A New Framework for Registration of Semantic Point Clouds from Stereo and RGB-D Cameras

Ray Zhang, Tzu-Yuan Lin, Chien Erh Lin, Steven A. Parkison, William Clark, Jessy W. Grizzle, Ryan M. Eustice and Maani Ghaffari

Abstract—This paper reports on a novel nonparametric rigid point cloud registration framework that jointly integrates geometric and semantic measurements such as color or semantic labels into the alignment process and does not require explicit data association. The point clouds are represented as nonparametric functions in a reproducible kernel Hilbert space. The alignment problem is formulated as maximizing the inner product between two functions, essentially a sum of weighted kernels, each of which exploits the local geometric and semantic features. As a result of the continuous models, analytical gradients can be computed, and a local solution can be obtained by optimization over the rigid body transformation group. Besides, we present a new point cloud alignment metric that is intrinsic to the proposed framework and takes into account geometric and semantic information. The evaluations using publicly available stereo and RGB-D datasets show that the proposed method outperforms state-of-the-art outdoor and indoor frame-to-frame registration methods. An open-source GPU implementation is also provided.

I. INTRODUCTION

Point cloud registration estimates the relative transformation between two noisy point clouds [1]–[5]. Point clouds obtained by RGB-D cameras, stereo cameras, and LIDARs contain rich color and intensity measurements besides the geometric information. The extra non-geometric information can improve the registration performance [6]–[8]. Deep learning can provide semantic attributes of the scene as measurements [9]–[11]. As illustrated in Fig. 1, this work focuses on the construction of a novel integrated framework to jointly process raw geometric and non-geometric information for point cloud registration.

Real-world applications such as SLAM [12] and 3D reconstruction [13] include noisy measurements, symmetries or dynamics objects, occlusion, and blurry observations. Examples are shown in Fig. 2. These cases make the data association process challenging. Existing Iterative Closest Point (ICP)-based work [1]–[3] approach this problem by adding appearance/semantic features [6], [7], [14], adding local or deep geometric features [3], [15], and introducing weighted many-to-many correspondences [16], [17].

Gaussian Mixture Model (GMM) based registrations [18]–[20] model data correspondences and point clouds as probabilistic densities. Instead of point pairs, GMM-based methods work on point clusters, and the associations are a part of the model parameters. The relative rigid body transformation is then estimated by fitting the second point cloud measurements into the first point cloud’s distributions [19], [21]–[25], or by minimizing a distance measure between the two distribution and inferring the weight parameters [26]–[28].

This paper presents a nonparametric registration framework that jointly integrates geometric and semantic measurements and does not require explicit data association. Unlike existing methods that rely on geometric residuals with regularizers to include appearance information [14], [29], the proposed framework formulates the problem using a single objective function, and is solved by the gradient ascent on Riemannian manifolds, similar to the work of [30], [31]. In particular, this work has the following contributions.

1) A novel framework for semantic point cloud registration that generalizes geometric, color, and semantic-assisted methods to a nonparametric continuous model via a hierarchical distributed representation of features.

2) A new point cloud alignment indicator that is intrinsic to the proposed framework and takes into account geometric and semantic information.

3) An open source GPU implementation available at [32];
A. Mixture of Gaussian-based Registration Frameworks

Probabilistic registration frameworks represent point clouds as discrete [19], [41] or continuous probability densities [20]–[22], [25], [42], [43]. Compared to this work, GMM-based methods also use a double sum of Gaussian kernels, combined with the soft data association, but they come from a different theoretical background than the Reproducing Kernel Hilbert Space (RKHS) [44].

Normal Distribution Transform (NDT) defines a discrete collection of bivariate Gaussian distributions to capture local surface structures [19], [45]. Discretization brings automatic soft data association without the need of inferring GMM weights. An effective discretization strategy requires suitable voxel sizes and efficient voxel deployment, for instance a forest of octrees [19], distance based voxel sizing [46], hierarchical voxel tree deployment [47], and cell clustering [48]. Comparing to NDT, the proposed method is also data association free, but it is further a continuous representation, thus avoids the above concerns caused by discretization.

Some continuous GMM-based methods minimize the distance between two distributions. Effective distance measures include Jensen-Shannon divergence [27] and the $l_2$ distance of the two dense [28] or sparse [42] GMMs. Kernel Correlation (KC) [26] maximizes the correlation between two point clouds using M-estimators, in particular a sum of Gaussian kernels. It has an identical loss function comparing to the proposed geometric only inner product, but its kernel length-scales stay fixed throughout the optimization. In addition, KC discretizes the space to avoid the quadratic time cost, while our methods remain continuous with the help of GPU parallel computations.

B. Deep Learning in Registration

Fully connected layers with symmetric operations (max-pooling) in PointNet [49], [50], convolutions of sparse tensors in FCGF [15], sparse bilateral convolution in SPLATNet [51], and graph convolution layers in DGCNN [52] can capture local and global geometric features of point clouds. Examples of utilizing deep geometric features include PCD-Net [53], 3D-Feat-Net [54], PointNetLK [50].

Given extracted features, point correspondences are calculated in the many-to-many [55], [56] or one-to-one way [57], [58]. Correspondences of a point can be interpreted as a probabilistic distribution of its nearby points, predicted by convolutions and softmax operations over those points’ feature embeddings [59]. DCP [56] directly multiplies two feature embedding vectors between all point pairs of the two point clouds, followed by softmax operations to get the correspondences. Deep Global Registration [60] adopts convolution layers that take a candidate pair of points $(x,z) = (x_1,x_2,x_3,z_1,z_2,z_3)$ as input, and classifies whether this pair of point lies in a lower-dimensional manifold.

This work is not an end-to-end deep learning solution, but our unified point cloud function representation can incorporate deep learning features, such as semantics, into the cost function. The potential way of using our inner product as a loss of an end-to-end framework can be a future study.

III. Problem Setup

Consider two (finite) collections of points, $X = \{x_1\}$, $Z = \{z_2\} \subset \mathbb{R}^3$. We want to determine which element
h ∈ SE(3), aligns the two point clouds X and Z = \{hzj\} the “best.” To assist with this, we will assume that each point contains information described by a point in an inner product space, \((I, ⟨⟨,⟩⟩)\). To this end, we will introduce two labeling functions, \(\ell_X : X \rightarrow I\) and \(\ell_Z : Z \rightarrow I\).

To measure their alignment, we turn the clouds, X and Z, into functions \(f_X, f_Z : \mathbb{R}^3 \rightarrow I\)\) that live in some reproducing kernel Hilbert space, \((\mathcal{H}, ⟨⟨,⟩⟩)\). The action, \(\mathcal{H} \circ \mathbb{R}^3\) induces an action \(\mathcal{H} \circ \mathbb{R}^3\) by \(h.f(x) := f(h^{-1}x)\). Inspired by this observation, we will set \(h.f_Z := f_{h^{-1}Z}\).

**Problem 1.** The problem of aligning the point clouds can now be phrased as maximizing the scalar products of \(f_X\) and \(h.f_Z\), i.e., we want to solve

\[
\arg\max_{h \in \text{SE}(3)} F(h), \quad F(h) := ⟨f_X, f_{h^{-1}Z}⟩_{\mathcal{H}}. \quad (1)
\]

**A. Constructing The Functions**

We follow the same steps in [30] with an additional step in which we use the kernel trick to kernelize the information inner product. For the kernel of our RKHS, \(\mathcal{H}\), we first choose the squared exponential kernel \(k : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}:\)

\[
k(x, z) = \sigma^2 \exp\left(-\frac{||x - z||^2}{2\ell^2}\right), \quad (2)
\]

for some fixed real parameters (hyperparameters) \(\sigma\) and \(\ell\) (the lengthscale), and \(||\cdot||_3\) is the standard Euclidean norm on \(\mathbb{R}^3\). This allows us to turn the point clouds to functions via

\[
f_X(\cdot) := \sum_{x_i \in X} \ell_X(x_i)k(\cdot, x_i),
\]

\[
f_{h^{-1}Z}(\cdot) := \sum_{z_j \in Z} \ell_Z(z_j)k(\cdot, h^{-1}z_j). \quad (3)
\]

Here \(\ell_X(x_i)\) encodes the appearance information, for example LIDAR intensity and image pixel color. \(k(\cdot, x_i)\) encodes the geometric information. We can now obtain the inner product of \(f_X\) and \(f_Z\) as

\[
⟨f_X, f_{h^{-1}Z}⟩_{\mathcal{H}} := \sum_{x_i \in X, z_j \in Z} ⟨\ell_X(x_i), \ell_Z(z_j)⟩_I \cdot k(x_i, h^{-1}z_j) \quad (4)
\]

We use the kernel trick in machine learning [37, 61, 62] to substitute the inner products in (4) with the appearance kernel. After applying the kernel trick to (4), we get

\[
⟨f_X, f_{h^{-1}Z}⟩_{\mathcal{H}} = \sum_{x_i \in X, z_j \in Z} k_c(\ell_X(x_i), \ell_Z(z_j))k(x_i, h^{-1}z_j). \quad (5)
\]

We choose \(k_c\) to be the squared exponential kernel with real hyperparameters \(\sigma_c\) and \(\ell_c\), which are set independently.

**B. Feature Embedding via Tensor Product Representation**

We now extend the feature space to a hierarchical distributed representation. Let \((V_1, V_2, \ldots)\) be different inner product spaces describing different types of non geometric features of a point, such as color, intensity, and semantics. Their tensor product, \(V_1 \otimes V_2 \otimes \ldots\) is also an inner product space. For any \(x \in X, z \in Z\) with features \(\ell_X(x) = (u_1, u_2, \ldots)\) and \(\ell_Z(z) = (v_1, v_2, \ldots)\), with \(u_1, u_1 \in V_1, u_2, v_2 \in V_2, \ldots,\) we have

\[
⟨\ell_X(x), \ell_Z(z)⟩_I = ⟨u_1 \otimes u_2 \otimes \ldots, v_1 \otimes v_2 \otimes \ldots⟩ = ⟨u_1, v_1⟩ \cdot ⟨u_2, v_2⟩ \cdot \ldots \quad (6)
\]

By substituting (6) into (4), we obtain

\[
⟨f_X, f_{h^{-1}Z}⟩_{\mathcal{H}} = \sum_{x_i \in X, z_j \in Z} k(x_i, h^{-1}z_j) \prod_k k_{V_k}(u_{ki}, v_{kj})
\]

\[
:= \sum_{x_i \in X, z_j \in Z} k(x_i, h^{-1}z_j) \cdot c_{ij}, \quad (7)
\]

Equation (7) describes the full geometric and non-geometric relationship between the two point clouds. Each \(c_{ij}\) does not depend on the relative transformation, thus it will be a constant when computing the gradient and the step size. In our implementation, the double sum in (7) is sparse, because a point \(x_i \in X\) is far away from the majority of the points \(z_j \in Z\), either in the spatial (geometry) space or one of the feature (semantic) spaces.

This formulation can be further simplified to a purely geometric model, if we let the label functions \(\ell_X(x_i) = \ell_Z(z_j) = 1\). Then (7) becomes

\[
⟨f_X, f_{h^{-1}Z}⟩_{\mathcal{H}} = \sum_{x_i \in X, z_j \in Z} k(x_i, h^{-1}z_j). \quad (8)
\]

Through (8), the proposed method can register point clouds that do not have appearance measurements. It is worth noting that, when choosing the squared exponential kernel, (8) has the same formulation as Kernel Correlation [26].

**C. An Indicator of Alignment**

We want to have an indicator that represents the alignment of two point clouds \(X\) and \(Z\). An intrinsic metric available in our framework is the angle, \(\theta\), between two functions. This indicator can be computed to track the optimization progress. The cosine of the angle is defined as

\[
\cos(\theta) = \frac{⟨f_X, f_Z⟩_{\mathcal{H}}}{∥f_X∥ \cdot ∥f_Z∥}. \quad (9)
\]

However, calculating \(∥f_X∥\) and \(∥f_Z∥\) is time-consuming as it requires evaluating the double sum for each of the two point clouds. To approximate (9), we use the following result.

**Remark 1.** Suppose \(k(x_i, x_j) = δ_{ij}\) and \(c_{ii} = 1\), where \(δ_{ij}\) is the Kronecker delta, then \(∥f_X∥ = √∥X∥\).

**Corollary 1.** Using the previous assumption, we define the following alignment indicator.

\[
i_θ := \frac{1}{√∥X∥ ∥Z∥} \sum_{x_i \in X, z_j \in Z} c_{ij} \cdot k(x_i, z_j). \quad (10)
\]
The behavior of the alignment indicator with respect to the rotation and translation errors is shown in Fig. 3. We manually rotate and translate the same point cloud and then calculate the indicator with the original point cloud. A larger transformation results in a smaller indicator value. Furthermore, the maximum indicator value occurs when the transformation error is zero.

Remark 2. OverlapNet [63] uses a neural network to predict a similar metric and detect loop closures. The cosine of the angle in (9) or the indicator in (10) provide such a metric for self-supervised learning while taking into account the semantic information. Given the promising results of [63], the combination of our metric with deep learning is an interesting future research direction.

IV. CONSIDERATIONS FOR BOOSTING THE PERFORMANCE

The alignment indicator can guide the lengthscale update during the optimization. When the lengthscale is large, each point is associated with farther points, which provides the point cloud function representation a global perspective. When the lengthscale is small, each point is only connected to its closest neighbors, resulting in local attention for the registration. For a single registration process, we use larger lengthscales at early iterations. Every time the alignment indicator value at the current lengthscale stabilizes, we decay the lengthscale by two percent. This mechanism is inspired by the deterministic annealing process in some GMM-based registration methods [16], [21], [64], which perform a fixed lengthscale decay every few iterations.

The hyperparameters to be tuned include the lengthscale of the geometric kernels and the appearance (color and semantic) kernels. We use the same hyperparameters within a dataset. At the first frame of an entire data sequence, we initialize the transformation to be identity. We use a large lengthscale only for this single frame (0.95 in all the KITTI Stereo sequences), at the cost of more iterations. In subsequent frames, we use the previous estimated transformation as the initial value, accompanied by a smaller starting lengthscale (0.1 in all KITTI Stereo sequences).

To address the costly double sum computation, we downsample the raw inputs. We adopt the FAST feature detector [65] implemented in OpenCV [66]. We automatically control FAST’s threshold of the pixel intensity difference and disable the non-max suppression, so that the number of selected pixels with non-empty depth values is between 3k and 15k.

V. EXPERIMENTAL RESULTS

We now present the frame to frame registration experiments on both outdoor and indoor datasets: KITTI stereo odometry [67] and TUM RGB-D dataset [33].

A. Experimental Setup

All experiments are performed in a frame-to-frame manner without skipping images. The first frame’s transformation is initialized with identity, and all later frames start with the previous frames’ results. Hyperparameters for the proposed method on KITTI stereo and TUM RGB-D are available at [32]. The same values were used within one dataset.

On KITTI, our baselines are GICP [3], Multichannel-ICP [6], and 3D-NDT [19]. GICP and NDT are compared with our geometric method (Geometric CVO). Multichannel-ICP competes with our color-assisted method (Color CVO). GICP and 3D-NDT implementation are from PCL [68]. The Multichannel-ICP implementation is from [8]. Both the baselines and the proposed methods remove the first 100 rows of image pixels that mainly include sky pixels, as well as points that are more than 55 meters away. All the baselines use full point clouds without downsampling. The discussions of more candidate baselines and point selectors are in Sec. VI.

On TUM RGB-D, we use the same baselines for geometric registration as KITTI. We compare Color CVO with Dense Visual Odometry (DVO) [69] and Color ICP [14]. We reproduced DVO results with the code from [70] because the original DVO source code requires an outdated ROS dependency [71]. The Color ICP implementation is taken from Open3D [72]. The baselines use full point clouds.

B. Outdoor Stereo Camera: KITTI Stereo Odometry

We select a subset of pixels from KITTI’s stereo images via OpenCV’s FAST [65] feature detector. The depth values of the selected pixels are generated with ELAS [74]. The semantic predictions of the images come from Nvidia’s pre-trained neural network [11], which was trained on 200 labeled images. Examples of the point clouds are in Fig. 2. Noise from the estimated depth, from the color sensor, and from the semantic predictions are visible.

The result of Geometric, Color, and Semantic CVO and other baselines are provided in Table I. From sequence 00 to

---

Fig. 3: Indicator value with respect to rotation angle and translation distance for KITTI Stereo (left figures) and TUM RGB-D (right figures) sequences.
TABLE I: Results of the proposed frame-to-frame method using the KITTI [34] stereo odometry benchmark as evaluated on the average drift in translation, as a percentage (%), and rotation, in degrees per meter (°/m). The drifts are calculated for all possible subsequences of 100, 200, ..., 800 meters.

| Method | t (%) | r (°/m) | Avg | Std |
|--------|-------|---------|-----|-----|
| GeometricCVO | 4.06 | 0.0173 | 4.55 | 0.0226 |
| GICP [3] | 8.66 | 0.0361 | 11.23 | 0.0452 |
| 3D-NDT [19] | 7.53 | 0.0388 | 8.50 | 0.0396 |
| ColorCVO | 3.19 | 0.00125 | 3.69 | 0.00159 |
| MC-ICP [6] | 7.77 | 0.0387 | 14.10 | 0.0488 |
| SemanticCVO (with color) | 3.22 | 0.0126 | 3.64 | 0.00155 |

Fig. 4: An illustration of the proposed registration method on KITTI stereo sequence 01 (top) and 07 (bottom) versus the baselines. The black dashed trajectory is the ground truth. The dot-dashed trajectories are the baselines. Plotted with EVO [73].

Fig. 5: From top to down: the average translation errors and rotation errors on KITTI Stereo sequences 00 to 10 with respect to the distance segment and the moving speed, respectively.

10, our geometric method has a lower average translational error (4.55%) comparing to the GICP (11.23%) and NDT (8.50%). Our color version has a lower average translational drift (3.69%) than Multichannel-ICP (14.10%). If we add semantic information the error is further reduced (3.64%). The addition of color and semantic information also yields a lower standard deviation. Meanwhile, the average rotational drift of the proposed methods are smaller. Specifically, on the highway sequence (01) where the point cloud pattern becomes repetitive and noisy, both NDT and GICP perform poorly (as shown in Fig. 4). Figure 5 shows the average translational and rotational errors at different distances and speeds. The proposed methods show a more consistent high accuracy across different speeds.

On our desktop computer, excluding the image I/O and point cloud generation operations, the current GPU implementation takes an average 1.4 sec per frame on registering less than 15k points. GICP, NDT, Multichannel-ICP all use full point clouds (150k-350k points), and take 6.3 sec, 6.6 sec, and 57 sec per frame, respectively.

C. Indoor RGB-D Camera: TUM RGB-D Dataset

For TUM RGB-D, a semi-dense point cloud is generated from the depth images with FAST [65] feature selector. We evaluated our method on the fr1 sequences, which are
TABLE II: The RMSE of Relative Pose Error (RPE) for fr1 sequences. The Trans. columns show the RMSE of the translational drift in m/sec and the rot. columns show the RMSE of the rotational error in deg/sec.

| Sequence | Geometric CVO | GICP [3] | 3D-NDT [19] | Color CVO | DVO [69] | Color ICP [14] |
|----------|---------------|----------|-------------|-----------|---------|---------------|
| fr1/desk | 0.0493        | 0.2337   | 0.2040      | 0.0384    | 2.1422  | 0.0387        |
| fr1/desk2| 0.0545        | 0.2719   | 0.1823      | 0.0515    | 2.8967  | 0.0518        |
| fr1/room | 0.0565        | 2.2946   | 0.1719      | 0.0501    | 2.3366  | 0.0518        |
| fr1/360  | 0.1001        | 2.8686   | 0.2245      | 0.1021    | 3.1086  | 0.1602        |
| fr1/feddy| 0.1063        | 4.2122   | 0.2095      | 0.0668    | 2.6016  | 0.0948        |
| fr1/door | 0.2267        | 2.7345   | 0.5560      | 0.0697    | 2.3663  | 0.0635        |
| fr1/eye  | 0.0238        | 0.9748   | 0.1102      | 0.0270    | 1.3799  | 0.3272        |
| fr1/hip  | 0.0413        | 3.1806   | 0.2329      | 0.0501    | 3.6598  | 0.0336        |
| fr1/plant| 0.0388        | 1.9027   | 0.1335      | 0.0347    | 1.6451  | 0.0272        |
| Average  | 0.0730        | 2.3805   | 0.2290      | 0.0545    | 2.4333  | 0.0623        |

TABLE III: The RMSE of Relative Pose Error (RPE) for the structure v.s texture sequence. The Trans. columns show the RMSE of the translational drift in m/sec and the Rot. columns show the RMSE of the rotational error in deg/sec.

| Structure-Texture | Geometric CVO | GICP [3] | 3D-NDT [19] | Color CVO | DVO [69] | Color ICP [14] |
|-------------------|---------------|----------|-------------|-----------|---------|---------------|
| ×                 |               |          |             |           |         |               |
| ✓                 | 0.0267        | 0.8745   | 0.4586      | 0.0250    | 0.8201  | 0.0563        |
| ×                 | 0.0498        | 1.1602   | 0.2034      | 0.0591    | 1.1393  | 0.1612        |
| ×                 | 0.0338        | 2.4081   | 0.0993      | 0.0505    | 3.5577  | 0.1906        |
| ✓                 | 0.0376        | 1.2435   | 0.0981      | 0.0456    | 1.2239  | 0.1171        |
| ✓                 | 0.0238        | 1.3058   | 0.1082      | 0.0344    | 1.6899  | 0.0715        |
| ✓                 | 0.0288        | 0.9314   | 0.0717      | 0.0293    | 0.9516  | 0.0434        |
| ×                 | 0.0357        | 10.8878  | 0.5901      | 0.2143    | 8.9564  | 0.3506        |
| ×                 | 0.1287        | 4.0173   | 0.3722      | 0.1449    | 2.9821  | 0.1983        |
| Average           | 0.0794        | 2.8536   | 0.2487      | 0.0754    | 2.6651  | 0.1386        |

recorded in an office environment, and fr3 sequences, which contain image sequences in structured/nostructured and texture/notextured environments. TABLE II shows the results of fr1 sequences. Geometric CVO outperforms the baselines and achieves a similar performance to DVO. Moreover, with color information, the average error of CVO decreases. The results of fr3 sequences are shown in TABLE III. CVO outperforms the baselines. The overall result of Color CVO is better than Geometric CVO. However, Geometric CVO has lower translation errors in some sequence. This might because of the motion blur effect in the image, where color information is noisy due to rapid camera motion.

VI. DISCUSSIONS AND LIMITATIONS

Besides the reported baseline results, we tried to run Semantic ICP [7] and Color ICP [14] with KITTI’s full stereo point clouds as well. However, Semantic-ICP takes 4 – 8 min per frame on our machine, thus is infeasible to complete all the 23190 frames. The original Color ICP work has not been tuned for stereo data, and failed on KITTI sequence 00 and 08. We also tried to use the FAST point selector for all the baselines, but only GICP shows improvements, with 7.98% translation drift and 0.0362°/m rotation drift, versus our geometric result being 4.55% and 0.0236°/m.

We noticed that the point selector has a significant influence on the performance of the proposed methods. DSO’s semi-dense point selector in [75] was unable to complete some challenging sequences such as KITTI sequence 01. We cannot use PCL’s Voxel Filter [68] either because the original color and semantic information is lost during its downsampling. Only FAST [65] feature selector from OpenCV [66] works for all the datasets we tested. A future direction is to find a more robust downsampling scheme for this framework.

Moreover, the proposed methods’ performance relies on the geometric lengthscale during the optimization. Adaptive CVO [76] addresses the lengthscale decay by regarding it as a part of the optimizing variable. Still we need to manually choose an initial lengthscale. For inputs with larger accelerations, the lengthscale needs a global perspective for such abrupt changes. In this case, another future direction is an algorithmic way of selecting the initial lengthscale, or more broadly, studying the hyperparameter learning problem.

VII. CONCLUSION

We developed a nonparametric registration framework that integrates geometric and semantic measurements and does not require explicit data association. The proposed approach can utilize the extra visual and semantic information from modern range sensors while not being restricted by pairwise data correspondences. The novel hierarchical distributed representation of features via the tensor product representation provides a mathematically sound and systematic way of incorporating semantic knowledge into the point cloud model. The evaluations using publicly available stereo and RGB-D datasets show that the proposed method outperforms state-of-the-art outdoor and indoor frame-to-frame registration methods. We also provided an open-source GPU implementation.

In the future, we shall explore the connections of the developed framework with deep learning for representation learning in applications such as multi-modal feature learning, place recognition, and robust tracking and SLAM in harsh and visually degraded situations.
transform for fast and long range scan matching,” Journal of Intelligent & Robotic Systems, vol. 71, no. 1, pp. 85–108, 2013.

[48] A. Das and S. L. Waslander, “Scan registration using segmented region growing NDT,” Int. J. Robot. Res., vol. 33, no. 13, pp. 1645–1663, 2014.

[49] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2017, pp. 652–660.

[50] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, “PointNetLK: Robust & efficient point cloud registration using pointnet,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., June 2019.

[51] H. Su, V. Jampini, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, “SPLATNet: Sparse lattice networks for point cloud processing,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2530–2539.

[52] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” ACM Transactions on Graphics (TOG), 2019.

[53] V. Sarode, X. Li, H. Goforth, Y. Aoki, R. A. Srivatsan, S. Lucey, and H. Choset, “PointNet: Point cloud registration network using pointnet encoding,” ArXiv, vol. abs/1908.07906, 2019.

[54] Z. J. Yew and G. H. Lee, “3dfeat-net: Weakly supervised local 3d features for point cloud registration,” in Proc. European Conf. Comput. Vis. Springer, 2018, pp. 630–646.

[55] G. Elbaz, T. Avraham, and A. Fischer, “3d point cloud registration for localization using a deep neural network auto-encoder,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2017, pp. 4631–4640.

[56] Y. Wang and J. M. Solomon, “Deep closest point: Learning representations for point cloud registration,” in Proc. IEEE Conf. Comput. Vis. 2019, pp. 3523–3532.

[57] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” arXiv preprint arXiv:1611.01144, 2016.

[58] G. D. Pais, S. Ramalingam, V. M. Govindu, J. C. Nascimento, R. Chellappa, and P. Miraldo, “3DRegNet: A deep neural network for 3d point registration,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2020, pp. 7191–7201.

[59] W. Lu, G. Yan, Y. Zhou, X. Fu, P. Yuan, and S. Song, “Deepvcp: An end-to-end deep neural network for point cloud registration,” Proc. IEEE Int. Conf. Comput. Vis., pp. 12–21, 2019.

[60] C. Choy, W. Dong, and V. Koltun, “Deep global registration,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

[61] C. Rasmussen and C. Williams, Gaussian processes for machine learning. MIT press, 2006, vol. 1.

[62] K. P. Murphy, Machine learning: a probabilistic perspective. The MIT Press, 2012.

[63] X. Chen, T. Läbe, A. Milioto, T. Röbling, O. Vysotska, A. Haag, J. Behley, C. Stachniss, and F. Fraunhofer, “OverlapNet: Loop closing for LiDAR-based SLAM,” in Proc. Robot.: Sci. Syst, 2020.

[64] S. Granger and X. Pennec, “Multi-scale EM-ICP: A fast and robust approach for surface registration,” in Proc. European Conf. Comput. Vis. Springer, 2002, pp. 418–432.

[65] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in Proc. European Conf. Comput. Vis. Springer, 2006, pp. 430–443.

[66] G. Bradski, “The OpenCV Library,” Dr. Dobb’s Journal of Software Tools, 2000.

[67] T.-Y. Lin, W. Clark, R. M. Eustice, J. W. Grizzle, A. Bloch, and M. Ghaffari, “Adaptive continuous visual odometry from RGB-D images,” arXiv preprint arXiv:1910.00713, 2019.