which is controlled by the motion parameters (such as the rigid transformation or joint angles), induce a GMM distribution over the 3d space. The observation points are independently sampled from this distribution. Later, several contributions [25, 32, 15] derived the EM procedure rigorously from the aforementioned probabilistic model. This formulation has also been applied to the registration of multi-rigid [10], articulated [42, 15] and deformable [25, 32] objects.

想了解一下E-M算法

Another type of algorithms is known as the correlation-based methods [38, 16, 4, 33]. These algorithms treat both observation points and model points as probabilistic distributions. The point-cloud registration can be interpreted as minimizing some distance between distributions, for instance the KL-divergence. To improve the efficiency, techniques such as voxelization [33] or Support Vector Machine [4] are used to create compact GMM representations.

In this paper, we assume that the observation points induce a probabilistic distribution over the space. Intuitively, the registration is to move the model points to positions with large posterior probability, subject to kinematic constraints. This formulation is related to several existing works [9, 22, 25], and a more technical comparison is presented in Sec. 3.2. In addition to the formulation, the key contribution of our work includes the introduction of the filter-based correspondence and twist parameterization built on the probabilistic model, as mentioned in Sec. 1. Combining these components, the proposed method is general, robust and efficient that outperform various competitive baselines.

3. Probabilistic Model for Registration

3.1. Probabilistic Formulation

In this subsection, we present our probabilistic model for point-set registration. 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. We define the **model** X as the point set that is controlled by the **motion parameter** θ. Another point set Y is defined as the **observation**, which is fixed during the registration.

We are interested in the joint distribution $p(X, Y, \theta)$. We assume given model geometry X, the observation Y is independent of θ, and the joint distribution $p(X, Y, \theta)$ can be factored as

$$p(X, Y, \theta) \propto \phi_{\text{kinematic}}(X, \theta)\phi_{\text{geometric}}(X, Y) \quad (1)$$

where $\phi_{\text{geometric}}(X, Y)$ is the potential function that encodes the geometric relationship, and the potential $\phi_{\text{kinematic}}(X, \theta)$ encodes the kinematic model. The $\phi_{\text{kinematic}}(X, \theta)$ can encode hard constraints such as $X = X(\theta)$ and/or soft motion regularizers, for instance the smooth terms in [25, 26] and the non-penetration term in [32].

We further assume the kinematic model $\phi_{\text{kinematic}}(X, \theta)$ has already captured the dependency within model points X. Thus, *conditioned on* the motion parameter θ, the points in X are independent of each other. The distribution can be further factored as

$$p(X, Y, \theta) \propto \phi_{\text{kinematic}}(X, \theta) \prod_{i=1}^{M} \phi_{\text{geometric}}(x_i, Y) \quad (2)$$

A factor graph representation of our model is shown in Fig. 1. With these factorization schemes, the conditional distribution can be written as

$$p(X, \theta|Y) \propto \phi_{\text{kinematic}}(X, \theta) \prod_{i=1}^{M} \phi_{\text{geometric}}(x_i|Y) \quad (3)$$

Following several existing work [9, 25], we let the geometric distribution of each model point $\phi_{\text{geometric}}(x_i|Y)$ be a GMM,

$$\phi_{\text{geometric}}(x_i|Y) = \sum_{j=1}^{N+1} P(y_j)p(x_i|y_j) \quad (4)$$

where $p(x_i|y_j) = \mathcal{N}(x_i; y_j, \Sigma_{xyz})$ is the Probability Density Function (PDF) of the Gaussian distribution, y_j is the Gaussian centroid and $\Sigma_{xyz} = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ 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 [25], 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 ratio of outliers.

不再是各向同性的方差

We estimate the motion parameter θ and model points X by maximizing the following log-likelihood,

$$L = \sum_{i=1}^{M} \log(\sum_{j=1}^{N+1} P(y_j)p(x_i(\theta)|y_j)) \quad (5)$$

here we restrict ourselves to the kinematic model $X = X(\theta)$ and leave the general case to supplemental materials. We use the EM [7] algorithm to solve this optimization. The EM procedure is

E step: For each x_i, compute

$$M^0_{x_i} = \sum_{y_k} \mathcal{N}(x_i(\theta^{\text{old}}); y_k, \Sigma_{xyz})$$
$$M^1_{x_i} = \sum_{y_k} \mathcal{N}(x_i(\theta^{\text{old}}); y_k, \Sigma_{xyz}) y_k \quad (6)$$

M step: minimize the following objective function

$$\sum_{x_i} \frac{M^0_{x_i}}{M^0_{x_i}+c}(x_i(\theta) - \frac{M^1_{x_i}}{M^0_{x_i}})^T \Sigma^{-1}_{xyz}(x_i(\theta) - \frac{M^1_{x_i}}{M^0_{x_i}}) \quad (7)$$

where $M^0_{x_i}$ and $M^1_{x_i}$ are computed in the E step (6), $c = \frac{w}{1-w}\frac{N}{M}$ is a constant, and w is the parameter that represents the ratio of outliers.

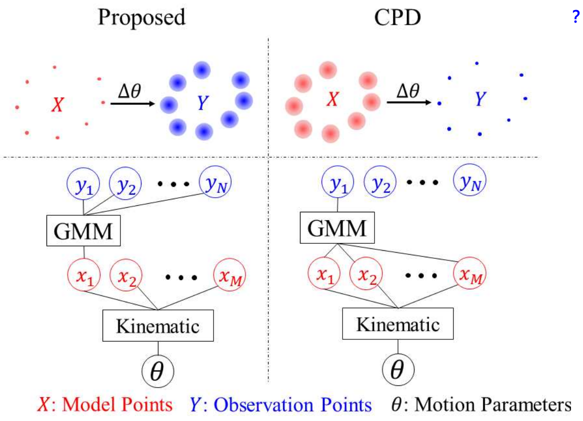
The EM procedure is conceptually related to ICP. The weight-averaged target point $(M^1_{x_i}/M^0_{x_i})$ replaces the nearest neighbour in ICP, and each model point is weighted by $\frac{M^0_{X_i}}{M^0_{X_i}+c}$. Intuitively, the averaged target provides robustness to noise in observation, while the weight for each model point should reject outliers in the model. Please refer to supplemental materials for the complete derivation.

3.2. Discussion and Comparison

At a high level, the proposed formulation can be viewed as an "inverse" of Coherent Point Drift (CPD) [25] and many similar formulations [10, 32, 15], as shown in Fig. 1. CPD [25] assumes the observation points are independently distributed according to a GMM introduced by model points, while the proposed formulation directly assumes the observation points induce a GMM over the space. Empirically, both methods are very robust to noise and outliers and significantly outperform ICP.

On the perspective of computation, the proposed method is much more simple and efficient than CPD [25] and similar formulations [10, 32, 15]. The proposed method only requires sum over Y (6), while CPD [25] requires sum over both Y and X. Moreover, if a spatial index is used to perform this sum, CPD [25] must rebuild the index every EM iteration as the model points X are updated. In our formulation, we only need to build the index once if the variance is fixed during EM iterations, which is sufficient for many applications [32, 39].

Several existing works [9, 22] also build a GMM representation of the observation points. Compared with our method, they do not explicitly account for the outlier distribution and miss the weight $\frac{M^0_{X_i}}{M^0_{X_i}+c}$. Furthermore, these methods assume each model point is only correlated with one or several "nearest" GMM centroids, while conceptually we assume each model point is correlated with all observation GMM centroids. Additionally, combined with the filter-based correspondence and twist parameterization in Sec. 4 and Sec. 5, our method tends to be much faster than these works, as demonstrated by our experiments.



？这有啥区别?

Figure 1. An illustration of the proposed probabilistic model. Top: at a high level, the proposed formulation assumes the observation Y introduces a probabilistic distribution, while CPD [25] assumes the model X introduces a distribution controlled by the motion parameter θ. Bottom: factor graph representations of both our formulation and the formulation of CPD [25].

3.3. Several Extensions

The presented probabilistic formulation can be extended to incorporate many well-established GMM-based registration techniques. Additionally, these extensions can be efficiently computed in a unified framework using the filter-based E step in Sec. 4 and the twist-based M step in Sec. 5. We select the optimized variance proposed in [25], feature correspondence in [32] and point-to-plane residual in [6] as three practically important examples, although many other methods can also be integrated in a very similar way.

Features: Currently in the E step (6), only the 3d position is used to measure the similarity between the model and observation points. Similar to [32], the E step can be extended to incorporate features such as normal, SHOT [37], learned features [31] or their concatenation. The E step for arbitrary feature is

$$M^0_{x_i} = \sum_{y_k} \mathcal{N}(f_{x_i}; f_{y_k}, \Sigma_f)$$
$$M^1_{x_i} = \sum_{y_k} \mathcal{N}(f_{x_i}; f_{y_k}, \Sigma_f) y_k \quad (8)$$

where f_{x_i} and f_{y_k} are the feature value for point x_i and y_k, Σ_f is the diagonal covariance for the feature.

Optimized Variance: In our previous formulation, the variance of Gaussian kernel Σ_{xyz} is used as a fixed parameter. Similar to CPD [25], if $\Sigma_{xyz} = \text{diag}(\sigma^2, \sigma^2, \sigma^2)$, the variance σ can be interpreted as a decision variable and optimized analytically. Please refer to supplemental materials for the detailed formula and derivation.

Point-to-Plane Distance: The objective in our M step (7) is similar to the point-to-point distance in ICP, which doesn't capture the planar structure. A simple solution is to compute a normal direction to characterize the local planar structure in the vicinity of the target $(M^1_{x_i}/M^0_{x_i})$

$$N_{x_i} = (\sum_{y_k} \mathcal{N}(x_i; y_k, \Sigma_{xyz}) N_{y_k})/M^0_{x_i} \tag{9}$$

where N_{y_k} is the normal of the observation point y_k. The objective in the M step is then a point-to-plane error

$$\sum_{x_i} \frac{M^0_{x_i}}{M^0_{x_i} + c} \text{dot}(N_{x_i}, x_i(\theta) - \frac{M^1_{x_i}}{M^0_{x_i}})^2 \tag{10}$$

4. E Step: Filter-based Correspondence

4.1. General Formulation

In this section, we discuss the method to compute the E step (6) and several extensions (8 and 9). These specific E steps can be written into the following generalized form

$$G(f_{x_i}) = \sum_{y_k} e^{-\frac{1}{2}(f_{x_i} - f_{y_k})^2} v_{y_k} \tag{11}$$

where v_{y_k} generalizes the 3d position y_k and the unit weight in (6, 8) and the normal N_{y_k} in (9). The $G(f_{x_i})$ generalizes $M^0_{x_i}$ and $M^1_{x_i}$ in (6, 8) and the normal N_{x_i} in (9). The features f_{x_i} and f_{y_k} generalize 3d positions x_i and y_k in the Gaussian PDF $\mathcal{N}(x_i; y_k, \Sigma_{xyz})$. The features f_{x_i} and f_{y_k} are normalized to have identity covariance. We also omit the normalization constant $\det(2\pi\Sigma_{xyz})^{-\frac{1}{2}}$ of the Guassian PDF $\mathcal{N}(x_i; y_k, \Sigma_{xyz})$ for notational simplicity.

Equ. (11) is known as the general Gaussian Transform and the Improved Fast Gaussian Transform (IFGT) [41] is proposed for it. However, IFGT [41] uses a k-means tree internally and there would be too many k-means centroids for typical parameters in the registration. Practically, [41] is not much faster than brute-force evaluation for our task.

We instead propose to compute (11) using Gaussian filtering algorithms [5, 1, 2], which demonstrate promising accuracy and efficiency on image processing. The filtering operation that these algorithms accelerated is

$$G(f_{y_i}) = \sum_{y_k} e^{-\frac{1}{2}(f_{y_i} - f_{y_k})^2} v_{y_k} \tag{12}$$

which is a subset of the general Gaussian transform: the feature f_{y_i} used to retrieve the filtered value $G(f_{y_i})$ must be included in the input point set Y.

In our case, we would like to retrieve the value $G(f_{x_i})$ using feature f_{x_i} from another point cloud X, which cannot be directly expressed in (12). To resolve it, we propose the following augmented input:

$$\begin{aligned} F_{\text{filter-input}} &= [F_X, \quad F_Y] \\ V_{\text{filter-input}} &= [0, \quad\;\; V_Y] \end{aligned} \tag{13}$$

where $F_X = [f_{x_1}, f_{x_2}, ..., f_{x_M}]$, $F_Y = [f_{y_1}, f_{y_2}, ..., f_{y_N}]$ and $V_Y = [v_{y_1}, v_{y_2}, ..., v_{y_N}]$. The new input feature $F_{\text{filter-input}}$ and value $V_{\text{filter-input}}$ are suitable for these filtering algorithms [5, 1, 2], and the filtered output is

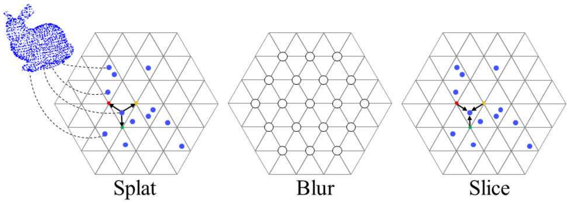


Figure 2. An illustration of the permutohedral lattice filter [1]. **Splat**: The input features are interpolated to permutohedral lattice using barycentric weights. **Blur**: lattice points exchange their values with nearby lattice points. **Slice**: The filtered signal is interpolated back onto the input signal.

$$\begin{aligned} G(f_{x_i}) &= \sum_{z_k \in F_{\text{filter-input}}} e^{-\frac{1}{2}(f_{x_i} - f_{z_k})^2} v_{z_k} \\ &= \sum_{y_k \in F_Y} e^{-\frac{1}{2}(f_{x_i} - f_{y_k})^2} v_{y_k} \end{aligned} \tag{14}$$

With this augmentation, we can apply these filtering algorithms [1, 2, 5] as a black box to our problem. However, by exploiting the structure of these methods, we can make them more efficient for our tasks. In the following text, the permutohedral lattice filter [1] is discussed as an example, which is used in our experiments.

4.2. Permutohedral Lattice Filter

We briefly review the filtering process of [1], an illustration is shown in Fig. 2. 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.

Although the permutohedral filter [1] has demonstrated promising performance on a variety of tasks, it is still not optimal for our problem. In particular, the index building in [1] can be inefficient when the variance Σ_{xyz} is too small. Additionally, naively apply [1] to the E step (6) requires rebuilding the index every EM iteration as the model point X is updated. To resolve these problems, we propose a customization of the permutohedral filter [1] that is more efficient while retaining sufficient accuracy for our task. The detailed method is presented in the supplemental material.

5. M Step: Efficient Twist Parameterization

In this section, we present methods to solve the optimizations (7, 10) with the twist parameterization. We first discuss the twist in the general kinematic model, then specialize it to articulated and node-graph deformable objects.

We focus on the following general kinematic model,

$$x_i = T_i(\theta)x_{i\_\text{reference}} \tag{15}$$

where $T_i(\theta) \in SE(3)$ is a rigid transformation, $x_{i\_\text{reference}}$ is the fixed reference point for the x_i. Note that $T_i(\theta)$ depends on i and the kinematic model is not necessarily a global rigid transformation.

Twist is a 6-vector that represents the locally linearized "change" of $SE(3)$. Let the twist $\zeta_i = (w_i, t_i) = (\alpha_i, \beta_i, \gamma_i, a_i, b_i, c_i)$ be the local linearization of T_i, we have

$$T_i^{\text{new}} \approx \begin{bmatrix} 1 & -\gamma_i & \beta_i & a_i \\ \gamma_i & 1 & -\alpha_i & b_i \\ -\beta_i & \alpha_i & 1 & c_i \\ 0 & 0 & 0 & 1 \end{bmatrix} T_i \tag{16}$$

Thus, the Jacobian $\frac{\partial x_i}{\partial \zeta_i} = [\text{skew}(x_i), I_{3\times3}]$ is a 3×6 matrix, where $I_{3\times3}$ is identity matrix, and $\text{skew}(x_i)$ is a 3×3 matrix such that $\text{skew}(x_i)b = \text{cross}(x_i, b)$ for arbitrary $b \in R^3$.

The optimization (7, 10) are least squares problems, and we focus on the following generalized form of them

$$\sum_{x_i} r_{x_i}^T r_{x_i} \tag{17}$$

where r_{x_i} is the concatenated least-squares residuals that depends on x_i. We use the Gauss-Newton (GN) algorithm to solve (17). In each GN iteration we need to compute the following A and b matrices by

$$A = \sum_{x_i} (\frac{\partial r_{x_i}}{\partial \theta})^T \frac{\partial r_{x_i}}{\partial \theta}, \quad b = \sum_{x_i} (\frac{\partial r_{x_i}}{\partial \theta})^T (r_{x_i}) \tag{18}$$

and the update of the motion parameters is $\Delta\theta = -A^{-1}b$. Thus, the primary computational bottleneck is to assemble the matrices A and b. In the following text, we only discuss the computation of the A matrix, while the computation of the b vector is similar and easier. The computation of the A matrix can be written as

$$A = \sum_{x_i} (\frac{\partial \zeta_i}{\partial \theta})^T ((\frac{\partial r_{x_i}}{\partial \zeta_i})^T \frac{\partial r_{x_i}}{\partial \zeta_i})(\frac{\partial \zeta_i}{\partial \theta}) \tag{19}$$

where $\frac{\partial \zeta_i}{\partial \theta}$ is the Jacobian that maps the change of motion parameter θ to the change of the rigid transformation T_i, while the change of T_i is expressed as its twist. Note that the term $\frac{\partial r_{x_i}}{\partial \zeta_i} = \frac{\partial r_{x_i}}{\partial x_i} \frac{\partial x_i}{\partial \zeta_i}$ is very easy to compute, as both $\frac{\partial r_{x_i}}{\partial x_i}$ and $\frac{\partial x_i}{\partial \zeta_i}$ are only dependent on x_i.

If the kinematic model is a global rigid transformation, we have $\frac{\partial \zeta_i}{\partial \theta} = I_{6\times6}$ and $A = \sum_{x_i}((\frac{\partial r_{x_i}}{\partial \zeta_i})^T \frac{\partial r_{x_i}}{\partial \zeta_i})$. In the following subsections, we proceed to the articulated and node-graph deformable kinematic models.

5.1. Articulated Model

Articulated objects consist of rigid bodies connected through joints in a kinematic tree. A broad set of real-world objects, including human bodies, hands and robots are articulated objects. If the kinematic model (15) is an articulated model, the motion parameter $\theta \in R^{N_\text{joint}}$ would be the joint angles, where N_joint is the number of joints. The $T_i(\theta)$ is the rigid transform of the rigid body that the point x_i is attached to. The computation of the A matrix can be factored as

$$A = \sum_{\text{body}_j} (\frac{\partial \zeta_j}{\partial \theta})^T (\sum_{x_i \text{ in body}_j} (\frac{\partial r_{x_i}}{\partial \zeta_i})^T \frac{\partial r_{x_i}}{\partial \zeta_i})(\frac{\partial \zeta_j}{\partial \theta}) \tag{20}$$

where ζ_j is the twist of rigid body j, and we have exploited $\frac{\partial \zeta_i}{\partial \zeta_j} = I_{6\times6}$ if point i is on rigid body j. Importantly, $\frac{\partial \zeta_j}{\partial \theta}$ is known as the spatial velocity Jacobian and is provided by many off-the-shelf rigid body simulators [34, 36, 19]. The algorithm that uses (20) is shown in Algorithm 1.

Algorithm 1 The A matrix for articulated kinematic

```
1:  for all body_j do                                  ▷ can be parallelized
2:      $J^T J_{\text{twist\_}j} = 0_{6\times6}$
3:      for all point $x_i$ in body_j do               ▷ can be parallelized
4:          $J^T J_{\text{twist\_}j}$ += $(\frac{\partial r_{x_i}}{\partial \zeta_i})^T \frac{\partial r_{x_i}}{\partial \zeta_i}$
5:  $A = 0_{N_\text{joint} \times N_\text{joint}}$
6:  for all body_j do
7:      ▷ The spatial velocity Jacobian can be computed
8:      ▷ using off-the-shelf simulators such as [36, 34]
9:      $J_{\text{spatial\_}j}$ = spatial velocity Jacobian of body_j
10:     $A$ += $J_{\text{spatial\_}j}^T (J^T J_{\text{twist\_}j}) J_{\text{spatial\_}j}$
```

The lines 1-4 of Algorithm 1 dominates the overall performance and the complexity is O($6^2 M$), where M is the number of model points and usually $M \gg N_\text{joint}$. Thus, the complexity of this algorithm is almost independent of N_joint. As a comparison, previous articulated registration methods [19, 35] need O($N_\text{joint}^2 M$) time to assemble the A matrix, and N_joint is usually much larger than 6. Furthermore, lines 1-4 of Algorithm 1 is very simple to implement and can be easily GPU parallelized. Combined with an off-the-shelf simulator, the overall pipeline can achieve promising efficiency. On the contrary, previous methods [19, 35] typically need a customized kinematic tree implementation for real-time performance, while requires substantial software engineering effort to realize.

5.2. Node-Graph Deformable Model

To capture the motion of objects such as rope or cloth, we need a kinematic model which allows large deformation while preventing unrealistic collapsing or distortion. In this paper, we follow [17] to represent the general deformable kinematic model as a node graph. Intuitively, the node graph defines a motion field in the 3D space and the reference vertex in Equ. (15) is deformed according to the motion field. More specifically, 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 $SE(3)$ motion defined on the jth node. The kinematic equation (15) can be written as

$$T_i(\theta) = \text{normalized}(\Sigma_{k \in N_i(x_{i\_\text{reference}})} w_{ki} T_k) \tag{21}$$

where $N_i(x_{i\_\text{reference}})$ is the nearest neighbor nodes of model point $x_{i\_\text{reference}}$, and w_{ki} is the fixed skinning weight. The interpolation of the rigid transformation T_k is performed using the DualQuaternion [18] representation of the $SE(3)$.

The A matrices for this kinematic model can be constructed using an algorithm very similar to Algorithm 1. The detailed method is provided in supplemental materials.

6. Results

We conduct a variety of experiments to test the robustness, accuracy and efficiency of the proposed method. Our hardware platform is an Intel i7-3960X CPU except for Sec. 6.5, where the proposed method is implemented with CUDA on a Nivida Titan Xp GPU. The video demo and the source code are available on our project page.

6.1. Robustness Test on Synthetic Data

We follow CPD [25] to setup an experiment on synthetic data. We use a subsampled Stanford bunny with 3500 points. The initial rotation discrepancy is 50 degrees with a random axis. The proposed method is compared with two baselines: CPD [25], a representative GMM-based algorithm; TrICP [6], a widely used robust ICP variant. Parameters for all methods are well tuned and provided in supplemental materials. We use the following metric to measure the pose estimation error

$$\text{error}(T) = \frac{1}{M} \Sigma_{i=1}^{M} |(T - T_{\text{gt}}) x_{i\_\text{reference}}|_2 \tag{22}$$

where T_{gt} is the known ground truth pose, $x_{i\_\text{reference}}$ defined in (15) is the reference position. We terminate the algorithm when the twist (change of transformation) is less than a threshold. In this way, the final alignment error (22) is about 1 [mm] for all methods. All of the statistical results are the averaged value of 30 independent runs.

Fig. 3 shows the robustness of different algorithms with respect to outliers in the point sets. We add different number of points randomly to both the model and observation clouds. An example of such point sets with initial alignment is shown in Fig. 3 (a), the converged alignment by the proposed method and TrICP [6] are shown in Fig. 3 (b) and Fig. 3 (c), respectively. The proposed method and CPD [25] significantly outperform the robust ICP.

Fig. 4 shows the robustness of different algorithms with respect to noise in the point sets. We corrupt each point in both model and observation clouds with a Gaussian noise. The unit of the noise is the diameter of the Bunny. An example of such point sets with initial alignment are shown in Fig. 4 (a). Fig. 4 (b) and (c) are the final alignment by the proposed method and TrICP [6] initialized from (a). Note that we use clean point clouds for better visualization. Our method and CPD [25] are more accurate than the robust ICP.

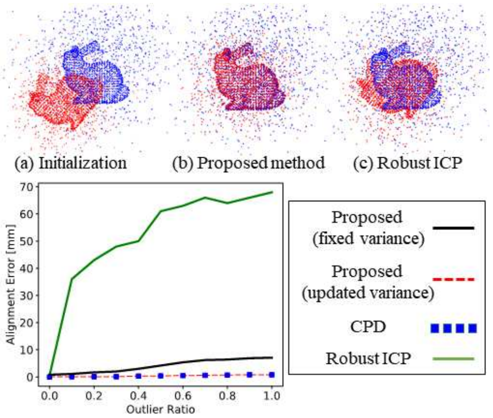


Figure 3. A comparison of the robustness of various algorithms with respect to outliers. Top: (a) shows an example initialization with 0.2 outlier ratio; (b) and (c) are the final alignment by the proposed method and TrICP [6] initialized from (a), respectively. Bottom: the alignment error (22) of each algorithm for different numbers of outliers.

	time[ms] per iteration	#iterations	overall time[ms]
Proposed fixed σ	**0.96**	40.4	**38.4**
Proposed updated σ	**1.16**	**27.6**	**32.1**
CPD	228	**26.8**	6110
Robust ICP	3.10	70.2	217.6

Table 1. The performance of different algorithms for the registration of the Stanford Bunny.

Table. 6.1 summarizes the computational performance of each method. The running time is measured on clean point cloud. Our method is about 7 times faster than TrICP [6] and two orders of magnitude faster than CPD [25]. The proposed method with fixed σ is faster per iteration, but need more iterations to converge. Overall the proposed method is as robust as the state-of-the-art statistical registration algorithm CPD [25], and runs substantially faster than the modern ICP implementation.

6.2. Rigid Registration on Real-World Data

We follow [9] to setup this experiment: the algorithm is used to compute the frame-to-frame rigid transformation on the Stanford Lounge dataset [43]. We register every 5th frame for the first 400 frames, each downsampled to about 5000 points. The average Euler angle deviation from the ground truth is used as the estimation error.

Fig. 5 (a) shows an example registration by the proposed method. Fig. 5 (b) shows the accuracy and speed of various algorithms. The results of baseline methods are from

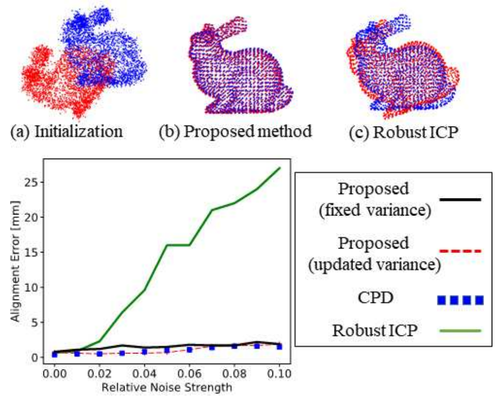
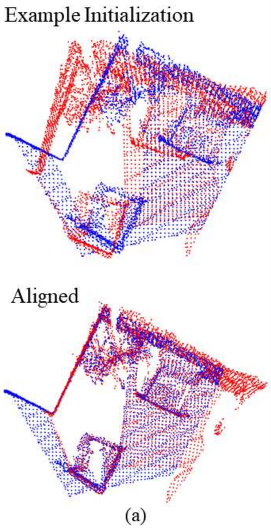


Figure 4. A comparison of the robustness of various algorithms with respect to noise. Top: (a) shows an example initialization with 0.03 relative noise; (b) and (c) are the final alignment by the proposed method and TrICP [6] initialized from (a). Note that we use clean point clouds for better visualization. Bottom: the alignment error (22) of each algorithm for different levels of noise.



Method	Ang. Error(°)	Speed(fps)
CPD	1.91	0.23
GMMReg	3.02	0.04
NDT-D2D	14.25	11.76
FICP	1.44	4.9
ICP	7.29	2.58
IRLS-ICP	2.29	7.1
EMICP (GPU)	10.47	1.44
SVR	2.67	0.35
ECMPR	2.21	0.059
JRMPC	8.27	0.042
TrICP-pt2pt	1.26	5.5
TrICP-pt2pl	**0.54**	8.4
GMMTree (GPU)	**0.46**	**19.8**
Fixed σ-pt2pt (ours)	2.30	**20.7**
Fixed σ-pt2pl (ours)	**0.78**	**38.5**
Updated σ-pt2pt (ours)	1.96	**21.2**
Updated σ-pt2pl (ours)	**0.53**	**26.7**

(b)

Figure 5. Rigid registration on the Stanford Lounge dataset [43]. The results for most baselines are from [9]. (a) shows an example registration by the proposed method. (b) shows the accuracy and performance of various algorithms.

[9][1] except for CPD [25]. For CPD [25] we use $\sigma_{\text{init}} = 20$ [cm] instead of the data-based initialization of [25], with which we observed improved performance. As the point-to-point error doesn't capture the planar structure, the point-to-point version of the proposed method as well as many other point-to-point algorithms [25, 22, 3, 6] are less accurate on this dataset. The proposed method with point-to-plane error achieves state-of-the-art accuracy. On the perspective of computation, the proposed method significantly outperforms all the baselines, including GMMTree [9] and EMICP [14] that rely on a high-end Titan X GPU.

[1] Our CPU (i7-3960X) is slightly inferior to [9] (i7-5920K), and we observe similar accuracy and slightly worse speed using CPD [25] and TrICP [6]. Thus, we think our speed result are comparable to [9] despite hardware difference.

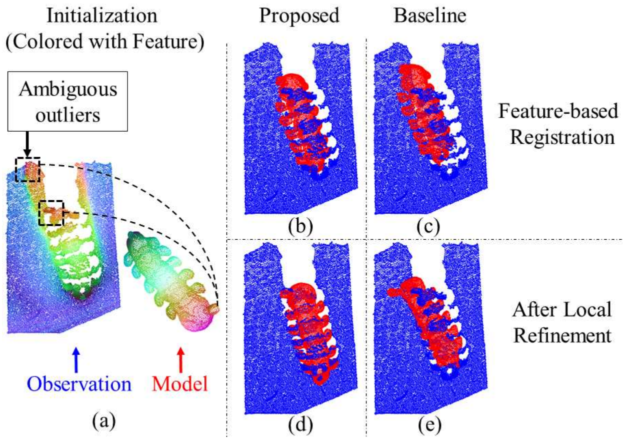


Figure 6. A feature-based global registration under ambiguous outliers and strong occlusion. (a) shows the initialization of the registration colored by the feature [11]. (b) and (c) are the feature-based registration by our method and the baseline. (d) and (e) show the final alignment using 3d local refinement initialized from (b) and (c). The proposed method converges to the correct pose while the baseline method is trapped to bad alignment.

6.3. Global Pose Estimation using Learned Features

We demonstrate global pose estimation using motion-invariant features. The task is to align a pre-built geometric model to observation clouds from RGBD images, where both the model and observation clouds are colored by the learned feature [11]. We use the proposed method with feature correspondence in Sec. 3.3 and fixed $\sigma = 0.05$ as the feature has unit norm. The proposed method is compared with a modified TrICP [6]: the nearest neighbour is searched in feature space (instead of 3d-space). After feature-based registration, we apply 3d-space local refinement to get the final alignment.

Fig. 6 shows an example registration. Note that we treat the background as outliers. As shown in Fig. 6 (a), the observation (RGBD cloud) is under severe occlusion and contains very ambiguous outliers. Fig. 6 (b) and (c) show feature-based registration by our method and the "feature" TrICP. The proposed method is more robust to the outliers and occlusion. Fig. 6 (d) and (e) show the final alignment using local refinement initialized from (b) and (c). The proposed method converges to correct pose while the baseline is trapped to bad alignment. Table. 2 summaries the success rate of both methods on 30 RGBD images with different view points and lighting conditions. Our method has a higher success rate and is more efficient than the baseline.

	success rate	time [ms]
Proposed	29/30	13
Feature ICP	25/30	34

Table 2. The success rate and speed on the feature-based global registration.

6.4. Articulated Tracking

The proposed method with articulated kinematic model is used to track a robot manipulating a box. The robot and box model has 20 DOFs (12 for the floating bases of the box and the robot, 8 for robot joints). We use drake [34] for the kinematic computation in (20). We use fixed $\sigma = 1$ [cm] and set the maximum EM iterations to be 15. Our template has 7500 points and the depth cloud has about 10000 points.

Fig. 7 (a) shows the snapshots of the tracked manipulation scenario. Fig. 7 (b) shows the live geometric model and the observation clouds. Points from observation are in black, while the geometric model is colored according to rigid bodies. Note that points from the table are treated as outliers. Fig. 7 (c) summaries the averaged per-frame performance of various algorithms. The proposed twist parameterization is an order of magnitude faster than direct parameterization. Combining the filter-based correspondence and twist parameterization leads to a real-time tracking algorithm and substantial performance improvement over articulated ICP and [42].

6.5. Application to Dynamic Reconstruction

The proposed method with node-graph deformable kinematic is implemented on GPU and used as the internal nonrigid tracker of DynamicFusion [26] (our implementation). The proposed method is compared with the projective ICP, the original non-rigid tracker of [26]. We use fixed $\sigma = 2$ [cm]. Fig. 8 shows both methods operate on a RGBD sequence with relative fast motions. The proposed method tracks it correctly, while the projective ICP fails to track the right hand of the actor. The proposed method is more robust to fast and tangential motion than the projective ICP.

To test the efficiency of the proposed twist parameterization on node-graph deformable objects, we compare it with Opt [8], a highly optimized GPU least squares solver using direct parameterization. The per-frame computational performance of various algorithms is summarized in Table. 3. The GPU parallelization of our filter-based E step achieves 8 times speedup over the CPU version, and the proposed twist parameterization is about 20 times faster than [8].

	Proposed (GPU)	Proposed (CPU)	Proposed (Opt [8])
E step [ms]	7.8	62	7.8
M step [ms]	21.6	Not implemented	382

Table 3. The per-frame performance of various algorithms for deformable tracking on the sequence in Fig. 8.

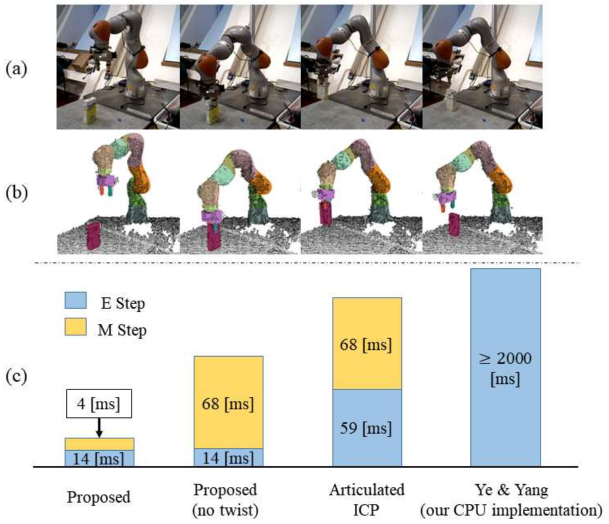


Figure 7. The proposed method is applied to track a robot manipulating a box. (a): the snapshots of the tracked manipulation scenario. (b) the observation point clouds (black) and the live geometric model (colored according to rigid bodies). (c): the per-frame performance of various algorithms on this dataset. Ye & Yang stands for our CPU implementation of [42].

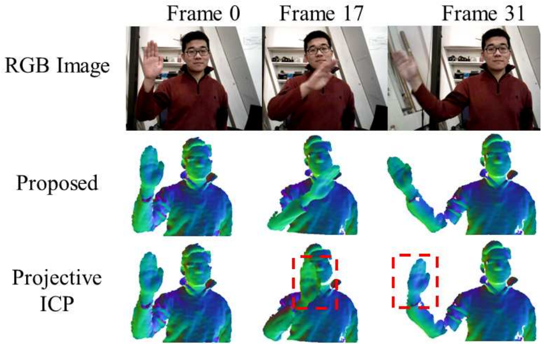


Figure 8. The proposed method with node-graph deformable kinematic is implemented on GPU and used as the internal non-rigid tracker of DynamicFusion [26]. For a relative fast motion, the proposed method tracks it correctly while the projective ICP used by DynamicFusion [26] fails to track the right hand of the actor.

7. Conclusion

To conclude, we present a probabilistic registration method that achieves state-of-the-art robustness, accuracy and efficiency. We show that the correspondence search can be formulated as a filtering problem, and employ advances in efficient Gaussian filtering methods to solve it. In addition, we present a simple and efficient twist parameterization that generalizes our method to articulated and deformable objects. Extensive empirical evaluation demonstrates the effectiveness our method.

Acknowledgments This work was supported by NSF Award IIS-1427050 and Amazon Research Award. The views expressed in this paper are those of the authors themselves and are not endorsed by the funding agencies.

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. 1, 4

[2] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian kd-trees for fast high-dimensional filtering. In *ACM Transactions on Graphics (ToG)*, volume 28, page 21. ACM, 2009. 1, 4

[3] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 1, 7

[4] D. Campbell and L. Petersson. An adaptive data representation for robust point-set registration and merging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4292–4300, 2015. 2

[5] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. In *ACM Transactions on Graphics (TOG)*, volume 26, page 103. ACM, 2007. 1, 4

[6] 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. 3, 6, 7

[7] 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. 2

[8] Z. DeVito, M. Mara, M. Zollhöfer, G. Bernstein, J. Ragan-Kelley, C. Theobalt, P. Hanrahan, M. Fisher, and M. Nießner. Opt: A domain specific language for non-linear least squares optimization in graphics and imaging. *ACM Transactions on Graphics (TOG)*, 36(5):171, 2017. 8

[9] B. Eckart, K. Kim, and J. Kautz. Fast and accurate point cloud registration using trees of gaussian mixtures. *arXiv preprint arXiv:1807.02587*, 2018. 1, 2, 3, 6, 7

[10] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis. A generative model for the joint registration of multiple point sets. In *European Conference on Computer Vision*, pages 109–122. Springer, 2014. 1, 2, 3

[11] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018. 7

[12] W. Gao and R. Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. 1

[13] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. *Pattern recognition*, 31(8):1019–1031, 1998. 1

[14] S. Granger and X. Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. In *European Conference on Computer Vision*, pages 418–432. Springer, 2002. 1, 2, 7

[15] R. Horaud, F. Forbes, M. Yguel, G. Dewaele, and J. Zhang. Rigid and articulated point registration with expectation conditional maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):587–602, 2011. 1, 2, 3

[16] B. Jian and B. C. Vemuri. A robust algorithm for point set registration using mixture of gaussians. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1246–1251. IEEE, 2005. 2

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

[18] 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. 6

[19] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu. Dart: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500, 2018. 5

[20] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic systems*, 18(3):249–275, 1997. 1

[21] B. Luo and E. R. Hancock. A unified framework for alignment and correspondence. *Computer Vision and Image Understanding*, 92(1):26–55, 2003. 2

[22] M. Magnusson. *The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection*. PhD thesis, Örebro universitet, 2009. 2, 3, 7

[23] G. McNeill and S. Vijayakumar. A probabilistic approach to robust shape matching. In *Image Processing, 2006 IEEE International Conference on*, pages 937–940. IEEE, 2006. 2

[24] L. Mundermann, S. Corazza, and T. P. Andriacchi. Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007. 1

[25] 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, 3, 6, 7

[26] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015. 2, 8

[27] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d slam3d mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007. 1

[28] F. Pomerleau, F. Colas, R. Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1):1–104, 2015. 1

[29] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3):133–148, Feb. 2013. 1

[30] A. Rangarajan, H. Chui, E. Mjolsness, S. Pappu, L. Davachi, P. Goldman-Rakic, and J. Duncan. A robust point-matching algorithm for autoradiograph alignment. *Medical image analysis*, 1(4):379–398, 1997. 2

[31] T. Schmidt, R. Newcombe, and D. Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017. 3

[32] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1130–1137. IEEE, 2013. 2, 3

[33] T. Stoyanov, M. Magnusson, and A. J. Lilienthal. Point set registration through minimization of the l2 distance between 3d-ndt models. 2

[34] R. Tedrake and the Drake Development Team. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2016. 5, 8

[35] A. Tkach, M. Pauly, and A. Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Transactions on Graphics (TOG)*, 35(6):222, 2016. 5

[36] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012. 5

[37] 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. 3

[38] Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. In *European conference on computer vision*, pages 558–569. Springer, 2004. 2

[39] B. Wang, L. Wu, K. Yin, U. Ascher, L. Liu, and H. Huang. Deformation capture and modeling of soft objects. *ACM Transactions on Graphics (TOG)*, 34(4):94, 2015. 3

[40] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016. 1

[41] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. In *null*, page 464. IEEE, 2003. 4

[42] M. Ye and R. Yang. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2345–2352, 2014. 2, 8

[43] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (ToG)*, 32(4):112, 2013. 6, 7